Syntax Deep Explorer

José Correia[1,2], Jorge Baptista[2,3], and Nuno Mamede[1,2]

[1] Instituto Superior Técnico, Universidade de Lisboa
[2] L^{2}F – Spoken Language Lab, INESC-ID Lisboa
Rua Alves Redol 9, P-1000-029 Lisboa, Portugal
jcorreia,Nuno.Mamede@inesc-id.pt
[3] Universidade do Algarve
Campus de Gambelas, P-8005-139 Faro, Portugal
jbaptis@ualg.pt

Abstract. The analysis of the co-occurrence patterns between words allows for a better understanding of the use (and meaning) of words and its most straightforward applications are lexicography and linguist description in general. Some tools already produce co-occurrence information about words taken from Portuguese *corpora*, but few can use *lemmata* or syntactic dependency information. *Syntax Deep Explorer* is a new tool that uses several association measures to quantify several co-occurrence types, defined on the syntactic dependencies (*e.g.* subject, complement, modifier) between a target word *lemma* and its co-locates. The resulting co-occurrence statistics is represented in *lex-grams*, that is, a synopsis of the syntactically-based co-occurrence patterns of a word distribution within a given *corpus*. These *lex-grams* are obtained from a large-sized Portuguese *corpus* processed by STRING [19] and are presented in a user-friendly way through a graphical interface. The *Syntax Deep Explorer* will allow the development of finer lexical resources and the improvement of STRING processing in general, as well as providing public access to co-occurrence information derived from parsed *corpora*.

Keywords: Natural Language Processing (NLP), co-occurrence, collocation, association measures, graphic interface, lex-gram, Portuguese

1 Introduction

The analysis of the co-occurrence patterns between words in texts shows the differences in use (and meaning), which are often associated with the different grammatical relations in which a word can participate [7,33]. The quantification of these patterns is a powerful tool in modern lexicography as well as in the construction of basic linguistic resources, like *thesauri*. The stakeholders on the study of those co-occurrence patterns are linguists, language students, translators, lexicographers and Corpus Linguistics' researchers, who study the behaviour of linguistic expressions in *corpora*. For all of these, co-occurrence data are essential for a better understanding of language use. Furthermore, comparing different association measures, each capturing different linguistic aspects of the co-location phenomena, enables the user to achieve a broader understanding of the distribution of a given word, hence its different meanings and uses. For a better analysis of co-occurrence patterns in a *corpus*, it is also important to provide the user with an interface that helps him/her to explore the patterns thus extracted, namely, by accessing concordances or moving on from an initial query to another interesting collocate.

To date, only one system, *DeepDict* [3], is known to produce co-occurrence information based on syntactic dependencies between (simple) word *lemmata*, but it only features a single association measure, Pointwise Mutual Information (PMI)[7]. Other systems available for Portuguese *corpora*, like *Sketch Engine* (see below), do not benefit from syntactic information nor lemmatization and also use a single association measure, LogDice [18].

This paper presents *Deep Syntax Exporer*[4]. Its main goal is to provide the general public a tool to explore syntactically-based collocations from Portuguese *corpora*, using a broad set of association measures, in view of an enhanced understanding of words' meaning and use. The information on these collocation patterns is made available through a web interface by way of *lex-grams*, a synopsis of the syntactically-based co-occurrence patterns of a word's distribution within a given *corpus*. This mode of presentation organizes the information for a simpler analysis by users. For now, only the four main lexical part-of-speech categories (adjectives, nouns, verbs and adverbs) are targeted by the *Deep Syntax Exporer*.

This paper is organised as follows: Next, in Section 2, key concepts and related work are presented; first, the Portuguese processing STRING [19] underlying the tool is briefly sketched; then, the main existing systems, that provide co-occurrence information for Portuguese *corpora*, are succinctly described; finally, the association measures implemented in *Deep Syntax Explorer* are briefly sketched. Section 3 presents the system architecture in detail, beginning with the database, then the co-occurrence information extraction module, and, finally, the web application and the way the *lex-grams* display this information to the end user. Section 4 presents the evaluation of the tool's performance. Section 5 concludes the paper and presents future directions of development.

2 Related Work

This section starts by briefly presenting the STRING system underlying the *Syntax Deep Explorer*, from whose output the co-occurrence information is extracted. Then, it describes three co-occurrence information extraction systems already existing for processing Portuguese *corpora*. Finally, it provides a quick overview of association measures implemented in *Syntax Deep Explorer*.

