TELP – Text Extraction with Linguistic Patterns

João Cordeiro[1], **Purificação Silvano**[2], **António Leal**[2], **Sebastião Pais**[1]
[1]University of Beira Interior and NOVA LINCS, [2]University of Porto and CLUP
[1]Caminho do Biribau, 6200-060 Covilhã, Portugal
[2]Via Panorâmica, s/n, 4150-564 Porto, Portugal
[1]{jpcc, sebastiao}@ubi.pt
[2]{msilvano, jleal}@letras.up.pt

Abstract

Linguistic studies in under-resourced languages pose additional challenges at various levels, including the automatic collection of examples, cases, and corpora construction. Several sophisticated applications, such as GATE (Cunningham, 2002), can be configured/adjusted/programmed by experts to automatically collect examples from the Web in any language. However, these applications are too complex and intricate to be operated, requiring, in some cases, skills in computer science. In this work, we present TELP, a tool that allows for the simplified expression of linguistic patterns to extract case studies automatically from World Wide Web sites. It is a straightforward application with an intuitive GUI and a quick learning curve, facilitating its broad use by researchers from different domains. In this paper, we describe the operational and technical aspects of TELP and some relatively recent and relevant use cases in the field of linguistic studies.

Keywords: Text Extraction, Web Mining, Linguistic Tools, Empirical Linguistic Studies

1. Introduction

We are currently living in an era of abundance of information which has significantly grown with the rise of the online community and which is expressed in multimodal dimensions such as video, images, sound, and text. The growing volume of online text opens up new avenues of investigation for various sciences, from social to computer sciences.

In Linguistics, text is an essential raw material to carry out and deepen various studies. The major obstacle is the difficulty in gathering many examples characterizing certain phenomena. These examples are accessible on the Web but are difficult to find manually.

One possibility for extracting information is using web crawlers, such as in Di Pietro et al. (2014) or in Sekhar et al. (2019), which systematically search the domains specified by users (URLs). They are practical tools but extract large volumes of information, as they record all the collected pages, therefore providing excessive information unsuitable for specific research purposes. They were designed to index web pages, not just extract specific information segments.

Another possibility is the use of sophisticated tools for Natural Language Processing, like *GATE* (Cunningham, 2002) or *Sketch Engine* (Kilgarriff et al., 2008), which allow multiple linguistic processing operations on corpora but, due to their sophistication, are complex tools with steep learning curves, requiring a significant effort from the user, including those outside computer science fields who are not already familiar with them.

This is where the existence of easy-to-use automatic tools crawling for specific information on the web is very needed, and it was the driving reason behind the application creation presented here. The need for such tools is even more pressing in under-resourced languages that lack adequate corpora and resources.

The *Text Extraction with Linguistic Patterns* (**TELP**) is a desktop application designed to extract textual expressions from the Web, satisfying user-defined language patterns. For example, in Linguistics, the study of Discourse Relations (DR) is of great interest, with diverse applications, including Natural Language Processing (NLP). In particular, we may be interested in studying the phenomenon of sentences involving *adverbial gerundive clauses with compound gerund*. These subordinated clauses have the auxiliary verb "to have" in the -ing form ("having"), followed by the past participle of the main verb (cf. E_1 and E_2).

E_1: *Having served his country, he became a great believer in the need for change and to stop unnecessary wars.*

E_2: *On November 13, the former Brazilian striker had already undergone kidney surgery, having been discharged two days later.*

Therefore, to carefully study this linguistic phenomenon, it is imperative to have a tool that can select hundreds or even thousands of cases from promising web sources. TELP can do this with high precision, depending on the level of rules/patterns the user indicates. In this case, it would be a simple

SIGUL2024 Workshop, pages 337–344

pattern, like "`having+$VBN`"[1].
TELP is a tool from the crawler family but with high precision, oriented towards pattern extraction, and highly configurable. The user can define the URL addresses from which the crawling will be carried out, the crawling depth on each website, and a corresponding timeout. General lexical and syntactic constraints can be activated to satisfy the user's needs better. The extracted textual segments (e.g., sentences) are presented in real time in the application's GUI[2] and stored in HTMl5 files with patterns duly marked in the text through CSS styling. These features are better explained in Section 3.
In Section 2 we briefly present the related work and our findings in our bibliographic research. Section 4 describes actual use cases performed by linguists through TELP. In Section 5 we conclude the paper and point out some possible further improvements.

