Language Resources and Evaluation
https://doi.org/10.1007/s10579-024-09772-6

ORIGINAL PAPER

From greatest simplicity to full power

Research-Infrastructure-as-a-Service for language science and technology

Luís Gomes[1] · **António Branco**[1] · **João Silva**[1] · **Ruben Branco**[1]

Accepted: 26 August 2024

Abstract
While language processing services are key assets for the science and technology of language, the possible ways under which they may be made available to the widest range of their end users are critical to support an Open Science policy for this scientific domain. Although providing such processing services under some web-based interface, at large, offers itself as an immediate and cogent response to that challenge, turning this view into an effective access to language processing services is an undertaking deserving a clear conceptual direction and a corresponding robust empirical validation. Based on an extensive overview of major undertakings towards making language processing tools available and on the design principles worked out and implemented in the PORTULAN CLARIN infrastructure, in this paper we advocate for a Research-Infrastructure-as-a-Service (RIaaS) model. This model unleashes accessibility to language processing services in as many web-based interface modalities as the current stage of technological development permits to support, in order to serve as many types of end users as possible, from IT developers to Digital Humanities researchers, and including citizen scientists, teachers, students and digital artists among many others.

Keywords Language processing services · Software-as-a-service · Research-Infrastructure-as-a-Service · CLARIN · PORTULAN

✉ Luís Gomes
lmdgomes@fc.ul.pt

António Branco
ambranco@fc.ul.pt

João Silva
jrsilva@fc.ul.pt

Ruben Branco
rmbranco@fc.ul.pt

[1] NLX - Natural Language and Speech Group, Dept of Informatics, Faculty of Sciences, University of Lisbon, Campo Grande, 1749-016 Lisbon, Portugal

Published online: 12 October 2024

1 Introduction

Transcription, tokenization, syllabification, part-of-speech tagging, dependency parsing or text classification are just a few examples, among many others, of language related procedures that can be automatically delivered by language processing tools. It is at the core of the mission of an open research infrastructure for the field of language science and technology to serve its users with a web-based access to the functionalities of these tools that is as adequate as possible to their different needs and levels of expertise.

A full scope, trans-disciplinary research infrastructure that is committed to universal public service, must provide online language processing services that meet the widest usage needs. Therefore, the most demanding and inescapable requirement to be faced is to cater for all its target users, which have the widest array of backgrounds, with the most disparate technical and IT skills. These range from highly skilled computer science experts to mere laypersons and including reasonably able PhD researchers from non IT areas, among many other sorts of users such as citizen scientists, language teachers and professionals, scholars from the Humanities, software developers, students, digital artists, etc.

A first and natural move to address this challenging requirement is to seek to build the most simple user interface possible for language processing services. The assumption underlying this move is that if the laypersons are able to handle it, then it can also be handled by experts and is suitable for them, and thus processing services can be made available to all types of users.

No doubt that the reverse implication clearly does not hold, given that a layperson is not able to cope with most of what experts are able to handle in this respect. Nonetheless, the critical point here is that, *in practice*, the former implication does not hold either. As a matter of fact, to reduce the complexity of the user interface of a processing tool, one eventually has to reduce also its flexibility, its interoperability potential and combinatorial affordances, and its serendipity—in a nutshell, one has to give away its full power that could be explored by non-novice users.

Hence, the non-obvious and somewhat counterintuitive, yet crucial, starting point is to avoid funneling the offer of language processing services towards the least possible technically skilled users. Instead, the processing services should be offered under as many as possible different ways and interfaces permitted by technology, each being suited to users with different needs and different levels of skills in handling the underlying language processing tools.

Driven by the experience with our end users, in the PORTULAN CLARIN[1] research infrastructure,[2] we conceived and followed this design principle. By exploring different types of technologies and respective options, for each language processing tool, a number of different types of user interfaces were designed, developed

[1] https://portulanclarin.net.

[2] PORTULAN CLARIN Research Infrastructure for the Science and Technology of Language belongs to the Portuguese National Roadmap of Research Infrastructures of Strategic Relevance and is part of the international research infrastructure CLARIN ERIC (https://www.clarin.eu/). A detailed introduction can be found in Branco et al. (2020).

and made available that permit to maximize its exploitation and serve the widest usage needs and skills.

Based on this experience, in this paper we discuss the different types of interfaces for online language processing services that were developed and are offered to our users. Our goal is to share the solutions arrived at and the lessons learned in view of contributing to the set of best practices in what concerns such a crucial aspect in the mission of language technology platforms, that of delivering language processing capacity to their end users effectively.

In the next Sect. 2, we elaborate on related work and further motivate our proposed approach. In Sect. 3, we provide contextual information on the PORTULAN CLARIN research infrastructure and offer an introduction to the multi-interface approach for language processing services under a RIaaS model. In Sects. 4 through 7, each one of the proposed interface types are discussed in detail. In Sect. 8, we elaborate on an outlook of PORTULAN CLARIN, as an instance of the RIaaS model, with respect to sustainability, licensing and user-involvement. In the last Sect. 9, we offer concluding remarks and consider future work.

2 Related work

Language processing tools, made accessible either as pieces of software to be downloaded or via web-based interfaces, are language resources and a convenient way to get a survey on them is to consult the most prominent inventories and repositories of language resources. In inventories, language resources are described by metadata records; in repositories, distribution of copies of them is also ensured. Some inventories and repositories also provide interfaces to allow users to run language processing tools, which are frequently hosted in some other hosting infrastructure.

In the next two subsections, major inventories and repositories of language resources will be screened (Rehm et al., 2020). In a third subsection, two CLARIN platforms that provide prominent online language processing services will be also considered. The distribution platforms of different types addressed below result from an attempt to select a worldwide representative set. The discussion, in this section, refers to the status of inventories, repositories and infrastructures at the time of writing this paper, when they were screened, in September 2021.

2.1 Language processing services in inventories

Robert Dale's Catalogue To the best of our knowledge, the most comprehensive overview of worldwide commercial language processing web services is provided by Dale (2020). Currently in its third edition, dated from 2020, and announced to get the next updated edition by 2022, this publication provides a reasoned survey of professionally run web services. There one finds detailed information ranging from pricing to input/output formats, and including assessment reports based on the testing of the services, distilled into several comparative tables.

This survey covers software-as-a-service text analytics APIs from 34 vendors, ranging from big players and their products, like Amazon's Comprehend,[3] Google's Natural Language AI,[4] IBM's Watson Natural Language Understanding[5] or Microsoft's Text Analytics,[6] to a myriad of smaller companies seeking to make a business from this type of products.[7]

These services are made available behind a paywall, or at best under a freemium pricing model, and are restricted to web services targeting IT experts in what concerns the options for the provision of web-based interfaces to language processing tools. There are also other important differences to the wider RIaaS setup advocated in the present paper. The above services address more coarse grained downstream processing capabilities that are relevant for their commercial clients (e.g. sentiment analysis, summarization, language detection, etc.),[8] which while certainly being of interest also to the users of research infrastructures, leave aside the finer grained capabilities that are required for combinatorial affordances that leverage groundbreaking research.

CEF Catalogue Available since 2016, another important catalogue that covers web-based language processing services is the Catalogue of CEF eTranslation Services.[9] With its 693 entries,[10] differently from Dale (2020), it is a catalogue browsable online, it is open for its entries to be entered by the owners of the respective services being filed, and it includes no testing, assessment reports or comparative tables concerning their entries. Being a catalogue promoted by the services of the European Commission, who acts as a most generous funding agency of R& D activities in all scientific areas, including the field of language technology, this catalogue is likely seen by many of its depositors as an important display window when applying to its funding initiatives. The entries in this catalogue likely refer thus to artifacts of quite different levels of availability, technical maturity or promise. Many of them may be downstream applications or business solutions that eventually cover well more than just language processing, and when the language processing tools are actually provided, they may not necessarily being delivered via some sort of web-based interface.

[3] https://aws.amazon.com/comprehend/.

[4] https://cloud.google.com/natural-language/.

[5] https://www.ibm.com/cloud/watson-natural-language-understanding/.

[6] https://azure.microsoft.com/en-us/services/cognitive-services/text-analytics/.

[7] AI Applied, AmplifyReach, Aylien, Bitext, Cloudmersive, Codeq, ConversusAI, Dandelion, Essencient, Event Registry, Geneea, hyScore, Intellexer, Klangoo NLP, MeaningCloud, MonkeyLearn, Neticle, ParallelDots, Refinitiv Intelligent Tagging, Repustate, Rosette, Sapien, Semantria, SummarizeBot, text-2data, Textgain, TextRazor, Tisane, TXTWerk, Yonder.

[8] The overview in Dale (2020) isolates 10 types of capabilities that are common to most products, under which the comparative assessment of the latter are organized, along 227 of its 381 pages.

[9] https://cef-at-service-catalogue.eu/.

[10] There is a separate section in this catalogue with 11 extra entries concerning only the CEF services, provided by the Directorate-General for Translation of the European Commission. There is also a separate, sister catalogue, the ELRC-SHARE Catalogue (https://www.elrc-share.eu/repository/). Though it is aimed at covering data sets, some 5% of its 2204 entries concern language processing tools, which are basically delivered as software to be downloaded.

Most of the entries in this catalogue (581) are indicated as being provided "for a fee", and in many cases when pressing the associated button "Get Started with the service", one is just redirected to the general website of the respective company. As to the little more than 10% that are indicated as being "free of charge", almost half are categorized as being concerned with "translation", and the rest with "language technologies". The latter are mostly provided by universities or other research related organizations and more than one third of them is for English while the rest is for one of some other 35 idioms. A cursory inspection seems to indicate that, leaving aside those catalogue entries that are simply offering the option for the software tools to be downloaded, those other entries that point towards some web-based interface to a language processing tool do it typically via only one of the four types of possible web-based services that are described in Sects. 4 to 7—and that are all made available by our instance of RIaaS, to be described in detail in the Sect. 3. In any case, hardly any of them resort to the *file processing service* or to the *notebook service* modalities.