The STRING NLP system

STRING [19] is a hybrid, statistical and rule-based, natural language processing (NLP) chain for the Portuguese language, developed by Spoken Language Sys-

[4] `string.l2f.inesc-id.pt/demo/deepExplorer` (last visit 29/02/2016).

tems Lab (L^{2}F) from INESC-ID Lisboa[5]. STRING has a modular structure and performs all basic text NLP tasks. The first module is the *LexMan* [35], which is responsible for text segmentation (sentence splitting and tokenization) for assigning to each token its part-of-speech (POS) and any other relevant morphosyntactic features. The module *RuDriCo* [10] is a rule-driven converter, used to revolve ambiguities and to deal with contractions and certain compounds words. The *MARv* [28,11] is a statistical part-of-speech disambiguator that chooses the most likely POS tag for each word, using the Viterbi algorithm. Finally, the *XIP* (Xerox Incremental Parser)[1] uses a Portuguese rule-base grammar [20] to structure the text into chunks, and to establish syntactic dependencies between their heads. This parser also performs other NLP tasks such as named entity recognition (NER) [15,25], anaphora resolution (AR) [22,26] and temporal expressions normalization [13,14,16,23]. The processing produces a XML output file.

The *Syntax Deep Explorer* uses the following (higher level) syntactic dependencies, produced by the XIP parser: (1) the `SUBJ` dependency, which links a verb and its subject; (2) the `CDIR` dependency, linking a verb and its direct complement; (3) the `CINDIR` dependency, which links the verb with a dative essencial complement; (4) the `COMPL` dependency, which links a predicate (verb, noun or adjective) to its essential `PP` complements; and (5) the `MOD` dependency that links a modifier with the element it modifies (*e.g.* adjetives as modifiers of nouns, or adverbs as modifiers of verbs); this is an umbrella dependency, as it also connects any `PP` complement to its governor, iff it has not been already captured by `COMPL` or another dependency); and (6) Named Entities, which also captured by the XIP parser by an unary `NE` dependency that delimits and assigns them to several general categories (`PERSON`, `ORGANIZATION`, `PLACE`, etc.); these categories are then used for co-occurrence statistics, instead of the individual entities themselves.

Current systems providing co-occurrence information

Nowadays, some tools already make it possible to obtain information on the co-occurrence patterns of a given word from Portuguese *corpora*, which will now be described briefly.

The *AC/DC* platform available through *Linguateca*[6] produces raw, quantitative data on words' co-occurrences from a large variety of Portuguese *corpora*, allowing the user to query complex combinatorial patterns by way of regular expressions, though it does not make use of any association measure.

The *DeepDict* [3] is a tool developed by GrammarSoft[7] and presented in the *Gramtrans* platform[8]. For Portuguese *corpora*, this tool uses the NLP analyzer PALAVRAS [2] and it collects `dep-grams` from the syntactical dependencies produced in the parsed texts. A `dep-gram` is a pair of *lemmas* of two words that

[5] `www.l2f.inesc-id.pt` (last visit 29/02/2016).
[6] `http://www.linguateca.pt/ACDC/` (last visit on 29/02/2016).
[7] `http://grammarsoft.com/` (last visit 29/02/2016).
[8] `http://gramtrans.com/gramtrans` (last visit on 29/02/2016).

feature a dependency relation between them. The *dep-grams* are quantified by PMI measure. In the database, the tool stores the `dep-gram`, the value of PMI, the absolute frequency and the ID of the sentence where these `dep-gram` occurred in the *corpus*. DeepDict has a simple form as an interface. The result is presented as a *lexicogram* of the searched *lemma*, which shows the different dependency relations that the word establishes with other lexical elements, sorted in PMI value descending order. The lexicogram displays the co-locates in the natural reading order of the related words. Words having different POS have different lexicograms. At this stage, DeepDict does not provide access to concordances of a given *dep-gram*.

The *Sketch Engine* [17] is a tool developed by Lexical Computing[9] This system uses Manatee [29] to manage *corpora*. The tool stores for each word its *lemma* and POS tag. Co-occurrences are identified by the tool and are quantified by the LogDice measure. The system has a graphical interface that is generated by the tool Bonito [29] and allows access to stored data. The system main features are: (i) *concordance*, that is, access to examples taken from the *corpora* but without any analysis; (ii) *word sketches*, which show the words more closely related to the target word; (iii) *sketch-diff*, which depicts the differences between two different word sketches; and (iv) *thesaurus*, which shows similar words for a given target word. The system allows the users to create and manage their own *corpora*.