2. Related Work

There are several available and popular web crawlers, such as *Apache Nutch*, *Storm Crawler*, *Octoparse*, and *Heritrix*, just to mention a few. These are ready-to-use products to satisfy general-purpose crawling tasks, and they all follow the same operation method: the user defines some crawling parameters, including a set of URLs, and commands the start of the process. The crawler will keep on downloading all the web pages from a given URL by recursively following its sub-links, totally or partially. Still, they are all general in scope, i.e., intended for general content extraction from web pages. Crawlers were created as auxiliary tools for Web indexing; therefore, the purpose is to extract full content. However, there are specific needs for extracting elements from the Web in different industrial or academic domains.
A systematic search in several scientific indexing engines failed to yield an application with the same or similar features as TELP. We have searched through three engines, *Google Scholar*, *IEEE Xplore*, and *ACM Digital Library*, using keywords like "*crawler*", "*linguistic crawler*", "*NLP crawler*", etc. The retrieved, analyzed and selected literary material led us to identify some general-purpose corpora creation crawlers (Di Pietro et al., 2014) or some tailored for specific problems/domains/needs like criminal activities (Westlake et al., 2011), sentiment analysis (Mei and Frank, 2015), software engineering (Ferrari et al., 2017), bioinformatics (Sekhar et al., 2019), among others. What changes most significantly among each of these crawlers is the theme of the pages that are obtained. These are selected according to established areas. However, none of the observed crawlers are concerned with selecting parts or segments of the texts contained in the pages. Despite being an application from the crawler family, TELP also addresses this need, only gathering the relevant information for the user according to the defined linguistic patterns.
Another possibility for the automatic extraction of specific corpora from the web would be using more general NLP applications/tools that are popular among the research community and capable of performing various text manipulation operations, including sophisticated linguistic operations. This raises the question: why not adapt these tools for the task at hand? In this regard, we analyze two such tools in what follows.

2.1. Sketch Engine

The first tool we could consider is *Sketch Engine* (Kilgarriff et al., 2008). It started as a corpus tool designed to generate automatic corpus-based summaries called *word sketches*, which detail a word's grammatical and collocational behavior. Initially developed for English, its capabilities have been extended to any language, offering features like thesauruses and sketch differences for linguistic research and lexicography. Key aspects include the development from traditional corpus lexicography to incorporating computational methods for handling large data sets, enhancing lexicographic efficiency, and supporting multi-language processing with advanced grammatical relation identification and analysis. The current version of *Sketch Engine*[3] has evolved significantly, becoming a comprehensive web application and commercial product that serves linguists, lexicographers, translators, students, teachers, and publishers. It analyzes texts from ready-to-use corpora in several languages to provide insights into language use, trends, and emergent linguistic phenomena, like co-occurrence analysis, text alignment, term extraction, etc.
However, "word sketches" were designed to operate on existing corpora and not to perform the extraction of collocations or another linguistic phenomenon directly from web text, bringing only the target segments, as TELP does. Moreover, it is a commercial product and significantly more complex in terms of usability due to its broader scale of functionalities.
Since its inception, the *Sketch Engine* has included a web crawler, *WebBootCaT* (Baroni et al., 2006), but it is a general-purpose crawler, i.e., it collects full text from web pages, allowing the user to define only simple restrictions, such as language. Moreover, WebBootCaT uses third-party tools, such as word searches on the Google search engine, whereas TELP does not.

[1] `VBN` is the *verb past participle tag* used in the *Penn Treebank tagset* (Marcus et al., 1993)
[2] Graphic User Interface.
[3] Source: https://www.sketchengine.eu/

2.2. GATE

The second tool we could consider for extracting textual segments from corpora is *GATE*[4] (Cunningham, 2002). This is a well-known and established application/framework among the natural language processing, computational linguistics, and machine learning communities, having been used for multiple research problems in these areas.