Other important inventories of language resources are the LRE Map,[11] organized by the European Language Resources Association,[12] and the Virtual Language Observatory,[13] organized by CLARIN—Common Language Resources and Technology Infrastructure.[14]

LRE Map Like in the Catalogue of CEF eTranslation Services, the entries in the LRE Map are contributed by the authors of the respective language resources. This inventory has been gathered with the help of the organizers of 26 computational linguistics conferences and workshops since 2010, including most of the more prominent ones in this field. The authors are required to describe the language resources associated to their papers at the time of the submission of their final version for publication.

Even though the last update of this inventory seems to have resulted from a conference already in 2016, to the best of our knowledge, it is the most extensive inventory of language resources related to published and peer-reviewed results, with its 6143 entries concerning over 100 idioms. As much as some 17% (1060) of these entries are categorized as "resource-tool", grouped into 20 types, one of them being "Web Services", with a residual set though with 9 entries. This is not entirely unexpected given that the language processing tools referred in published research papers concern pieces of software rather than web-based services.

CLARIN VLO As for the Virtual Language Observatory (VLO) Van Uytvanck et al. (2012), Goosen and Eckart (2014), its entries are contributed by the centers affiliated with CLARIN. Its records refer to 791,460 resources, of which 106 appear categorized as "webservice" and 868 as "software-webservice". A cursory inspection seem to indicate that these entries concern either online interfaces to browse some data sets, or web-based interfaces to language processing tools falling typically under one of the four types that is described in Sect. 7, namely as *web service*.

[11] http://lremap.elra.info/.

[12] http://www.elra.info/.

[13] https://vlo.clarin.eu/.

[14] https://www.clarin.eu/.

Like in the services covered in Dale (2020), while certainly being of interest also to the users of research infrastructures, the language processing services covered in the Catalogue of CEF eTranslation Services, the LRE Map and the Virtual Language Observatory, given their interfacing modalities, do not fully instantiate the concept of services for a RIaaS for the science and technology of language.

2.2 Language processing services from repositories

In addition to inventories of language processing services, like the ones just discussed, it is relevant to consider also repositories that are set up specifically for this type of web-based services.

Language Grid The Language Grid[15] is a prominent effort aiming at setting up a repository of web services for language processing, pioneered in Japan about a decade ago (Ishida, 2011). While the respective portal is still online and being run by a non-profit association since 2017, most of its developments took place in the late 2000s.

These language processing services are available to the members, both individual and institutional, of the respective association, upon the payment of an annual membership fee; they are fostered mostly around the goal of supporting machine translation; and they are provided under the form of web services targeting software developers and aimed at supporting downstream applications.

European Language Grid A recent initiative being pursued along this vein is the European Language Grid (ELG) (Rehm et al., 2020).[16] Like the Catalogue of CEF eTranslation, it hosts an inventory, and it also aims at acting as a repository, like the Language Grid.

At the time of writing this paper, around 21% (885) of its 4137 entries concern "tools/services", with some 40 input/output different capabilities, related to one of some 35 different idioms. A cursory browsing through the entries of these "tools/services" indicates that, besides their code eventually being available for download, they are made accessible under two types of web-based interfaces: the first, in the "try out" tab, implements the modality we term *online service* in the next Sect. 3, and the second, in the "code samples" tab, implements the modality we term *file processing*, also in the next section.

In the *online service* modality, at least for a number of annotation tools tested, there is only one output format available. In some cases, for instance in part of speech annotation tools, the output is presented in an interactive visualization that requires mouse clicks on tokens to reveal their annotations in a pop up window. Only one pop up window remains open at any time, and thus only the annotations of one token are visible at each time.

On the one hand, this interactive visualization enables a clean presentation of the text itself, which facilitates reading, but on the other hand, it poses a barrier to

[15] https://langrid.org/.

[16] https://www.european-language-grid.eu/.

accessing and interpreting the annotations by the users. If a user wants to see the annotations for a whole sentence at once, this visualization will not allow it.

The main point to be made here is not to criticise any particular output format or any of the tools available in this platform, but to emphasize the need for multiple output formats in an online service interface. Whenever multiple formats are available, users have a choice, and the scenarios where a user is not served with an adequate format given his expertise level are minimized. If there is only one format available, and it does not fulfill a particular user's needs, the user may give up, even though the underlying tool might have the functionality he needs.

In the *file processing* modality, in turn, the processing service is made available only as a `curl` command template to be entered in the command line interface in the local machine of the user after being instantiated with relevant values. As this requires swift IT skills well beyond the basic level, the range of possible end users becomes restricted.

The URL addresses of the web services corresponding to the processing tools in the repository are not indicated in their metadata records. The *web service* modality is possible, though one has to search for these addresses as one of the components of the curl commands expressions provided.

Additionally, in any of the modalities available, documentation to interpret the tags may not be readily available to the users in the respective entry of the repository, thus adding an extra barrier to its usability.[17]

To get access to the services in this ELG repository, a prospective user is required to register in order to be given credentials to login in the repository. While the services can be used for free at the time of writing the present paper, the financial viability and continuation of this repository after its ongoing 3 year implementation project being concluded had not been announced yet.[18]

Language Resource Switchboard Another repository of language processing services is the Language Resource Switchboard (Zinn, 2018). This repository is provided by CLARIN, which is legally an European Research Infrastructure Consortium (ERIC), founded in 2012 and supported by the membership fees of the European Union member States and organizations that belong to this consortium.

The repository contains 55 language processing services, grouped into 28 input/output capabilities, with some services concerning up to some 120 different idioms.

Though these services are not exploited commercially and no utilization fee is charged, as much as about 40% (22) of them are available only behind the CLARIN Service Provider Federation login wall, whose members includes typically universities and research related organizations. It is worth noting that it is also possible for individual researchers from non CLARIN countries to request for a CLARIN account. In any case, the requirement of a previous registration such that login and usage become possible turns out to be a usability barrier, and thus a negative incentive to the usage of the services by the widest range of potential users, including

[17] In a cursory visit at the time of writing the present paper, we found, for instance, that for GATE, there is no link to the respective documentation; there is a link for the documentation of Cogito but is empty; and for UDPipe, in turn, there is a link that works and leads to the documentation.

[18] https://www.european-language-grid.eu/.

all those not in the academic world. This is in contrast with the access policy to PORTULAN CLARIN, where the level of access is decided not for the entire infrastructure or repository, but for each resource based on the licensing assigned by the respective depositors—more on this in Sect. 8.

The web-based language processing services in the Language Resource Switchboard are delivered under the modalities that we term as *online service* and as *file processing* (cf. Sects. 4–7), in both cases through a user-friendly Graphical User Interface, displayed through under disparate interface layouts as the services are ensured by different members of CLARIN.

ELRA and LDC On a par with CLARIN, for the sake of completeness, it is worth mentioning two other major repositories of language resources that have been financially self-sustaining in the past decades, run by the ELRA Language Resources Association and by the Linguistic Data Consortium (LDC). With some 900 language resources, the operation of the LDC repository[19] is supported basically by the membership fees and sales revenues; while the ELRA repository, commercially run by ELDA,[20] with some 1400 entries, is supported basically by sales revenues and research-related activities. Unlike the CLARIN Switchboard, these repositories concern basically language data rather than language resources of other types, like software tools or web-based language processing services.

Hugging Face Hub Provided by Hugging Face,[21] a company set up in 2016, its repository of web-based language processing services, Hugging Face Hub, is devoted to language processing tools developed as neural network models.[22] At the time of writing, this repository includes 17,154 entries, concerning one of over 160 idioms. While Hugging Face is also the name of a Python library for models designed under the Transformer architecture, this repository accepts also other types of machine learning models.

These tools are made available as *web services*[23] for a dozen tasks[24] via paid subscription under layered pricing conditions, with access granted at no cost only for up to 30,000 input characters per month.

Each tool in the repository can be run as an *online service* for free. The output can be viewed in two ways, namely in JSON format, which is somewhat cumbersome and requires post processing to get at the actual information of relevance, and in highly user friendly format, with the tokens being coloured and the tags boxed, which eventually also makes it also somewhat cumbersome to copy and paste into a lean string format for subsequent processing and research purposes.

[19] https://www.ldc.upenn.edu/.

[20] http://catalog.elra.info/en-us/.

[21] https://huggingface.co/.

[22] https://huggingface.co/models/.

[23] https://huggingface.co/inference-api/.

[24] Automatic speech recogniton, feature extraction, text classification, NER, question answering, translation, summarization, text generation, zero-shot classification, conversational AI, table question answering.

A small portion of these processing tools are also made available at no cost as *notebook services*, in part contributed by the staff of this platform (14),[25] in part provided by external contributors (50),[26] concerning different types of input/ouput capabilities and downstream tasks.

2.3 Language processing services in research infrastructures

WebLicht WebLicht Hinrichs et al. (2010), Zinn and Campbell (2023) is a CLARIN web-based service for the automatic annotation of texts that is hosted and run by the national German node. It includes some language processing tools, such as tokenizers, part-of-speech taggers and parsers as webservices, and its major design desideratum is to allow users to concatenate these tools into processing chains, which can then be used to process user-supplied input.

The workflow of WebLicht starts with the specification of an input text, for which three options are available: (1) a short text snippet may be copied from a third party digital source and pasted into an input text box; (2) a sample text provided by WebLicht may be used; or (3) a file selected by the user may be uploaded. Next, two choices are offered: to use a pre-defined chain or to create a new chain. In both cases, the format of the input text is automatically identified and that narrows down the options shown to the user to include only chains or tools that can accept that format as input.