The *Wortschatz* [27] system has been developed at Leipzig University, and it creates lexical similarity networks from *corpora*. The system identifies two types of co-occurrences: (i) words occurring together in a sentence and (ii) words occurring next to each other. The system uses tree-based algorithms that allows a quick co-occurrence analysis of the entire *corpus* [4]. However, it does not use the words *lemmata* for these calculations. Each co-occurrence is quantified by a Significance Measure (see below) and this data is stored in a MySQL database. The system interface provides, for a target word, its frequency class; examples of its use (sentences); significant co-occurrences and a co-occurrence chart.

In sum, all these systems are able to list the co-occurrences of a word, though Wortschatz does not use *lemmas*. The Sketch Engine is the system that offers more features, particularly the *corpus* management tools and the sketch-diff. On the other hand, DeepDict shows co-occurrence statistics based on syntactic dependencies between *lemmata* and does so in a more readable, user-friendly way.

Association Measures

To understand how two words relate to each other, it is essential to quantify the co-occurrence patterns found between them. Six different association measures, commonly used in *corpus*-based linguistic analysis, were adopted and implemented in *Syntax Deep Explorer*. The first association measure is *Pointwise Mutual Information* (PMI) [7], which links the number of co-occurrences between two words with the number of occurrences of each word. The *Dice Coefficient* [9,34] is another association measure that calculates the degree of cohesion be-

[9] `http://www.sketchengine.co.uk/` (last visit 29/02/2016).

tween two words. The *LogDice Coefficient* [30] is a variant of the Dice Coefficient and it is said to fix the problem of very small calculated values. The *Pearson's Chi-Squared* measure [21] is a statistical method of hypothesis testing. This association measure compares the observed frequencies with the expected frequencies for a given pattern in order to verify if there is independency between events (null hypothesis). If the difference between observed and expected frequencies is larger than a statistical significance threshold, then the null hypotheses is rejected and the frequency of the pattern is deemed to be statistically relevant, given the *corpus*. This measure, however, should not be used in small *corpora* nor with patterns with low frequency. The *Log-likelihood Ratio* [12] is another approach to hypothesis testing, and it is considered to be more suitable for sparse data than the previous method. Finally, the so-called *Significance Measure*, used by the *Wortschatz* system [4], is comparable to the statistical *G-Test* method for Poisson distribution. This measure associates the co-occurrences of two words with the number of sentences in which they occurred.

3 *Deep Syntax Explorer* Architecture

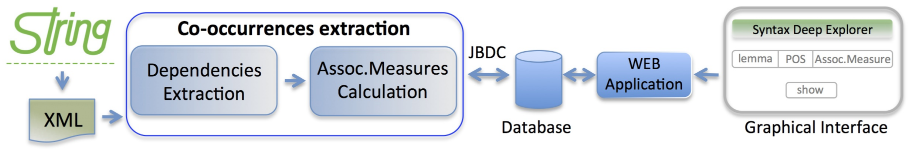


Fig. 1. *Syntax Deep Explorer* architecture.

This section presents the *Syntax Deep Explorer* architecture, sketched in Figure 1. The tool consists of three modules that will be briefly presented below: (i) the database, (ii) the co-occurrence information extraction module, and (iii) the web interface application.

Database

To store the co-occurrence statistics extracted from *corpora*, a consistent data model with low information redundancy is required. An Entity-Relationship (ER) Model [6] was developed, allowing for a more natural perception of the problem. Since typical database systems use a relational model [8], it is necessary to convert the ER model. The relational model is defined by a set of relations, represented in tables, where the columns are attributes of the relationship and each row is a tuple (entry), which is unique. To identify each tuple, primary keys are used to distinguish them. If a relation refers to another one, it has reference attributes called *foreign key*, where the values of these attributes have the *primary key* value of a tuple in the referenced relation [32].