GATE provides a framework and a graphical development environment for developing and deploying software components that process human language. It is designed to work with texts of any language and is flexible enough to handle various tasks, including information extraction, document classification, sentiment analysis, and more. It supports various NLP tasks through a collection of customizable plugins and components. Researchers and developers can use GATE to create complex text processing pipelines that incorporate existing components and plugins or develop their own. Its architecture is based on the principle of modularity, allowing for the easy addition and integration of new components. While advantageous for creating customized solutions, this modular design introduces complexity through its wide range of options and configurations. Users must navigate many components, each with its parameters and functionalities, making the initial stages of learning GATE daunting.

The GATE framework is developed and maintained using Java, which allows it to be platform-independent and capable of running on any operating system that supports a Java Virtual Machine (JVM)[5]. A good understanding of Java (since GATE is Java-based) and familiarity with NLP principles are necessary for more complex tasks, such as developing custom processing resources or plugins. This can make the learning curve steeper for those uncomfortable with programming or the underlying concepts of NLP.

GATE is primarily focused on text and natural language processing tasks and does not inherently include web crawling capabilities as part of its core functionalities. Users typically integrate GATE with other tools or scripts designed for web crawling to fetch the data that can then be processed and analyzed using GATE's extensive NLP features.

Therefore, despite all the potential and importance of this framework, it cannot be used directly for the purposes for which TELP was specifically designed, as already explained, much less by those who do not possess advanced programming skills.

2.3. Conclusion

Thus, to the best of our knowledge, TELP addresses a previously unmet need, being a handy tool for collecting rich, specific textual examples to serve as relevant raw material in various studies. It does not require advanced computational skills, and thus, researchers and practitioners from different communities and backgrounds can effectively use it to fulfill their particular needs.

3. The TELP Application

In this section, we present TELP's graphical interface (GUI) and describe its most relevant operational features. At the end of the section, we will focus more on the language for defining the linguistic patterns that govern text extraction from web pages.

In Figure 1, we show the TELP main view marked with five labels so the reader can easily follow the subsequent reference and description. Each label designates an essential area of the view and will be described below. Areas (1) and (2) are for input, and areas (3) and (5) are for output. Area (4) is also for input but more for parameterization and control.

3.1. GUI Component Description

In area (1), there is a multi-line text box where the user may insert a list of base URLs from which he/she aims to extract text. In this example, two URLs are shown in the box. In area (2), there is another multi-line text box where the user inserts the list of extraction patterns to which the text segments must comply. These text segments are extracted from the URLs indicated in area (1). The example illustrates three extraction patterns, one per line, separated by a comma.

In area (3), the last text segment extracted with the pattern occurrence highlighted (in yellow) is visible. In area (5), a set of relevant information relating to the ongoing extraction process is presented. For instance, a blue progress bar is related to the time-out defined in the area (4), and the *Time spent* is also shown in (5). The field *Extracted* reveals the number of extracted segments so far. The set of all extractions is displayed in another view, accessible by the *Extractions* button available at the top of the view, next to *Main Control*.

Finally, area (4) defines a set of parameters to control the extraction process. The "*Stop*" button is a *Start/Stop* button that dynamically changes its label depending on the current state: *pre-extraction* or *in-extraction*. The "*Clear*" button resets the extracted elements if we wish to restart the extraction without the previously extracted data to avoid new cases accumulating with those from previous extractions. Additionally, in area (4), a *combobox* permits the user to choose the language. So far, the two available languages are English (selected) and Portuguese. Furthermore, four *checkboxes* can be utilized for activating/deactivating the correspond-

[4]GATE: *General Architecture for Text Engineering*

[5]The same holds on TELP.