Chains are built by appending tools to the end of the chain, one by one. At each step of this process, the list of tools available that can be inserted at next chaining step is updated, on the basis of the output format of the previous tool chain. For example, part-of-speech taggers will appear in the list of available tools only after a tokenizer has been added to the chain.

This automatic filtering of the available tools for the next step helps users to build chains more efficiently but it still requires a considerable level of expertise. For example, to create a chain for dependency parsing, a user must know that the dependency parser requires part-of-speech tagging, which in turn requires tokenization. Thus, while the automatic filtering of available tools helps users to select a valid continuation for a chain, it does not help non-expert users in the task of building a chain that attains a certain desired goal, such as for instance dependency parsing. For non-experts, the construction of chains starting from the last tool in the chain and working backwards based on the input-format requirements of each tool would be more helpful.

The RIaaS model proposed in this article makes web-based language processing services available to users in a task-oriented manner. For example, if a user wants to perform dependency parsing, he only needs to provide the input in plain text format and, internally, the service will perform tokenization, part-of-speech tagging and then dependency parsing. This type of basic use is offered in two of the four types of interface proposed, namely *online services* and *file processing*—while advanced use

[25] https://huggingface.co/transformers/notebooks.html.

[26] https://huggingface.co/transformers/master/community.html#community-notebooks.

is also possible through the other two types of interface, namely *notebooks* and *web services*. This task-oriented design approach has some advantages with regards to a chain-oriented one.

An important advantage is that it allows simpler usage and thus caters to a wider range of users, including less skilled ones. Another advantage is that the internal tool chains supporting each service have been carefully constructed and optimized to give the best results for the task at hand.[27] An apparent drawback, in turn, is that, instead of using a graphical interface for building customized chains, users will need to write some code to interact with the corresponding webservices of each tool in the chain. However, it is worth noting that building custom chains is an advanced usage scenario, and users with this type of need are likely to have the skills to write code; or if they don't have it, they can be helped with a few code examples that they can then easily customize for their purposes.

LINDAT/CLARIAH-CZ CLARIN makes available a repository and natural language processing services (Hajič et al., 2022) hosted and run by LINDAT/CLARIAH-CZ, the national Czech node. These services can be accessed via three web-based interfaces, that we term *online services*, *file processing* and *web services*, and are presented to the user in a task-oriented manner, rather than in a chain-oriented fashion.

There are though important aspects that distinguish it from a full instantiation of the RIaaS model, as proposed in this paper. For instance, when a file is uploaded to be processed by the UDPipe tool,[28] the processing takes place in the same way as if the text had been pasted from the file into the input text box provided in its interface, independently of the size of that input file. This has sub-optimal implications.

First, when the file is large, it will take a long time to be processed and the user must keep the browser window open and the computer connected to the network until this processing is finished. Closing the browser window or refreshing the page will result in loss of any progress made thus far, as the output is returned only when the whole file has been processed.

Furthermore, since the result is returned and displayed inline in the browser window—instead of prompting the user to download a separate file, the size of the output can only be as large as the one supported by the browser and the hardware of the user, with the risk of filling all the available memory and slowing down less powerful machines.[29]

[27] An example of such optimizations is the following: in abstract, any part-of-speech tagger and dependency parser could be used together in a chain, provided that both use the same POS tagset. In practical terms, though, we know that training a dependency parsing model on POS tags produced by a tagger usually yields better performance, when both models, for tagging and parsing, are to be used together in a chain, than if the parser model had been trained with gold POS annotations. However, if the same parser is expected to be combined with several different taggers, then it may be preferable to train it on gold POS annotations. Mutatis mutandis this tends to happen with other types of chains as well.

[28] https://lindat.mff.cuni.cz/services/udpipe/.

[29] For example, the UDPipe tool page takes only 7MB of RAM in the Firefox browser, when loaded. However, when processing an input file with only 160KB of text, consisting of the first 1000 sentences of the Europarl English version, the memory usage of the page increases to about 600MB once the results are returned from the server and displayed in an HTML table.

Given these design choices, the *file processing* modality here is not viable to process large input data, thus limiting the range of possible users and usage scenarios. As discussed in Sect. 5, the file processing interface of the RIaaS model proposed in this paper caters for long processing jobs of large input data by notifying users upon job completion via an email containing an URL to download the processed data.

Other aspect of the RIaaS model proposed in this paper not present in LINDAT/CLARIAH-CZ is the systematic availability of Jupyter notebooks Jupyter et al. (2018), on a par with the other three web-based modalities of access to a given language processing service. Notebooks encourage and facilitate experimentation with webservice APIs.

Additionaly, the RIaaS model proposed in this paper also addresses the concerns with quality of service and fairness of concurrent access to the finite compute resources of the hosting infrastructure by means of usage quotas for webservices based on API-access keys. These keys or other access-control measures do not seem to be present in the LINDAT/CLARIAH-CZ services.

Summing up, the web-based language processing services that happen to be mentioned or provided by some of the different inventories, repositories and platforms just considered in this section lack some or many of the characteristics of the Research-Infrastructure-as-a-Service model proposed to support Open Science: some are commercial and do not provide services free of charge for research; others happen to be beyond pay or registration walls; some other yet are supported by short-term project grants, with its long-standing service unclear; and importantly, what is common to all those that are accessible for free, they offer limited or constrained web-based interface types, missing to be of interest for the widest range of potential users.

3 PORTULAN CLARIN

PORTULAN CLARIN Research Infrastructure for the Science and Technology of Language[30] belongs to the Portuguese National Roadmap of Research Infrastructures of Strategic Relevance, and it is part of the international research infrastructure CLARIN ERIC.

Its mission is to support researchers, innovators, citizen scientists, students, language professionals and users in general whose activities resort to research results from the Science and Technology of Language by means of the distribution of scientific resources, the supplying of technological support, the provision of consultancy, and the fostering of scientific dissemination.

It supports activities in all scientific and cultural domains with special relevance to those that are more directly concerned with language—whether as their immediate subject, or as an instrumental mean to address their topics, including among others, the areas of Humanities, Arts and Social Sciences, Artificial Intelligence, Computation and Cognitive Sciences, Healthcare, Language teaching and promotion, Cultural creativity, Cultural heritage, etc.

[30] https://portulanclarin.net.

The PORTULAN CLARIN infrastructure includes a Repository, and a Workbench. This paper will focus mostly on the latter.

3.1 Repository

PORTULAN CLARIN includes a repository of resources for the science and technology of language and related activities, whose interface allows users to search for resources using free text in combination with faceted filtering based on metadata attributes. The Repository also provides resource metadata, conforming to CMDI specification Windhouwer and Goosen (2022), through an OAI-PMH endpoint Lagoze et al. (2002), which is periodically harvested by the CLARIN Virtual Language Observatory (VLO) (Van Uytvanck et al., 2012; Goosen & Eckart, 2014). Thus, all resources in the PORTULAN CLARIN repository are also findable through VLO. The Repository holds a certification by CoreTrustSeal, based on requirements established by the World Data Systems (WDS) and the Data Seal of Approval (DSA).

To deposit resources in the repository, users receive a depositor account, which is created by contacting the PORTULAN CLARIN helpdesk. Throughout the deposit process, depositors may seek advice and support from the helpdesk in all aspects concerning resource deposition, including but not limited to file formats, licensing and metadata. The deposition process ends with a validation of resource metadata by the PORTULAN CLARIN staff and final publication, where a permanent identifier (PID) is assigned to the resource.

The license under which each resource is distributed is decided by the respective depositor. While most of the resources in the Repository have licenses that allow them to be downloaded by users, some resources have more restricted licenses. To get access to these resources, users must contact their depositors via a contact form. Striving to cater for the needs of the widest range of users, PORTULAN CLARIN does not limit the types of licenses under which depositors make their resources available. We find it is more beneficial to let users know about the existence of restricted resources and provide them a convenient way to reach out to the depositors and arrange for resource access, than presenting only resources with licenses that allow them to be downloaded directly from the Repository.

3.2 Workbench

While the Repository grants access to copies of scientific resources, thus including software and thus to copies of language processing tools, the Workbench, in turn, grants access to web-based interfaces of language processing services supported by some of such tools.

In the Workbench, at the basic level, a tool is accessible as what we termed simply as an *online service*. Users are presented with a web page with a text field. They just need to copy, from some third-party digital source, the excerpt of interest to be processed, paste it into that text field, click a button to run the service, then select and copy the result that will be displayed, and finally paste it to

some digital support. As its greatest advantage, this type of user interface has its unsurpassed simplicity and the fact that users can see the results of their requests immediately, and right away understand the essential functionality of the underlying tool. But this interface constrains users to work only with short sized inputs and provides no combinatorial affordances.

In a different user interface, a tool is accessible as what we termed a *file processing service*. Users find a dialog box that allows them to upload files of their choice, click the *upload* button below that box, and finally download the returned file containing the output. While still being served with no combinatorial affordances, users are no longer limited to short sized inputs, and for most practical purposes they will not feel a limitation to the eventual size of the data to be processed.

In another user interface, the functionality of a given tool is available as what we termed a *notebook service*. At a fundamental level, notebooks Jupyter et al. (2018) are documents that are both human-readable and machine-executable: they contain textual and graphical elements such as headings, paragraphs, lists, equations and figures, as well as executable code (e.g. Python code). While the general idea of human-readable documents that interweave text and machine-executable code descends from the *literate programming* paradigm introduced by Knuth (1984), notebooks further extend this paradigm by incorporating visualizations of the results produced by executing the code sections.

Therefore, a notebook can be simultaneously:

- an effective communication medium for describing an experiment to users;
- an executable computer program that implements an experiment;
- a medium for presenting the results of an experiment;
- a versatile sandbox to run experiments;
- a pedagogical instrument to support teaching and learning.