The conversion to the relational model must follow the defined conversion rules and the previous data integrity constraints. SQLite[10] was used to implement the obtained relational model. SQLite works locally, and it allows a di-

[10] `http://www.sqlite.org/about` (last visit 29/02/2016).

rect and faster access to data, without a remote server. Access to the data is made through SQL language, making it possible to manipulate the tables in the database and to obtain the desired results.

Co-occurrence information extraction module
Figure 2 introduces the UML (Unified Modelling Language) packages diagram representing the tool that is responsible for the extraction of dependency co-occurrence information, the *DeepExtractor*. The diagram shows the main components used.

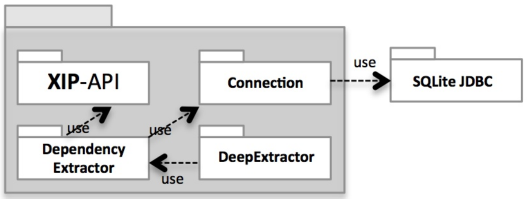


Fig. 2. Packages diagram for dependency co-occurrence extraction.

The XIP-API [5,24] enables the transformation of the XML files content from the STRING parsing module, the XIP parser, into Java structures and it organizes the information imported. The result is the `XIP-Document` that consists in a set of `XIP-Node`s and `XIP-Dependency`(ies). A `XIP-Node` represents a node from the XIP parser's output, the basic element of *chunking tree*. It contains other `XIP-Node`s (child nodes) and their `Features`, with the properties of each node. A `XIP-Dependency` contains the information about a dependency detected by the XIP parser and the `XIP-Node`s to which it applies. To store the data in the database, the JBDC (Java Database Connectivity) API is used. The SQLiteJDBC[11], the JDBC for SQLite, is used.

The package `Connection` facilitates the access to the *Syntax Deep Explorer* database. In this package, a connection is established to the database through the SQLiteJDBC. Here, a group of classes has been developed, where the methods are used to manipulate the database information. Each class represents a database table and has methods to verify if an entry already exists and to insert new entries to that table. Other classes have additional methods to compute the values of the association measures.

The package `DependencyExtractor` is the core of the extraction tool. With the help of the XIP-API, the package analyses the texts produced by STRING to collect the statistics on co-occurrence syntactic dependencies. The `DependencyExtractor` method `Extract` accepts a `XIP-Document`, and for each sentence (indicated by a `XIP-Node` named `TOP`) in this document, it obtains within this sentence: (i) the set of named entities; (ii) set of `XIP-Dependency`(ies), each one linking two `XIP-nodes`; and (iii) the sentence string.

Afterwards, for each `XIP-Dependency`, the analysis process depends on the type of dependency it belongs to. A set of classes was developed, each one rep-

[11] `https://bitbucket.org/xerial/sqlite-jdbc` (last visit 29/02/2016).

resenting a `XIP-dependency` (*v.g.* `SUBJ`, `CDIR`, `CINDIR`, `COMPL`, and `MOD`); and it runs the correspondent code. In these classes, the method `getDepInformation` accepts a `XIP-Dependency` and a set of named entities in the sentence. In this method, to prevent the analysis of dependencies and properties that are not relevant to the goals of *Syntax Deep Explorer*, the dependency-property pattern of `XIP-Dependency` is checked against a list of predefined patterns. If that pattern is present, the analysis proceeds normally. The list of dependency-property patterns present in the system indicates for each word POS which are the relevant relations stored in the database. Then, for each `XIP-Node` the word *lemma* and POS are obtained. In the case a `XIP-Node` is present in the set of named entities, the word *lemma* is replaced by the correspondent named entity category (*e.g. João Silva* by `PERSON`, *NATO* for `ORGANIZATION`, *Lisboa* for `PLACE`).

During the process, it is necessary to store the information that is being found. Once a processed *corpus* is divided into several files, for each file, the information extracted is gathered in Java structures. The `DeepStorage` class works as a local database and helps to organize the information until this is stored in the database. After the extraction from the *corpus* file is completed, all gathered information is then stored in the database. When the *corpus* extraction is completed, it is necessary to calculate the values of the association measures for all syntactic dependency co-occurrences existing between two words. The `DeepMeasures` calculates the six association measures described above.

Web Application

Figure 3 shows the architecture of the web application.