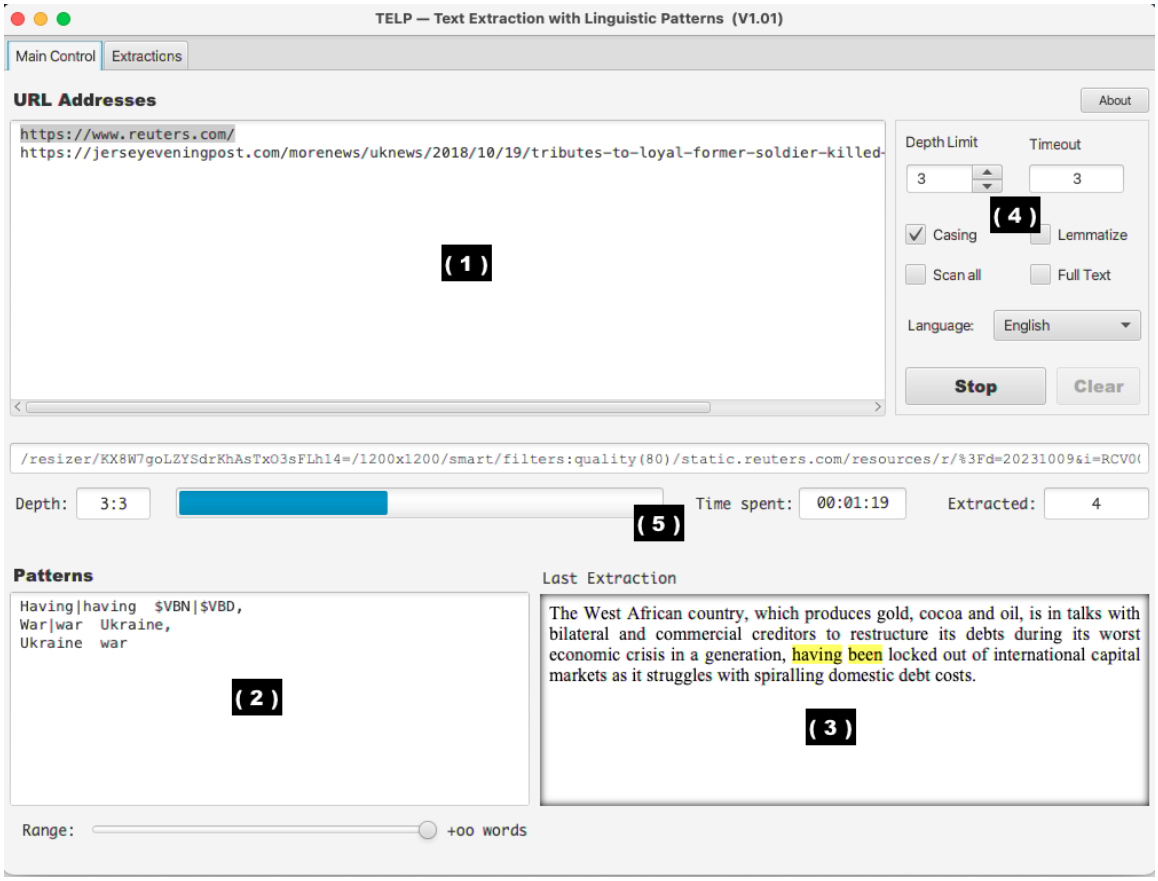

Figure 1: Main view of the TELP application.

ing parameters. Thus, *Casing* controls case sensitivity; *Lemmatize* allows one to lemmatize the text and work with the lemmas of words instead of their derived forms; when *Scan all* is active it will process all URLs indicated in the area (1), instead of just processing the selected one; and *Full text* serves to explore areas of HTML considered less conventional to store text. Finally, in area (4), the *Depth Limit* defines the level of extractive depth in the site's hyperlink hierarchy and *Timeout* fields stipulate the maximum extractive time allowed for each site/URL. More about this will be explained in Section 3.2.

3.2. Text Extraction Operation

The extraction process begins after the user presses the *Start* button, assuming that he/she has already entered the URLs/links in (1), from which he/she intends to obtain the text and the linguistic patterns, in (2), for the extraction. This is the minimum the user must do before the extraction begins. As mentioned before, the user can also adjust some parameters for the extraction in area (4). For example, extraction will be sequentially performed for all links inserted in area (1) if the *Scan all* checkbox is selected. In this case, each of the links is searched sequentially, from the first one that is selected to the last one. Otherwise, the search will be conducted only on the link selected in (1).