Furthermore, notebooks can be edited interactively, and the code sections re-run at will, making them ideal for experimentation. What is more, notebooks can also be opened in a browser and their code run online by resorting to some non-local server that otherwise would have to be provided by the user. Like in the previous type of interface, users are no longer limited to short inputs. But they have here the advantage that now combinatorial affordances are available by adjusting the code made available, for which though some minimal acquaintance with programming is already called for.

In yet another user interface that is even more demanding in terms of technical skills that are required for it to be resorted to, a tool can be used as a *web service* through a remote procedure call (RPC) interface. From within any piece of code, simple or sophisticated, written in any programming language of their preference, users can invoke a function to which they pass the input data to be processed and that returns the respective processed output. As its greatest advantage, this type of user interface grants users full flexibility and combinatorial affordances, but it requires sufficient programming skills, which for as much as minimal these may be, may eventually be an insurmountable hindrance for many users of the infrastructure.

Naturally, the processing tools can be used also by downloading them from the infrastructure repository and installing them in a local disk accessible to the user, as mentioned above. In this scenario, there is no infrastructure-specific interface provided to users to intermediate between the software tool and its functionality. This is the most demanding option, in terms of end user skills, as it requires considerable IT expertise, especially if the tool and its functionality is to be embedded into some software of interest.

At their root, the web-based interfaces to language processing hinted at above can be seen as instantiations of the Software-as-a-Service (SaaS) model. As part of cloud computing, SaaS eventually evolved to support multiple forms of more complex and sophisticated web-based services, including datacenter as a service (DCaaS), desktop as a service (DaaS), mobile backend as a service (MBaaS), among others.

In the next four sections below, each one of the four web-based services introduced above is presented in more detail.

4 Online services

Every tool in the PORTULAN CLARIN workbench has an *online service* type of interface. This provides the central entry point for each underlying tool, serving the following purposes:

1. to allow users to experiment with the tool by changing its input and options and immediately see the effect in the output;
2. to offer one-click usage examples to help users start experimenting with the least amount of effort;
3. to grant access to several forms of documentation;
4. to provide an entry point to the other types of web-based interfaces, viz. *file processing*, *notebook* or *web services*.

As an example, Fig. 1 presents the interface of the LX-DepParser tool,[31] which is a prototypical interface for sentence-based text-processing tasks, such as POS tagging, dependency and constituency parsing, or semantic role labeling, etc.

Every online service interface follows a common layout, which is sectioned vertically into 5 groups of elements, identified in Fig. 1 with the letters (a) to (e) for easier reference. At the topmost position, in group (a), we find a row of buttons that give access to examples, to the other types of interfaces (file processing, notebook and web services), and to documentation. The subsequent groups follow the order of user interaction with each of the interface elements: input for the tool is accepted in group (b); options affecting the behaviour and output format of the tool are specified in group (c); processing of current input is started or cleared in group (d); and finally, the results are shown in group (e).

[31] https://portulanclarin.net/workbench/lx-depparser/ (based on Branco et al. (2011)).

The "Examples" button is the first button on the interface, and thus one of the most prominent, because it provides the best starting point for newly-arrived users to start interacting with the tool. Running an example via a simple button click requires no effort from the users, while if the common practice of providing examples only as part of the documentation had been followed, users would be required to copy and paste inputs and options from the documentation into the interface.

The "File Processing", "Notebook" and "Web Service" buttons open, each, a dialog interface, which will be described in detail in Sects. 5, 6 and 7, respectively.

The documentation button opens a window that will be displayed over the interface, containing relevant information about the tool, such as:

- a description of the tool, the problem it solves and the method used;
- the datasets used to train the underlying models, when applicable;
- the tagsets used by the tool, when applicable;
- the input and output formats;
- a user manual or tutorial, in cases where the tool complexity justifies them;
- references to scientific publications describing the tool or its components;
- authorship and contact information;
- acknowledgements;
- licensing terms;
- and release info, when applicable.

The documentation window is presented in a modal form over the tool interface, which means that all page elements not belonging to the documentation window, will appear behind a semi-transparent gray background, allowing users to focus on the documentation without being disturbed by any other elements in the page.

Also, because the documentation is often a long document, an hyperlinked table of contents is displayed at the top of the window, allowing users to jump to any section. A floating button, with an upward-pointing arrow, appears at the top-left corner of the screen whenever the document is scrolled down. By clicking this button, users may jump back to the table of contents from any point in the document. These navigation aids are implemented in the interface logic that is shared across all tools in the workbench, contributing to a more uniform and thus less distracting user experience when reading documentation.

Besides being included in the main documentation, tagsets can also be accessed directly by clicking the respective "Tagset" button, the rightmost one in group (a) of Fig. 1. Once pressed, this button will slightly change its appearance to a depressed state and a new panel opens on the right-hand side of the interface, sharing half of the horizontal space that was previously fully dedicated to the interface, as shown in Fig. 2. Having the tagset shown side by side with the output of the tool is much more convenient to users (that need the tagset likely to analyze some output by the tool) than having to go back and forth between the documentation view and the interface.

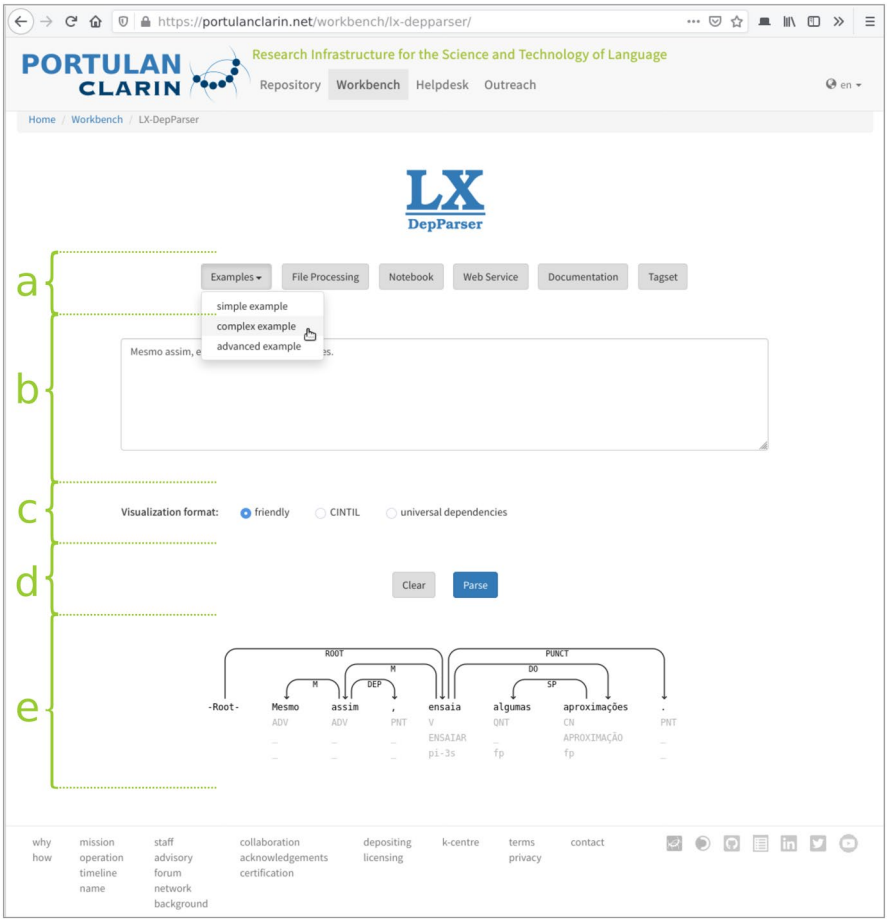

Fig. 1 Example of the online service interface. Our guidelines for positioning elements in the interface follows a top-down layout with 5 groups, (a) to (e) indicated in green (superimposed to this screenshot, and not part of the interface)

For some tools, instead of a tagset, this side panel may show other types of reference documentation, such as a cheat sheet for the query syntax, as it is the case, for instance, of the CINTIL Concordancer tool.[32]

Among the output formats of each tool there is usually one termed as *friendly*, which is the default option in this respect and is specifically targeted at human users, as opposed to being suited for further processing by another automatic tool. This friendly format is often graphical in nature, such as the dependency tree output in Fig. 1. By contrast, the other formats are generally textual, even if they encode some form of graph structure, and thus tend to be harder to interpret for humans, such as

[32] https://portulanclarin.net/workbench/cintil-concordancer/ (based on Barreto et al. (2006)).