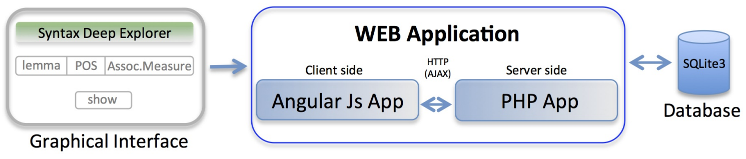


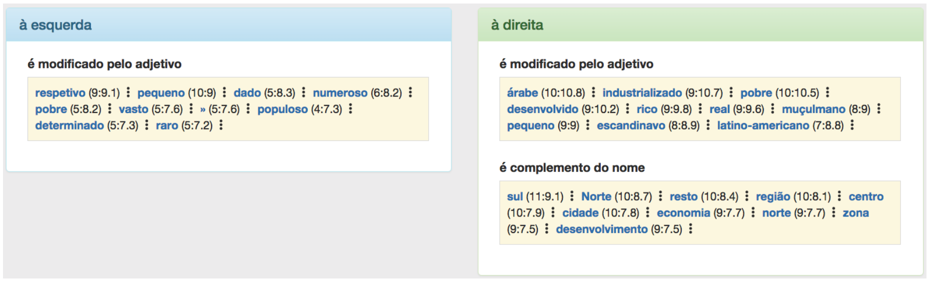
Fig. 3. Web application architecture.

This application has two main components, the server-side and the client-side (browser). The first runs the code that implements the processing of information in the database and the second runs the code that implements the graphical interface. The communication between them is performed through asynchronous AJAX (Asynchronous Javascript and XML) requests (via HTTP) and the transmitted data is a JSON (JavaScript Object Notation) object. On the server-side, the code was developed in PHP. A request from the client consists in asking the co-occurrences of a word *lemma* and its POS, or a request for the sentences that exemplify a given co-occurrence. From the client request, several SQL queries are made in the database to get the information about that word or that co-occurrence. This information is organised in a JSON object and sent to the client. On the client-side, the code was developed from the AngularJS[12] frame-

[12] `https://angularjs.org` (last visit 29/02/2016).

work, which allows the use of HTML and CSS, as declarative and presentation languages, respectively.

To begin the process, the user is presented a form where s/he enters the target word *lemma* and chooses its POS; the user also indicates the association measure to be used. Some additional options can also be selected, such as the minimum frequency of each co-occurrence and the maximum number of words to be shown in each dependency-property pattern. By default, these values are set to '2' and '10', respectively. The collected information is organised in a JSON object and sent to the server. The user is informed when the target word does not exist in the *corpus* or no results were found for it. Otherwise, the system produces what we designate as a *lex-gram*, that is, a synopsis of a word distribution in a given *corpus*. This information is presented in two groups: words appearing to the left and the right of the target word. Figure 4 shows the *lex-gram* for the noun ***país*** (country), using the *LogDice* association measure on the CETEMPúblico *corpus* [31].


à esquerda

é modificado pelo adjetivo

respetivo (9:9.1) ⋮ **pequeno** (10:9) ⋮ **dado** (5:8.3) ⋮ **numeroso** (6:8.2) ⋮ **pobre** (5:8.2) ⋮ **vasto** (5:7.6) ⋮ **»** (5:7.6) ⋮ **populoso** (4:7.3) ⋮ **determinado** (5:7.3) ⋮ **raro** (5:7.2) ⋮

à direita

é modificado pelo adjetivo

árabe (10:10.8) ⋮ **industrializado** (9:10.7) ⋮ **pobre** (10:10.5) ⋮ **desenvolvido** (9:10.2) ⋮ **rico** (9:9.8) ⋮ **real** (9:9.6) ⋮ **muçulmano** (8:9) ⋮ **pequeno** (9:9) ⋮ **escandinavo** (8:8.9) ⋮ **latino-americano** (7:8.8) ⋮

é complemento do nome

sul (11:9.1) ⋮ **Norte** (10:8.7) ⋮ **resto** (10:8.4) ⋮ **região** (10:8.1) ⋮ **centro** (10:7.9) ⋮ **cidade** (10:7.8) ⋮ **economia** (9:7.7) ⋮ **norte** (9:7.7) ⋮ **zona** (9:7.5) ⋮ **desenvolvimento** (9:7.5) ⋮

Fig. 4. *Lex-gram* of the noun `país` ordered by the LogDice measure.