For a given URL u, the extraction follows a conventional crawling algorithm that visits each sub-link of u, let us say $u_1, u_2, ...u_n$, where u is a prefix for any u_i which is a hyperlink/link contained in u. For example, if:

$$\begin{aligned} u &= \texttt{www.reuters.com} \\ u_i &= \texttt{www.reuters.com/world/} \\ u_j &= \texttt{www.reuters.com/world/europe/} \end{aligned}$$

both u_i and u_j are sub-links of u, let us represent it as $u \triangleright u_i$ and $u \triangleright u_j$. Furthermore, there are two sub-link levels here, i.e., $u \triangleright u_i \triangleright u_j$. For a given u only sub-links of u are visited in a systematic recursive method up to a pre-defined depth. This depth is exactly what "*Depth Limit*" means in area (4). In our previous example, we have:

$$depth(\ u \triangleright u_i \triangleright u_j\ ) = 3.$$

The textual content is carefully extracted for each web page read. By default, the usual small text segments related to the site's structure or advertisement are avoided. Here, some heuristics are used to extract text composed of well-formed sentences that effectively relate to the main subject

of that page. If the checkbox "*Full text*" from area (4) is selected, this filtering care will not hold, and all textual content will be extracted. Afterward, NLP operations are performed on the extracted text, starting with sentence tokenization, the part-of-speech (POS) tagging of each sentence, and the possible[6] word case lowering and word lemmatization. The current version of TELP uses the *Apache OpenNLP* (Foundation, 2023) for POS tagging and *Morphadorner* (Burns, 2013) for lemmatization.
After the NLP operation is performed, the sentences are ready to be submitted to the list of extraction patterns defined by the user in area (2). For a given sentence, the first applicable pattern generates an extraction case, causing the sentence to be actually stored and the occurrence of the respective pattern marked with CSS styling. In Section 3.4, the language for defining extraction pattern is described.

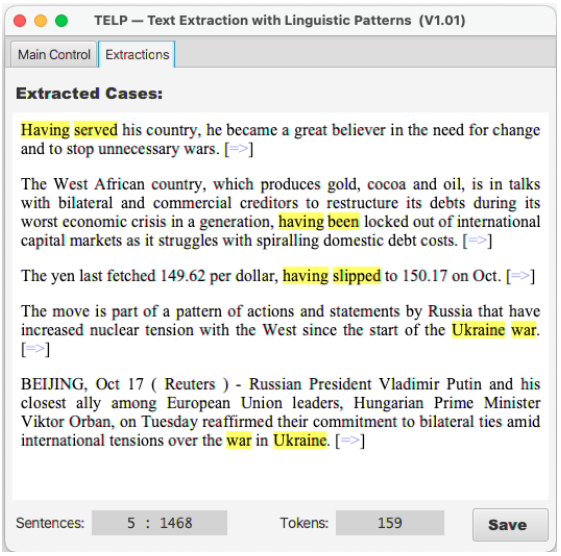

Figure 2: Extractions with patterns marked.

The cases that are being extracted are dynamically presented in the "*Extractions*" view, accessible from the top of the main view ("*Main Control*"), as shown in Figure 2.

3.3. Text Crawling Process

The extraction of well-structured text from web pages poses several key obstacles, like the issue of navigating through "spurious textual segments," such as those found in advertisements, web page structural components (e.g., menus, sidebars), which bear no relation to the central document theme and very likely hold no value for the user. A common feature of these extraneous segments is their deficiency in syntactical integrity, often evident just by looking at the inadequate or nonexistent punctuation in these segments. Consequently, our approach acknowledges these characteristics, ensuring the extraction of well-formed text segments. There are recent and sophisticated methods, like in *Trafilatura* (Barbaresi, 2021), yielding almost perfect text scraping from web pages. In our case, we observed that we achieved a very satisfactory result by following lexical heuristics that closely matched the one mentioned. If the *Full text* parameter (area 4 from Figure 1) is not set, almost all extracted text is well-formed.
The method followed to gather valid sentence examples (named here as $sentexes$) from the *World Wide Web* can be synthesized in Algorithm 1.