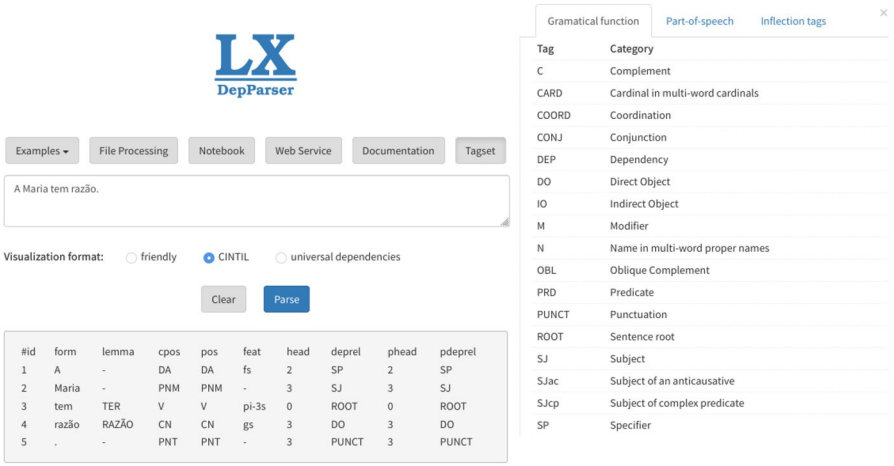

Fig. 2 Tagset of LX-DepParser shown side by side with the interface, for user's convenience. Also note that a different output format was selected from the one shown in Fig. 1

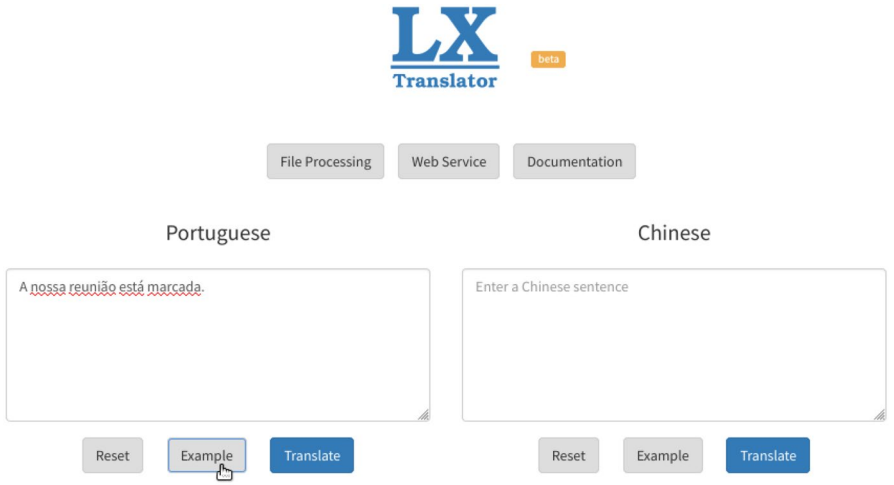

Fig. 3 Interface of the LX Translator online service, illustrating a case where the overall design guidelines may have to be weakened for the benefit of the interface usability, depending on the functionality of the service at stake

the tabular output shown within the gray rectangle in Fig. 2, which encodes a short sentence and its annotated dependency tree graph.

To conclude the present section, it is worth mentioning that the layout presented in Fig. 1 is a general guideline for organizing components in online service interfaces, which aims at increasing consistency across the interfaces of different tools, but ultimately, these guidelines should always be overridden as needed for the benefit of the interface.

Fig. 4 File processing service interface workflow. Depending on the size of the input file supplied by the ▶ user, the user interaction follows one of two main branches: (a) if the file is smaller than a fixed threshold, or (b) otherwise. The threshold size varies from one processing service to another and is determined as the average number of bytes that each specific service can process taking up to two minutes

For example, in the LX-Translator[33] online service, shown in Fig. 3, which is an interface for a bi-directional machine translation system, there is not one text input box but two of them, one for each language, displayed side by side. Each of these boxes is used for both input and output, which breaks the guideline of displaying the output on a dedicated area at the bottom of the page. At the beginning, both text boxes are empty and the user may input text in either one, click the "Translate" button below, and the translation will appear in the other box.

5 File processing services

The file processing interface, or *fileproc* for short, is a multi-step workflow that is launched by clicking on the "File Processing" button at the top of the online service interface. Figure 4 depicts this workflow, using screenshots of the dialog windows presented to the user at each step.

The first dialog window allows the user to select an input file from their computer to be processed and proceed to upload the file by clicking the "Upload" button. At this point, the workflow will take one of two possible courses, depending on the size of the file that is being uploaded: Small input files are handled by the path on left-hand side of Fig. 4, and we call these *short* (file processing) jobs; while large input files are handled by the path on right-hand side of Fig. 4, and we call these *long* jobs. The threshold size used to determine if a file is to be considered small or large is computed for each tool separately, based on the maximum amount of data that it can process under two minutes (this time interval will be clarified further ahead).

If the file is small enough so that it can be processed under two minutes, then we consider this to be a short job and processing will start immediately after the file is uploaded. The user is informed of the processing progress through a progress bar, as shown in step 2(a) of Fig. 4. As soon as the processing is complete, the user will be able to download the processed output files by clicking the "Download" button shown in window 3(a) of Fig. 4.

If the file being uploaded is large enough such that its processing time is estimated to be longer than two minutes, then the processing will take place in the background, without requiring the user to suspend other activities while waiting for its completion. Instead, in this type of job, when the processing is complete, the user will receive an email with a URL for downloading the output file.

Since users are not required to be registered (in order to foster the easy and open usage of the infrastructure), there is not any information about the user who is requesting this concrete file processing service. Thus, carrying on with the

[33] https://portulanclarin.net/workbench/lx-translator/ (based on Santos et al. (2019)).


File processing
1
Input format: Input files must be in .txt FORMAT with UTF-8 ENCODING and contain PORTUGUESE TEXT. Input files and folders can also be compressed to the .zip format.
Drag here
or
Click here to select from your computer
either a file or a compressed folder with files.
Privacy: The input file you upload and the respective output files will be automatically deleted from our computer after being processed and the result downloaded by you. No copies of your files will be retained after your use of this service.
Cancel
Upload
(a)
file smaller than
the threshold size
(b)
file larger than
the threshold size
Tracking
2(a)
Your file is being processed.
7%
Cancel
Email address validation
2(b)
To continue, please enter your email address.
A confirmation code will be sent to this address:
Email
Cancel
Continue
Download
3(a)
To download your processed file, press the download button below.
After download, your input and processed files were automatically deleted from our computer.
Cancel
Download
Email address validation
3(b)
To continue, please insert the code sent to luis.gomes@di.fc.ul.pt:
Code
This field is required.
Cancel
Continue
Processing your files
4(b)
Your files have been successfully uploaded and are being processed.
By email you will receive an URL address from which to download the result as soon as it will be concluded.
Close
5(b)
Subject: [PORTULAN CLARIN] Processing of your file is completed
From: <no-reply@portulanclarin.net>
To: <user@example.com>
Date: Sun, 16 May 2021 12:51:10 +0000
Greetings,
Your files test.txt have been processed with LXDEPPARSER and are ready to be downloaded at the following URL (which will be active for the next 5 days until the downloading is performed):
https://portulanclarin.net/workbench/lx-depparser/fileproc/553a61deac34a5d625/download
Your files will be deleted from our computer after their downloading is concluded, or after the deadline made available for it to be performed by you.
Best regards,
PORTULAN CLARIN Team
https://portulanclarin.net/

processing, it is necessary to know the email address where the message should be sent to. For this purpose, a simple email address validation method was implemented that sends an automatically generated code to the email address specified by the user in the dialog shown in screenshot 2(b), which should then be copied over by the user, from the email into a text field as shown in screenshot 3(b) of Fig. 4. Because the generated codes are long randomly generated strings, if the code inserted by the user matches the one that was sent, we assume that the user has access to the specified email account and did not guess the code by chance.

Once the user email address has been validated, the job processing begins. The user is notified that the job has been successfully submitted and that an email message will be sent upon the job completion, as illustrated in screenshot 4(b) of Fig. 4.

When the processing of a long job finishes, an email like the one shown in the 5(b) screenshot of Fig. 4 is sent to the user. The download URL included in the email message will be valid for 5 days. As soon as the user finishes downloading the output file, both the user's email address associated with the job and the output file will be deleted from the server (and thus the URL will no longer be valid). If, after 5 days since the email was sent, the user did not download the output file, it will be automatically removed from the server along with the user's email address.

Now that the workflow paths for short and long jobs have been presented, let us present the rationale for the two minute time threshold which is used to decide whether a file processing job should be considered short or long: two minutes is about the time after which we find it more costly, in terms of inconvenience to users, to require them to go through the extra steps of email address validation and then waiting for an email with the URL from which the output files can be downloaded, than simply wait for the processing to get completed.

Compared to the online service interface, presented in detail in the previous section, here in the file processing mode the user does not have to choose an output format. Instead, all possible output formats are included in the output file, which will be a zip archive containing one directory for each format. The reasoning for this decision is that the time required by a tool to process the input data largely exceeds the time required to convert the processed output into all available output formats. Thus, not only this is convenient for the users, which do not have to worry about which output format to choose, but it also avoids unnecessary re-processing of the same input data if a user finds out, after a job has been processed into one output format, that a different one is needed.

The accepted formats for the input file will depend on the tool at stake, but in general, the file should be either a UTF-8 encoded plain text file, or a zip archive containing any number of UTF-8 encoded plain text files. In the case of a zip archive, if the contained files are organized within a directory tree structure, this structure will be preserved during the processing.

The output file will always be a zip archive, containing several directories, one for each output format. If the input file was a zip archive containing multiple files organized within a directory tree structure, the same structure will be replicated under each output format directory.

6 Notebook services

The notebook interface, or *notebook* for short, is launched by clicking on the "Notebook" button at the top of the online service interface. The notebook interface is intended as an easy way for users to learn how to prototype their own language processing experiments. Figure 5 depicts part of the instructions on the steps to run it, using a partial screenshot of the window presented to the user.

While the online service and fileproc interfaces, introduced above, offer language processing services in a standalone manner, the notebook interface provides an easy path for users to explore the combinatorial affordances of web services—which will be discussed in the next section—in a non local fashion, by resorting to a browser for coding and to possible non local servers to run their respective coding.

The web-based notebook interface for the tools in the PORTULAN CLARIN workbench is based on Jupyter notebooks Jupyter et al. (2018) and offers its users different, local and non local, options to execute notebooks:

- launch notebooks on the Binder platform,[34] and make the code run remotely in the respective Jupyter Project's servers;
- launch notebooks on Google's Colab platform[35] and make the code run remotely in Google's servers.
- download notebooks and execute them locally on users' own computer.