For each dependency pattern in which the target word appears, the words that co-occur with it are presented in descending order of the values obtained with selected association measure value. Each word that co-occurs is displayed in the format *lemma (l:m)*, where *l* is the base 2 logarithm of the co-occurrence frequency, and *m* the value of the selected measure. The user may change the association measure without having to re-entering the information about the current word.

When the target word is a verb, the corresponding *lex-gram* shows the words appearing as its subject, direct complement, other essential complements, prepositional complements (eventually adjuncts) and the adverbs that modify the verb. In the case of an adjective, the *lex-gram* shows the adverbs that modify it and the nouns that it modifies. For an adverb, the *lex-gram* presents other adverbs that modify it and the adjectives, nouns, verbs, and other adverbs that the target adverb modifies. For this category, it also shows how many times the adverb modifies an entire sentence. Finally, from the *lex-gram*, and by clicking on any co-occurrent word, it is also possible to obtain the details of this specific collocation and view a set of sentences, taken from the *corpus*, that illustrate that pattern.

4 Evaluation

For evaluating the *Deep Syntax Explorer* performance, the CETEMPúblico [31] *corpus*, processed by STRING, was used. The processed *corpus* occupies 237 GB, divided into 20 parts with 12 GB each. Each part has 210 XML files with 60 MB each. The extraction tool was executed in a x86_64 Unix machine, with 16 CPUs Intel(R) Xeon(R)E5530 2.40GHz and 48 390 MB of memory.

For a single XML file from the *corpus* (with 60.8 MB), the extraction tool is executed in 32.5 seconds, where 9.76 seconds are consumed by the XIP-API to import the file, 10.65 seconds by the co-occurrence information extraction and 11.69 seconds are used to store this information and the sentences in the database. After the execution, that information occupies 4.33 MB. For the following parts of the *corpus*, only the time taken to store the information increases, as the database response time also increases.

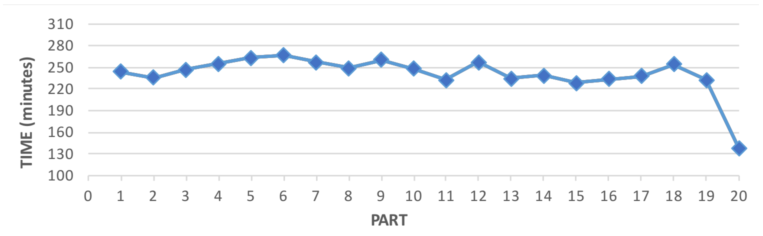


Fig. 5. Time consumed by each part of the CETEMPúblico *corpus*.

Figure 5 depicts the performance of the tool, presenting the time spent on each part of the *corpus*. The entire *corpus* is processed in 4,878 minutes (3 days, 9 hours and 18 minutes), which corresponds to, on average, 243 minutes (4 hours and 3 minutes) by each part of the *corpus* and 1 minute and 9.4 seconds per file. Figure 5 also shows that the execution time is constant along the *corpus*, except in part 20, which is smaller. The association measures are calculated in 146 minutes (2 hours and 26 minutes). The evolution of the database during the process is not constant. Initially, the growth is more pronounced, but it tends to decrease as the process evolves, because, in the beginning, the database is empty, and after processing some files fewer words have to be added. At the end of the process, the database has 6.8 GB. With the association measures data (0.5 GB), this value grows to 7.3 GB. During the process of data extraction from the *corpus*, 308,573 different word *lemmas* were collected, which produced 11,244,852 different co-occurrence patterns occurring 51,321,751 times in the *corpus*.

The evaluation of the web application consisted in measuring the time the application took to show the *lex-gram* for a word. It consists mainly in server-side execution time, running the different SQL queries for each word POS. Table 1 shows the results from this evaluation. Two *lemmas* were used for each POS, one with the highest frequency and another with a median frequency (around 1,000 instances). The time for the word with greater frequency is slightly lower than for the word with smaller frequency. In order to obtain a faster system response, indexes were implemented in the database tables to reduce the time for each SQL query.

Table 1. Time the application takes to show the *lex-gram* to a word.

	Noun		Verb		Adverb		Adjective	
Lemma	`caneta`	`país`	`otimizar`	`ser`	`nitidamente`	`não`	`lindo`	`grande`
Frequency	998	196,140	972	2,533,385	964	1,362,286	997	263,518
Time (s)	0.102	0.111	0.103	0.760	0.037	0.174	0.067	0.256

Table 2 shows the time the application takes to show the examples of sentences for a co-occurrence. The higher the frequency of the co-occurrence, the lower is the waiting time. This is due to the size of the examples' table, since, for a co-occurrence with low frequency, much of the table may need to be iterated.