Algorithm 1 – Crawling "Sentexes" from the Web

```
1:  Input: websites W = {w_1, ..., w_n}, patterns.
2:  Output: collected text sentexes.
3:  sentexes ← {}
4:  memory ← {}
5:  for w_i ∈ W do
6:      stxs ← crawlPage(w_i, memory, patterns)
7:      sentexes ← sentexes ∪ stxs
8:  end for
9:
10: Store(sentexes)
11:
12: function CRAWLPAGE(url, memo, patterns)
13:     text ← selectText(url)
14:     stxs ← selectSentexes(text, ptrs)
15:     for u ∈ subLinks(url) do
16:         if u ∉ memo then
17:             memo ← memo ∪ {u}
18:             s ← crawlPage(u, memo, patterns)
19:             stxs ← stxs ∪ s
20:         end if
21:     end for
22:     return stxs
23: end function
```

The crawling function (line 12), called at the beginning (line 6) receives the base url from which the crawling starts, the set of links/URLs already visited ($memo$), and the set of $patterns$ to apply. This function is recursive (line 18) and will "dive" until the predefined depth (area (4) from Figure 1).
We can observe the verification of well-formed text segments in line 13, 'selectText(urls),' during web page extraction, as well as the fulfillment of linguistic patterns predefined by the user, in line 14, "selectSentexes($text$, $patterns$)".
An important point here that needs clarification is what happens, for example, when two or more patterns are applicable to the same sentence from a text ("selectSentexes" function, line 14), usually at different positions in the sentence. In such cases, each pattern applicable to the sentences produces a different case, i.e., an independent $sentex$ corre-

[6] Depending on the settings in area (4).

sponding to each applicable pattern ($patterns$).

3.4. Extraction Language

This application was essentially designed to be operated mainly by people outside the field of Computer Science and certainly unfamiliar with the very notion of *regular expression*[7]. It is the case of linguists who need to extract sentences in which certain grammatical conditions are satisfied. Thus, the pattern definition language also constitutes an interface, a mediator between the user's needs and the complexity of defining regular expressions involving constraints on strings of different categories (lexical, syntactic, semantic, etc.). Therefore, a relatively simple yet expressive language was designed and incorporated into TELP, enabling users to define sentence extraction patterns from online text.

In this pattern language, the simplest level is the lexical one, where sequences of words that must appear in a sentence are indicated for it to be extracted. For example, in the "*Patterns*" box in area (2), three patterns are visible, separated by commas. The last pattern is the simplest one, requiring the word "`Ukraine`" to be present in the sentence and the word "`war`" in a later position[8]. This pattern was satisfied in the fourth example presented in Figure 2.

The language uses two operators, the disjunction "`|`" and the conjunction "`&`", which by default may be omitted. For instance, "`Ukraine war`" means exactly the same as "`Ukraine & war`". The disjunctive operator allows a combination of lexical variations within a single pattern. Thus, the pattern "`War|war Ukraine`" represents two combinations and a pattern like:

```
war|conflict  Ukraine|Russia
```

represents four simple lexical combinations: "`war & Ukraine`", "`war & Russia`", "`conflict & Ukraine`", and "`conflict & Russia`". Note that the first combination would match the last sentence from Figure 2. The user can combine/conjugate as many disjunctive conditions as needed and quickly define a complex and powerful lexical pattern. We have also defined a negation operator, the tilde "`~`", with which the user can force a word not to occur in the sentence. For example, "`war & ~Ukraine`" would match any phrase that contains "`war`" but not "`Ukraine`".

Additionally, the user may incorporate syntactic conditions through POS tags. We can thus force, for example, that, after a word (lexical constraint), there must be the past participle of any verb, or an adjective, or both, etc. One may use any tag from the *Penn Treebank tagset* (Marcus et al., 1993).

[7] How computer scientists define information patterns.

[8] It does not have to be immediately followed.

The first pattern in area (2) illustrates one such example, where area (3) displays the corresponding extracted sentence with the pattern satisfaction highlighted by TELP. In any extracted case these patterns will be marked and thus visible in the interface (the *Extractions* view, Figure 2).