To start using a notebook for any workbench tool that supports this type of interface, the user begins by clicking the "Notebook" button on the tool's online interface. A modal dialog is then presented to the user, similar to the one shown in Fig. 5. At the top of the dialog, the user is informed of any pre-requisites for executing the notebook, such as obtaining access keys for web services invoked from the notebook, and a brief description of the available options to execute the notebook is given. Following this informative section, there is a row of buttons, each for launching the notebook in a different manner. Below the buttons there is a preview of the notebook, allowing the user to quickly read through the notebook without launching it.

The options to launch notebooks on Binder and Colab provide users with the easiest and quickest way to start experimenting: by clicking on either of these buttons, the notebook for the tool under consideration will be automatically downloaded from the public PORTULAN CLARIN's GitHub repository[36] and launched on the respective notebook platform (Binder or Colab). Once launched, the user is able to modify and run the notebook at will and save any modifications (each platform provides a different set of options for saving notebooks).

The third option requires more expertise and work from users, which will have to install Python and Jupyter on their own computers to be able to execute the notebook. However, this option also presents some unique advantages:

[34] https://jupyter.org/binder.

[35] https://colab.research.google.com/.

[36] https://github.com/portulanclarin/jupyter-notebooks.

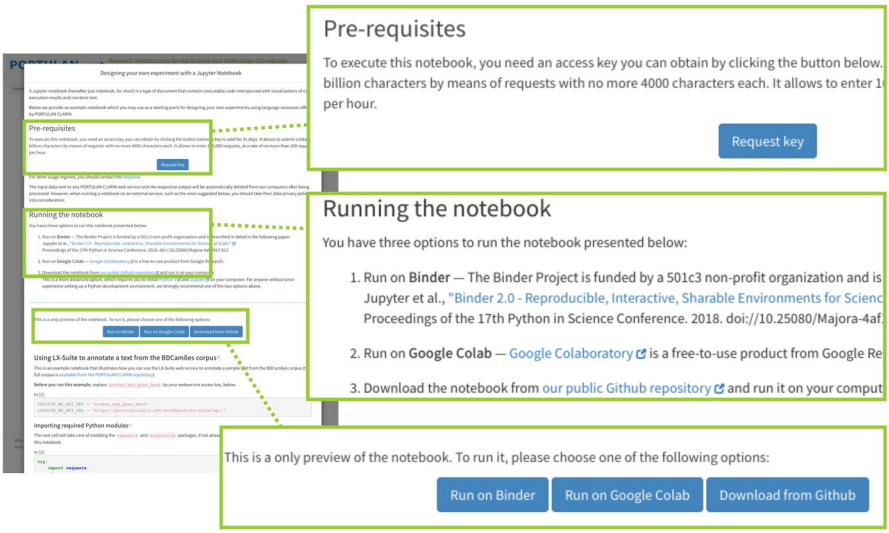

Fig. 5 Modal dialog of the notebook interface, with some sections zoomed in for readability

- it allows users to use software installed on their own computers that may not be available for installation on the notebooks when running on the Binder or Colab platforms;
- the input data to be processed as well as the output data is kept in the users' computers and is only shared with the PORTULAN CLARIN infrastructure, which has a very clear and user-minded data privacy policy: all data provided by the users is deleted after being processed;
- users do not have to abide to the usage terms and conditions required by the Binder and Colab platforms.

Notebooks are provided for several tools in the workbench, and each one has been written in a manner that plays to the strengths of this kind of interface. Instead of containing short and contrived pieces of code that merely show how to get output from each tool, the notebooks strive to show a concrete usage scenario of a workbench tool within an experiment, namely including code for downloading and pre-processing example data, code for processing the data, and code for some sort of subsequent analysis or graphical visualization of the result, as exemplified in Fig. 6.

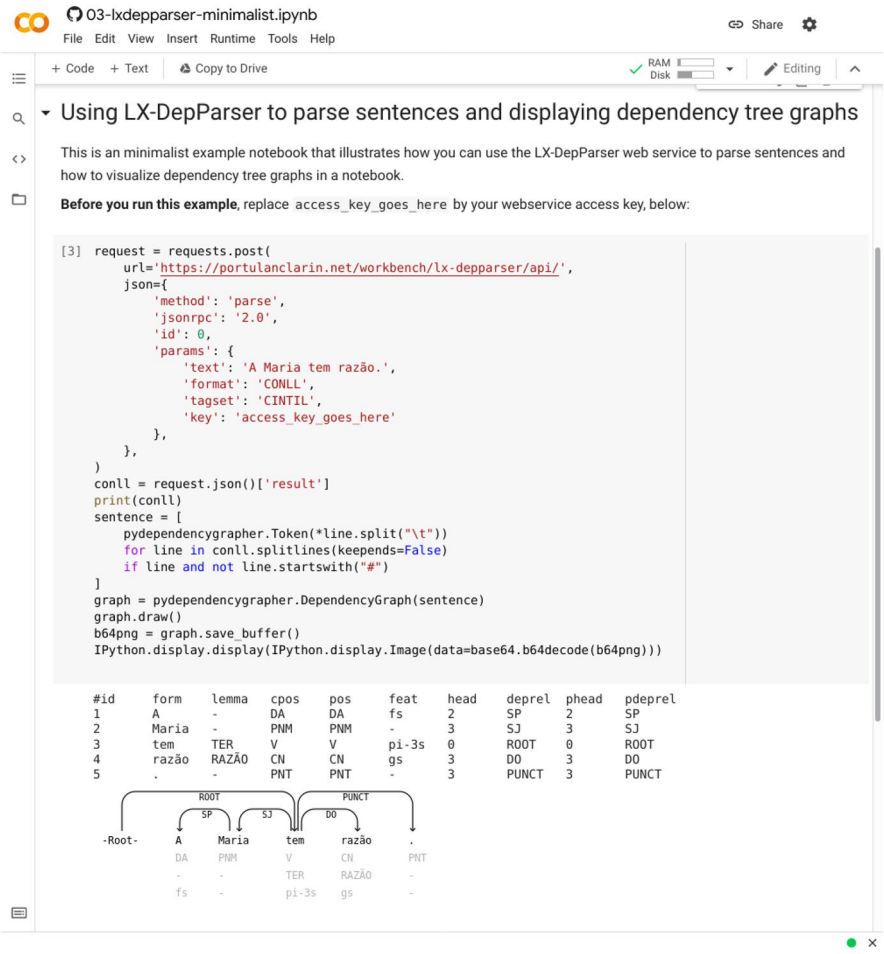

Fig. 6 Example snippet from notebook

7 Web services

The web services interface is a remote procedure call (RPC) type of interface, through which it is possible to interact with one or several tools in the infrastructure by means of computer programs.[37] We chose to implement this service using JSON-RPC, which is a lightweight, (programming-)language-agnostic protocol, for which implementations are readily available in many programming languages.

The web services interface is available for most tools in the workbench. Exceptions that do not offer this type of interface are, for example, tools that naturally

[37] An early approach to deliver language processing via web services can be found in Branco et al. (2008).

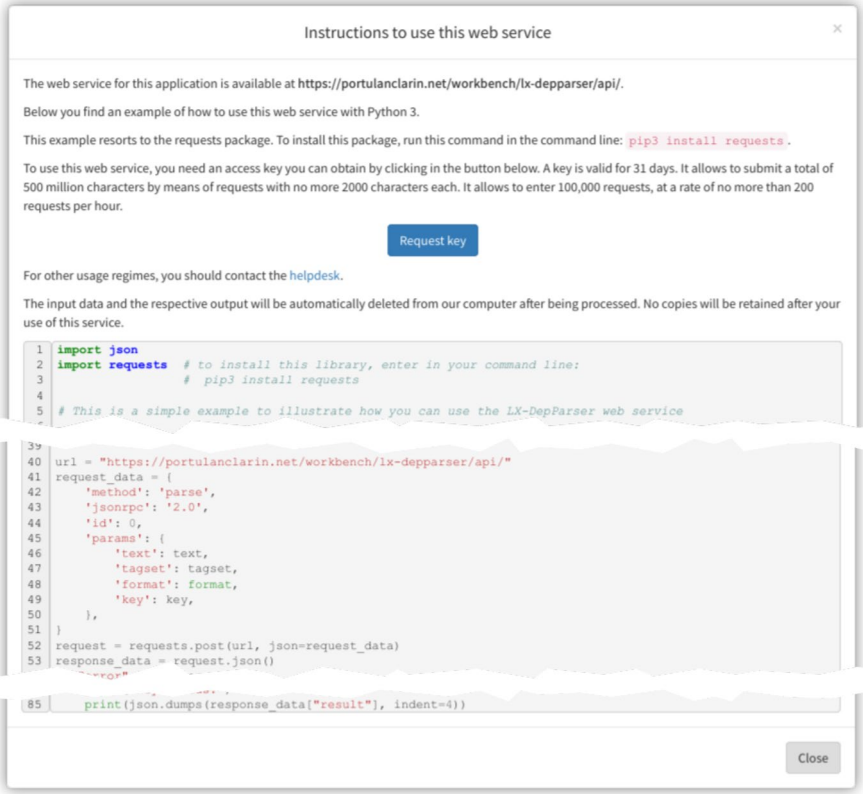

Fig. 7 Example web service dialog containing detailed instructions and example Python code (truncated in this screenshot) for using the LX-DepParser web service interface

lend themselves more to an interactive usage, through their online service interface, rather than to a data-processing usage scenario. The CINTIL Concordancer[38] and the CINTIL Treebank Searcher[39] are two examples of such tools.

To start using a web services, a user will click the "Web Service" button in the desired tool's online service interface, which will bring up a dialog as the one shown in Fig. 7. This dialog contains detailed information about the requirements that have to be met before this service can be used, as well as a simple and self-contained Python program that serves as an illustration of key commands and that can be used as a starting point for users with little programming experience to develop their own programs.