Table 2. Time the application takes to show the examples for a co-occurrence pattern.

Co-occurence	**Frequency**	**Time (s)**
`(centro,país)`	789	0.024
`(país,praticamente)`	46	0.025
`(país,populoso)`	16	0.035

5 Conclusion

This paper presented *Syntax Deep Explorer*, a tool created for extracting co-occurrence patterns from *corpora* processed by STRING. The tool takes advantage of the rich lexical resources of STRING, as well as of its sophisticated syntactic and semantic analysis, and finds the syntactically-based co-occurrences patterns of a given word *lemma* storing that information in a database. Then, the tool calculates different association measures, producing a *lex-gram* with the co-occurrence statistical information, a snapshot representing the main distributional patterns of a given word. The *lex-gram* also makes it possible to move from any given pattern to another co-locate of interest found within. Furthermore, STRING rich lexical resources feature a large number of multiword expressions, which are processed in the same way as any simple (single-word) unit. Thus, it is also possible to analyse multiwords' distribution and their co-occurrence patterns. Results from evaluation show that the runtime of the extraction tool remains constant throughout the *corpus*, while the size of the database does not grow linearly, indicating that information is not being repeated. The web application response time also allows fast queries to the database.

In the future, it is necessary to increase the number of *corpora* present in the database and to allow the comparison of different *lex-gram* for the same *lemma* across *corpora*. The automatic creation of *thesauri* from the stored distributional information is also envisaged.

Acknowledgment This work was supported by national funds through FCT–Fundação para a Ciência e a Tecnologia, ref.UID/CEC/50021/2013. Thanks to Neuza Costa (UAlg) for revising the final version of this paper.

References

1. Aıt-Mokhtar, S., Chanod, J.P., Roux, C.: Robustness beyond shallowness: Incremental deep parsing. Natural Language Engineering 8, 121–144 (2002)
2. Bick, E.: The Parsing System PALAVRAS. Automatic Grammatical Analysis of Portuguese in a Constraint Grammar Framework. Arhus University Press (2000)
3. Bick, E.: DeepDict - A Graphical Corpus-based Dictionary of Word Relations. In: Proceedings of NODALIDA 2009. NEALT Proceedings Series. vol. 4, pp. 268–271. Tartu: Tartu University Library (2009)
4. Biemann, C., Bordag, S., Heyer, G., Quasthoff, U., Wolff, C.: Language-independent methods for compiling monolingual lexical data. In: Proceedings of CicLING. pp. 215–228. Springer (2004)
5. Carapinha, F.: Extração Automática de Conteúdos Documentais. Master's thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa (June 2013)
6. Chen, P.: The entity-relationship model—toward a unified view of data. ACM Transactions on Database Systems 1(1), 9–36 (1976)
7. Church, K., Hanks, P.: Word Association Norms, Mutual Information, and Lexicography. Computational Linguistics 16(1), 22–29 (1990)
8. Codd, E.: A relational model of data for large shared data banks. Commun. ACM 26(6), 64–69 (1983)
9. Dice, L.: Measures of the Amount of Ecologic Association Between Species. Ecology 26(3), 297–302 (Jul 1945)
10. Diniz, C., Mamede, N., Pereira, J.: RuDriCo2 - A Faster Disambiguator and Segmentation Modifier. In: INFORUM II. pp. 573–584 (September 2010)
11. Diniz, C., Mamede, N., Pereira, J.D.: RuDriCo2 - a faster disambiguator and segmentation modifier. In: Simpósio de Informática - INForum. pp. 573–584. Universidade do Minho, Portugal (2010)
12. Dunning, T.: Accurate methods for the statistics of surprise and coincidence. Computational Linguistics 19(1), 61–74 (1993)
13. Hagège, C., Baptista, J., Mamede, N.: Identificação, Classificação e Normalização de Expressões Temporais em Português: a Experiência do Segundo HAREM e o Futuro. In: Mota, C., Santos, D. (eds.) Desafios na Avaliação Conjunta do Reconhecimento de Entidades Mencionadas: o Segundo HAREM, chap. 2, pp. 33–54. Linguateca (2008), `http://www.inesc-id.pt/ficheiros/publicacoes/5758.pdf/`
14. Hagège, C., Baptista, J., Mamede, N.: Portuguese Temporal Expressions Recognition: from TE Characterization to an Effective TER Module Implementation. In: 7th Brazilian Symposium in Information and Human Language Technology. pp. 1–5. STIL '09, Sociedade Brasileira de Computação, São Carlos, Brazil (2009)
15. Hagège, C., Baptista, J., Mamede, N.J.: Reconhecimento de entidades mencionadas com o xip: Uma colaboração entre o inesc-l2f e a xerox. In: Mota, C., Santos, D. (eds.) Desafios na avaliação conjunta do reconhecimento de entidades mencionadas: Actas do Encontro do Segundo HAREM (Aveiro, 11 de Setembro de 2008). Linguateca (2009)
16. Hagège, Caroline; Baptista, J., Mamede, N.J.: Caracterização e processamento de expressões temporais em português. Linguamática 2(1), 63–76 (2010)
17. Kilgarriff, A., Baisa, V., Bušta, J., Jakubíček, M., Kovář, V., Michelfeit, J., Rychly, P., Suchomel, V.: The Sketch Engine: ten years on. Lexicography 1(1), 7–36 (Jul 2014)
18. Kilgarriff, A., Rychly, P., Tugwell, D., Smrz, P.: The Sketch Engine. In: Proceedings of Euralex. vol. Demo Session, pp. 105–116. Lorient, France (Jul 2004)