TELP has specific controls for recording these extracted sentences/segments. The simplest way is through the "Save" button in the lower left corner. The data is saved in an HTML file whose name consists of the extraction time stamp. For each sentence, the patterns satisfied in the sentences are delimited by specific tags, allowing both the visualization (HTML+CSS) and subsequent processing by other applications. For example, the third sentence visualized in the view of Figure 2 could be saved as follows:

```
The yen last fetched 149.62 per dollar,
<ptr id="1">having slipped</ptr> to
150.17 on Oct.
```

Note the delimitation of the pattern occurrence (having slipped) through the `<ptr>...</ptr>` tags (abbreviation for pattern). Furthermore, the argument `id="1"` means that it is related to the first pattern in the list of patterns defined by the user in (2), in Figure 1. Therefore, it is not just a visual marking but also a semantic one, allowing the recorded file to be subsequently processed automatically by other tools.

4. Use Cases

In this section, we describe three actual scenarios in which the TELP application was used to extract relevant sentences for linguistic studies. The first case we want to mention involves the extraction of sentences combining verbs of movement and prepositions in European Portuguese. Examples were extracted from online newspapers and a corpus was built using a sample from this extraction, with sentences combining the verbs "ir" (to go) or "vir" (to come) with either preposition "para" (to, towards) or "até" (up to). According to the data, these movement verbs can occur with both prepositions (with minor changes in meaning) when the predications they project are understood as non-fictive motion events. On the contrary, when the predications exhibit a fictive motion reading, (i) prepositions are not interchangeable, and (ii) only "ir" combines with both prepositions, whereas "vir" combines only with "para", rejecting the cooccurrence with "até". In the theoretical proposal put forward in (Leal et al., 2018) the data collected using TELP was paramount to detect these regularities and to validate the actual use of these expressions by native speakers.

The second case involves adverbial perfect participial clauses, that is, clauses with an auxiliary verb 'have' in the *-ing* form followed by the main

verb in the past participle. In this case, five language varieties were considered: British English and European, Brazilian, Angolan and Mozambican Portuguese. Again, TELP was used to search and extract complete sentences with this construction from different online journals of these five countries. These sentences were annotated with several linguistic features, such as tense, temporal interpretation or aspectual classes of predications. The analysis of this corpus, presented in (Silvano et al., 2021), revealed, for instance, that the temporal readings of adverbial perfect participial clauses depend on different linguistic elements in English and Portuguese (irrespective of the national varieties). Later, this corpus was also annotated with discourse relations using ISO 24617-8 (ISO, 2016), and it was released in 2023 to the community, together with an application with a graphical user interface (Silvano et al., 2023). The collection of data for this study would have been much more complex and time-consuming if it were not for TELP.

The third use case demonstrates how TELP can be helpful in collecting data with specific patterns in under-resourced languages. Such patterns may be difficult or even impossible to access otherwise. In this case, TELP was used to extract linguistic patterns involving dative constructions. These constructions typically express a change of possession or location, as in the example sentence *dar o dinheiro ao povo*, which means "give money to the people". TELP played a crucial role in acquiring actual data from online Angolan newspapers, including both news articles and comment boxes.

It is essential to highlight the importance of TELP in obtaining data automatically for languages and variants (e.g. African Portuguese variants) that still have very few resources, both in terms of corpora and case studies and in terms of automatic tools for their processing. The research cases described demonstrate the tool's strategic importance in ensuring the necessary material for conducting the linguistic studies intended in these under-resourced languages and varieties.

5. Conclusions

In conclusion, TELP (Text Extraction with Linguistic Patterns) has emerged as an effective tool in Computational Linguistics, meeting the critical need for extracting specific textual segments from the web through user-defined linguistic patterns. Unlike existing tools and frameworks such as Sketch Engine and GATE, which are either too complex for non-specialists or lack direct web text extraction capabilities, TELP offers a user-friendly interface and allows for precise and efficient linguistic data collection. It stands out for its simplicity, flexibility, and ability to accommodate the specific needs of researchers, particularly in under-resourced language studies. The demonstrated use cases underscore TELP's utility in facilitating empirical linguistic research across different languages and linguistic phenomena.

In terms of the application's usability, formal evaluation has yet to be conducted according to the principles of Human-Computer Interaction (Dix, 2003), because up to the current version of TELP, there has been no need for it, given that its operational complexity is extremely low. The application has been used by various users, from students to senior researchers. We have observed that users need no more than 15 minutes to become thoroughly familiar with the application. This is impossible with any other application or tool reported in Section 2. Currently, this version of TELP does not consider the restrictions specified in "`robots.txt`" files, which guide automated web access to respect website owners' wishes. However, future releases intended for community use will incorporate adherence to these protocols. This inclusion aims to ensure ethical web scraping practices, respecting site owners' preferences and legal requirements, thereby addressing potential concerns about unauthorized data access and content extraction.