One of the requirements to use a web service is an access key that each user must obtain by clicking the "Request key" button on this dialog. This key is used to

[38] https://portulanclarin.net/workbench/cintil-concordancer/ (based on Barreto et al. (2006)).

[39] https://portulanclarin.net/workbench/cintil-treebank-searcher/ (based on Branco et al. (2010)).

implement a basic access control mechanism with the purpose of preventing abusive use, by any individual user, either intentionally or inadvertently, of the finite computational resources available on PORTULAN CLARIN to serve all its users. By clicking on the "Request key" button, users will go through an email validation process identical to the one that is required for submitting long file processing jobs, described in Sect. 5. After their email address has been validated, users are sent an email with an access key and information about usage quotas associated with it, namely the total number of requests allowed, the total number of characters allowed (accumulated over all requests), and the expiry date for the key.

Whenever a user requests a new key using an email address that was used before, if the previous key is still valid, i.e. has not expired and its usage quotas have not been exhausted, that previous key is returned in the response email, along with the remainder usage quota. Thus, at any point in time, only one valid key is associated with any given email address.

Because any user can have access to several email addresses, this access control mechanism does not prevent a single user from having multiple access keys, each associated with a different address. However, creating new email addresses and requesting access keys requires some effort which should be enough to discourage fortuitous abuse.

Besides the total number of requests and of characters allowed during the lifespan of a key, there is also a maximum number of requests and characters allowed per a defined period of time. If any of these maximum rates are reached, subsequent requests will receive an appropriate error code and message, until enough time has passed since the last successful request such that all rates become lower than their maximum allowed values.[40]

8 Outlook

PORTULAN CLARIN[41] belongs to the Portuguese National Roadmap of Research Infrastructures of Strategic Relevance[42] and is part of the international research infrastructure CLARIN ERIC Eskevich et al. (2020), de Jong et al. (2022), which is a landmark infrastructure in the European Strategy Forum on Research Infrastructures (ESFRI) Roadmap ESFRI (2020, 2021), and contributing towards European Open Science Cloud (EOSC) (Jones et al., 2021). Accordingly, it is supported by FCT-Fundação para a Ciência e Tecnologia,[43] the Portuguese national funding agency for R& D, in the scope of the long-term national research infrastructure undertaking. Additionally, while PORTULAN CLARIN web-based language processing services are running under the RIaaS model to leverage Open Science, they are available also to support commercial purposes under specific for-a-fee conditions.

[40] Requests from users that for their research purposes, exceptionally need to go beyond default usage quotas are supported on a case-by-case basis.

[41] https://portulanclarin.net.

[42] https://www.fct.pt/apoios/equipamento/roteiro/index.phtml.en.

[43] https://www.fct.pt/.

Its Repository is aimed at supporting the widest possible range of users. It is thus open to distributing all types of language resources, with all types of licenses, under which the respective authors may decide to distribute them. As the PORTULAN CLARIN infrastructure is committed to Open Science, while this approach does not hamper every free resource to be distributed, it permits resources with other types of licenses to be also distributed—our experience let us understand that many times this is a first step for the respective authors eventually to change these licenses to terms more aligned with the aims of Open Science.

For many of the language processing tools that underlie the services in the PORTULAN CLARIN Workbench, which are accessible by means of different web-based interfaces each, their authors happen also to distribute the respective code and/or compiled binaries through the PORTULAN CLARIN Repository. Like any other resource distributed through this repository, the licenses under which these tools are distributed are decided by the respective depositor.

For the language processing services in the Workbench, in turn, they can be used under a license that is common to all of them and that is indicated in the respective documentation pages. Basically, this license exempts PORTULAN CLARIN from liability when users resort to these services, requires attribution of authorship, etc., and restricts commercial usage, among other conditions.[44]

As for the code of the workbench itself, it is worth noting that it is quite tied with the integrated tools and services. As such, the potential scenarios for its reuse elsewhere may be limited. Still, the source code for the workbench is also made available for interested parties upon request.

The maintenance and development of a research infrastructure is, by its intrinsic nature, an ongoing endeavor, and there are a few priority aspects where the current implementation of the PORTULAN CLARIN workbench will be improved, both in terms of hardware and software. First, the hardware is limited in terms of available GPUs for running neural models. These models are ubiquitous in all areas of language processing, and thus GPUs are in very high demand, and will be expanded in our hardware setup.

On the software side, one area for improvement is related to metadata. Presently, we provide metadata and persistent identifiers (PIDs) for all resources in the repository. The repository pages containing metadata and PIDs for the tools underlying each workbench service are linked from its documentation page.[45] However, it would also be desirable to have metadata and PIDs for the workbench services themselves, for the sake of the reproducibility of scientific results based on them. Versioning of services, however, is not a trivial matter. For several reasons, security being the top one, all components in the software stack of any service made available on the web should be kept as up-to-date as possible. Because any component in the stack may, directly or indirectly, impact the behaviour of a service, any versioning scheme adopted for workbench services should take into account the versions of all components in the software stack. Thus, the following question arises: how to harmonize the persistent nature of PIDs and associated immutable metadata, which

[44] The license is available at https://portulanclarin.net/workbench/license/.

[45] For all tools whose authors have decided to make them available in the repository.

obviously includes the service version number, and a software stack that must be kept up to date? This topic is presently under discussion by the Standing Committee for CLARIN Technical Centres (SCCTC) aiming at progressing towards a consensus on how to cope with this issue.

Taking into account the usage of the services provided, in turn, alongside gathering user feedback through the helpdesk service, user-involvement activities are undertaken in order to disseminate the PORTULAN CLARIN infrastructure, in general, and its web-based language processing services, in particular.[46] The collected feedback is used to improve and extend all services on offer by the infrastructure, including the workbench and the repository.

Aside from fixing bugs and minor interface refinements to improve usability, user feedback is also valuable to prioritize development efforts. As an example, upon learning from a group of users that transcription of speech into text is a service in high demand, we prioritized the addition of that service to our workbench. Presently, transcription is one of the most used services in the workbench.

Looking forward, we envision the organization of dissemination workshops and training activities, giving priority to those users that have lower technical expertise in using language processing tools.

9 Conclusion

In this paper, we have advocated for the Research-Infrastructure-as-a-Service model as an approach that permits to leverage the full potential of language processing services, aiming at promoting an effective Open Science policy with respect to the area of language science and technology.

To empirically illustrate this vision, we described the multi-interface approach implemented at PORTULAN CLARIN, that we believe opens up language processing services to a wider range of users, coming from and carrying the most diverse backgrounds and motivations. We argued against making language processing services available through a single interface, designed with a specific user profile in mind, which would necessarily be too inflexible for some users or too complex for some others. Differently from the language processing services surveyed, we propose as many different types of web-based interfaces as the current level of technological development permits, each one demanding an increased level of technical skill from the users, and progressively empowering them in return with wider combinatorial affordances and strengthened research and development capabilities.

The most basic type of interface, which we termed *online service*, is designed to be attractive and to invite users to self exploration, for example by allowing to run one-button-click examples. The second type of interface, termed *file processing*, allows users to upload large input files and have them processed with minimal effort.

[46] Two examples, among other cases, are a master class for PhD students from the area of Social Sciences and the Humanities from the NOVA University of Lisbon, and the presentation of the web-based language processing services at the 2nd Cool Tools Workshop (https://cooltools.tecnico.ulisboa.pt/).

The third type, *notebook service*, requires at least a basic knowledge of programming, but enhances a quite powerful research asset with minimal computational resources from the side of the user. The fourth, and most technically demanding but also the most empowering interface, termed *web service*, is a remote procedure call interface to be used from within a computer program.

Turning to future work, it is worth mentioning that the processing tools underlying the web-based services, together with their companion data sets, are made available by the respective authors that decided so through the PORTULAN CLARIN Repository, and are thus accessible from the CLARIN VLO inventory once all entries in this Repository are accessible from that inventory. Additionally, the work on the web-based services is underway to make them available through the CLARIN Switchboard (Zinn, 2018) as well.

As mentioned in Sect. 8, metadata and PIDs for language processing services is a topic undergoing discussion by the SCCTC. Once this discussion converges into practical guidelines, we will align our implementation with those.

As the technology keeps evolving, certainly further types of web-based interfaces will emerge in the future, which should become integrated in language processing facilities offered by research infrastructures if these will want to keep pursuing their mission. Given the point argued for and the empirical validation presented in this paper, we believe that the model of Research-Infrastructure-as-a-Service will provide the best conceptual ground to ensure the more effective direction to this evolution.

Acknowledgements The results reported here were partially supported by PORTULAN CLARIN-Research Infrastructure for the Science and Technology of Language, funded by Lisboa2020, Alentejo2020 and FCT-Fundação para a Ciência e Tecnologia under the Grant PINFRA/22117/2016. The PORTULAN CLARIN Workbench comprises a number of tools that are based on a large body of research work contributed by different authors and teams, which continues to grow and is acknowledged here: (Barreto et al., 2006; Branco et al., 2010, 2011, 2012, 2014; Cruz et al., 2018; Veiga et al., 2011; Branco & Henriques, 2003; Silva et al., 2009; Rodrigues et al., 2016, 2020; Branco & Silva 2006; Costa & Branco, 2012; Santos et al., 2019; Miranda et al., 2011).

Funding Open access funding provided by FCT|FCCN (b-on).

Declarations

Conflict of interest The authors are (or have been) part of the PORTULAN CLARIN staff, with the following roles: Luís Gomes as technical manager, António Branco as the director general, João Silva as scientific resources and users support manager, and Ruben Branco as developer.