19. Mamede, N., Baptista, J., Diniz, C., Cabarrão, V.: STRING: An Hybrid Statistical and Rule-Based Natural Language Processing Chain for Portuguese. In: PROPOR 2012. vol. Demo Session (April 2012)
20. Mamede, N.J., Baptista, J.: Nomenclature of chunks and dependencies in Portuguese XIP Grammar 4.5. Technical report, L2F-Spoken Language Laboratory, INESC-ID Lisboa, Lisboa (January 2016)
21. Manning, C., Schütze, H.: Foundations of Statistical Natural Language Processing. MIT Press, Cambridge, MA, USA (1999)
22. Marques, J.S.: Anaphora Resolution. Master's thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa., Lisboa (2013)
23. Maurício, A.: Identificação, Classificação e Normalização de Expressões Temporais. Master's thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa, Lisboa (November 2011)
24. Nobre, N.: Resolução de Expressões Anafóricas. Master's thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa (June 2011)
25. Oliveira, D.: Extraction and Classification of Named Entities. Master's thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa (2010)
26. Pereira, S.: Linguistics Parameters for Zero Anaphora Resolution. Master's thesis, Universidade do Algarve and University of Wolverhampton (2010)
27. Quasthoff, U., Richter, M., Biemann, C.: Corpus Portal for Search in Monolingual Corpora. In: Proceedings of the 5th LREC. pp. 1799–1802 (2006)
28. Ribeiro, R.: Anotação Morfossintática Desambiguada do Português. Master's thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa. (March 2003)
29. Rychly, P.: Manatee/Bonito – A Modular Corpus Manager. In: P. Sojka, A. Horák (eds.) RASLAN 2008. pp. 65–70. Masaryk University, Brno (2007)
30. Rychly, P.: A Lexicographer-friendly Association Score. In: RASLAN 2008. pp. 6–9. Masarykova Univerzita, Brno (2008)
31. Santos, D., Rocha, P.: Evaluating CETEMPúblico, a Free Resource for Portuguese. In: Proceedings of the 39th Annual Meeting of ACL. pp. 450–457. ACL '01, Association for Computational Linguistics, Stroudsburg, PA, USA (2001)
32. Silberschatz, A., Korth, H., Sudarshan, S.: Database System Concepts. Connect, learn, succeed, McGraw-Hill Education (2010)
33. Sinclair, J.: Corpus, concordance, collocation. Oxford University Press (1991)
34. Smadja, F., McKeown, K., Hatzivassiloglou, V.: Translating collocations for bilingual lexicons: A statistical approach. Computational Linguistics 22(1), 1–38 (Mar 1996)
35. Vicente, A.M.F.: LexMan: um Segmentador e Analisador Morfológico com Transdutores. Master's thesis, Instituto Superior Técnico, Universidade de Lisboa, Lisboa (June 2013)