Regarding future improvements, we are committed to facilitating the easy integration of linguistic resources from different languages into TELP. This initiative aims to enhance the tool's versatility and utility across diverse linguistic landscapes. We are also exploring the potential to incorporate semantic conditions into our extraction patterns, possibly resorting to new language models (Devlin et al., 2018; Min et al., 2023), thereby enriching the context and relevance of the data collected. By integrating these models, TELP aims to extract text based on surface patterns and understand the underlying meaning, enabling more nuanced and targeted data extraction.

Acknowledgments

This work has been partially funded by NOVA LINCS (UIDB/04516/2020) with the financial support of FCT.IP and the European Commission through the Horizon 2020 project Pharaon, grant agreement no. 857188.

6. References

Adrien Barbaresi. 2021. Trafilatura: A web scraping library and command-line tool for text discovery and extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 122–131, Online. Association for Computational Linguistics.

Marco Baroni, Adam Kilgarriff, Jan Pomikálek, Pavel Rychlỳ, et al. 2006. Webbootcat: a web tool for instant corpora. In *Proceeding of the EuraLex Conference*, volume 1, pages 123–132.

Philip R. Burns. 2013. Morphadorner v2: A java library for the morphological adornment of english language texts. *Northwestern University, Evanston, IL*.

Hamish Cunningham. 2002. Gate, a general architecture for text engineering. *Computers and the Humanities*, 36:223–254.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Giulia Di Pietro, Carlo Aliprandi, Antonio E De Luca, Matteo Raffaelli, and Tiziana Soru. 2014. Semantic crawling: an approach based on named entity recognition. In *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014)*, pages 695–699. IEEE.

Alan Dix. 2003. *Human-Computer Interaction*. Pearson Education.

Alessio Ferrari, Beatrice Donati, and Stefania Gnesi. 2017. Detecting domain-specific ambiguities: an nlp approach based on wikipedia crawling and word embeddings. In *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, pages 393–399. IEEE.

Apache Software Foundation. 2023. Apache opennlp developer documentation. https://opennlp.apache.org. Access: 2023-10-18.

ISO. 2016. ISO 24617-2: 2016. Language resource management, Part 8: Semantic relations in discourse (DR-core). Standard, Geneva, CH.

Adam Kilgarriff, Pavel Rychly, Pavel Smrz, and David Tugwell. 2008. The sketch engine. *Practical Lexicography: a reader*, pages 297–306.

António Leal, Fátima Oliveira, and Purificação Silvano. 2018. Path scales. *Tense, Aspect, Modality, and Evidentiality: Crosslinguistic perspectives*, 197:335.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank.

Joseph Mei and Richard Frank. 2015. Sentiment crawling: Extremist content collection through a sentiment analysis guided web-crawler. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, pages 1024–1027.

Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. 2023. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Surveys*, 56(2):1–40.

S.R. Mani Sekhar, G.M. Siddesh, Sunilkumar S. Manvi, and K.G. Srinivasa. 2019. Optimized focused web crawler with natural language processing based relevance measure in bioinformatics web sources. *Cybernetics and Information Technologies*, 19(2):146–158.

Maria da Purificação Silvano, João Cordeiro, António Leal, and Sebastião Pais. 2023. Dripps: a corpus with discourse relations in perfect participial sentences. In *Language, Data and Knowledge 2023 (LDK 2023): Proceedings of the 4th Conference on Language, Data and Knowledge*.

Purificação Silvano, António Leal, and João Cordeiro. 2021. On adverbial perfect participial clauses in portuguese varieties and british english. *Romance Languages and Linguistic Theory 2018: Selected papers from'Going Romance'32, Utrecht*, 357:263.

Bryce G. Westlake, Martin Bouchard, and Richard Frank. 2011. Finding the key players in online child exploitation networks. *Policy & Internet*, 3(2):1–32.