References

Barreto, F., Branco, A., Ferreira, E., Mendes, A., Bacelar do Nascimento, M. F., Nunes, F., & Silva, J. R. (2006). Open resources and tools for the shallow processing of Portuguese: The TagShare project. In *Proceedings of the 5th international conference on language resources and evaluation (LREC)* (pp. 1438–1443).

Branco, A., Castro, S., Silva, F., & Costa, F. (2011). CINTIL DepBank handbook: Design options for the representation of grammatical dependencies. Technical Report DI-FCUL-TR-2011-03, University of Lisbon.

Branco, A., Costa, F., Martins, P., Nunes, F., Silva, J., & Silveira, S. (2008). LXService: Web services of language technology for Portuguese. In *Proceedings of the 6th international conference on language resources and evaluation (LREC)* (pp. 2577–2583).

Branco, A., Costa, F., Silva, J., Silveira, S., Castro, S., Avelãs, M., Pinto, C., & Graça, J. (2010). Developing a deep linguistic databank supporting a collection of treebanks: the CINTIL DeepGramBank. In *Proceedings of the 7th international conference on language resources and evaluation (LREC)* (pp. 1810–1815).

Branco, A., & Henriques, T. (2003). Aspects of verbal inflection and lemmatization: Generalizations and algorithms. In *Proceedings of XVIII annual meeting of the portuguese association of linguistics (APL)* (pp. 201–210).

Branco, A., Mendes, A., Quaresma, P., Gomes, L., Silva, J., & Teixeira, A. (2020). Infrastructure for the science and technology of language PORTULAN CLARIN. In *Proceedings of the 1st international workshop on language technology platforms* (pp. 1–7). European Language Resources Association.

Branco, A., & Nunes, F. (2012). Verb analysis in a highly inflective language with an MFF algorithm. In *Proceedings of the 11th international conference on the computational processing of Portuguese (PROPOR)*, number 7243 in Lecture Notes in Artificial Intelligence (pp. 1–11). Springer.

Branco, A., Rodrigues, J., Silva, J., Costa, F., & Vaz, R. (2014). Assessing automatic text classification for interactive language learning. In *Proceedings of the IEEE international conference on information society (iSociety)* (pp. 72–80).

Branco, A., & Silva, J. (2006). A suite of shallow processing tools for Portuguese: LX-Suite. In *Proceedings of the 11th conference of the European chapter of the association for computational linguistics (EACL)* (pp. 179–182).

Costa, F., & Branco, A. (2012). Aspectual type and temporal relation classification. In *Proceedings of the 13th conference of the European chapter of the association for computational linguistics* (pp. 266–275).

Cruz, A. F., Rocha, G., & Cardoso, H. L.. (2018). Exploring Spanish corpora for Portuguese coreference resolution. In *2018 fifth international conference on social networks analysis, management and security (SNAMS)* (pp. 290–295).

Dale, R. (2020). *Text analytics APIs: A consumer guide*. The Language Technology Group.

de Jong, F., Van Uytvanck, D., Frontini, F., van den Bosch, A., Fišer, D., & Witt, A. (2022). Language matters. The European research infrastructure clarin, today and tomorrow. In D. Fišer & A. Witt, (Eds.), *CLARIN. The infrastructure for language resources* (pp. 31 – 57). de Gruyter.

Eskevich, M., de Jong, F., König, A., Fišer, D., Van Uytvanck, D., Aalto, T., Borin, L., Gerassimenko, O., Hajic, J., van den Heuvel, H., Kahusk, N., Liin, K., Matthiesen, M., Piperidis, S., & Vider, K. (2020). CLARIN: Distributed language resources and technology in a European infrastructure. In *Proceedings of the 1st international workshop on language technology platforms* (pp. 28–34). European Language Resources Association.

European Strategy Forum on Research Infrastructures (ESFRI). (2020). Making science happen—a new ambition for research infrastructures in the European research area. White paper.

European Strategy Forum on Research Infrastructures (ESFRI). (2021). Roadmap 2021—strategy report on research infrastructures. White paper.

Goosen, T., & Eckart, T. (2014). Virtual language observatory 3.0: What's new. In *CLARIN annual conference*.

Hajič, J., Hajičovà, E., Hladkà, B., Mišutka, J., Košarko, O., & Straňàk, P. (2022). Lindat/clariah-cz: Where we are and where we go. In D. Fišer & A. Witt (Eds.), *CLARIN—the infrastructure for language resources* (pp. 61–82). De Gruyter.

Hinrichs, E. W., Hinrichs, M., & Zastrow, T. (2010). WebLicht: Web-based LRT services for German. In *Proceedings of the ACL 2010 system demonstrations* (pp. 25–29).

Ishida, T. (Ed.). (2011). *The language grid: Service-oriented collective intelligence for language resource interoperability*. Springer.

Jones, S., Dimper, R., Dillo, I., Hanahoe, H., & Kurapati, S. (2021). Making the European Open Science Cloud (EOSC) work: Where to go from here?.

Jupyter, P., Bussonnier, M., Forde, J., Freeman, J., Granger, B., Head, T., Holdgraf, C., Kelley, K., Nalvarte, G., Osheroff, A., Pacer, M., Panda, Y., Perez, F., Ragan-Kelley, B., & Willing, C. (2018). Binder 2.0—reproducible, interactive, sharable environments for science at scale. In F. Akici, D. Lippa, D. Niederhut, & M. Pacer (Eds.), *Proceedings of the 17th Python in science conference* (pp. 113–120).

Knuth, D. E. (1984). Literate programming. *The Computer Journal, 27*(2), 97–111.

Lagoze, C., Van de Sompel, H., Nelson, M., & Warner, S. (2002). Open archives initiative-protocol for metadata harvesting-v. 2.0. http://www.openarchives.org/OAI/openarchivesprotocol.html

Miranda, N., Raminhos, R., Seabra, P., Sequeira, J., Gonçalves, T., & Quaresma, P. (2011). Named entity recognition using machine learning techniques. In *EPIA-11, 15th Portuguese conference on artificial intelligence* (pp. 818–831).

Rehm, G., Berger, M., Elsholz, E., Hegele, S., Kintzel, F., Marheinecke, K., Piperidis, S., Deligiannis, M., Galanis, D., Gkirtzou, K., Labropoulou, P., Bontcheva, K., Jones, D., Roberts, I., Hajič, J., Hamrlová, J., Kačena, L., Choukri, K., Arranz, V., Vasiļjevs, A., Anvari, O., Lagzdiņš, A., Meļņika, J., Backfried, G., Dikici, E., Janosik, M., Prinz, K., Prinz, C., Stampler, S., Thomas-Aniola, D., Gómez-Pérez, J. M., Silva, A. G., Berrío, C., Germann, U., Renals, S., & Klejch, O. (2020). European language grid: An overview. In *Proceedings of the twelfth language resources and evaluation conference* (pp. 3366–3380). European Language Resources Association.

Rehm, G., Marheinecke, K., Hegele, S., Piperidis, S., Bontcheva, K., Hajič, J., Choukri, K., Vasiļjevs, A., Backfried, G., Prinz, C., Gómez-Pérez, J. M., Meertens, L., Lukowicz, P., van Genabith, J., Lösch, A., Slusallek, P., Irgens, M., Gatellier, P., Köhler, J., Le Bars, L., Anastasiou, D., Auksoriūtė, A., Bel, N., Branco, A., Budin, G., Daelemans, W., De Smedt, K., Garabík, R., Gavriilidou, M., Gromann, D., Koeva, S., Krek, S., Krstev, C., Lindén, K., Magnini, B., Odijk, J., Ogrodniczuk, M., Rögnvaldsson, E., Rosner, M., Pedersen, B., Skadiņa, I., Tadić, M., Tufis, D., Váradi, T., Vider, K., Way, A., & Yvon, F. (2020). The European language technology landscape in 2020: Language-centric and human-centric AI for cross-cultural communication in multilingual Europe. In *Proceedings of the Twelfth Language Resources and Evaluation Conference* (pp. 3322–3332). European Language Resources Association.

Rodrigues, J., Branco, A., Neale, S., & Silva, J. (2016). LX-DSemVectors: Distributional semantics models for the Portuguese language. In *Proceedings of the 12th international conference on the computational processing of Portuguese (PROPOR'16)* (pp. 259–270).

Rodrigues, J., Costa, F., Silva, J., & Branco, A. (2020). Automatic syllabification of Portuguese. *Revista da Associação Portuguesa de Linguística*, (1).

Santos, R., Silva, J., Branco, A., & Xiong, D. (2019). The direct path may not be the best: Portuguese-Chinese neural machine translation. In *Proceedings of the 19th EPIA conference on artificial intelligence* (pp. 757–768).

Silva, J., Branco, A., Castro, S., & Reis, R. (2009). Out-of-the-box robust parsing of Portuguese. In *Proceedings of the 9th international conference on language resources and evaluation (LREC)* (pp. 75–85).

Van Uytvanck, D., Stehouwer, H., & Lampen, L. (2012). Semantic metadata mapping in practice: the virtual language observatory. In *LREC 2012: 8th international conference on language resources and evaluation*, (pp. 1029–1034). European Language Resources Association (ELRA).

Veiga, A., Candeias, S., & Perdigão, F. (2011). Generating a pronunciation dictionary for European Portuguese using a joint-sequence model with embedded stress assignment. In *Proceedings of the 8th Brazilian symposium in information and human language technology*.

Windhouwer, M., & Goosen., T. (2022). Component metadata infrastructure. In D. Fišer & A. Witt, (Eds.), *CLARIN—the infrastructure for language resources* (pp. 191–222). De Gruyter .

Zinn, C. (2018). The language resource switchboard. *Computational Linguistics, 44*(4), 631–639.

Zinn, C., & Campbell, B. (2023). Weblicht-batch—a web-based interface for batch processing large input with the weblicht workflow engine. In *CLARIN annual conference* (pp. 133–141).