TSeg – A Text Segmenter for Corpus Annotation

Felipe Rodrigues[1], **Richard Semolini**[1]
Norton Trevisan Roman[2], **Ana Maria Monteiro**[1]

[1]Campo Limpo Paulista Faculty (FACCAMP)
R. Guatemala 167, Jd. América – 13231-230 – Campo Limpo Paulista – SP – Brazil

[2]School of Arts, Sciences and Humanities – University of São Paulo (EACH-USP)
Av. Arlindo Béttio 1000, Ermelino Matarazzo – 03828-000 – São Paulo – SP – Brazil

rodrigues_felipe7@hotmail.com, richard-se@uol.com.br
norton@usp.br, anammont.per@gmail.com

Abstract. *This paper describes TSeg – a Java application that allows for both manual and automatic segmentation of a source text into basic units of annotation. TSeg provides a straightforward way to approach this task through a clear point-and-click interface. Once finished the text segmentation, the application outputs an XML file that may be used as input to a more problem specific annotation software. Hence, TSeg moves the identification of basic units of annotation out of the task of annotating these units, making it possible for both problems to be analysed in isolation, thereby reducing the cognitive load on the user and preventing potential damages to the overall outcome of the annotation process.*

1. Introduction

Within research on corpus linguistics, a considerable amount of time is usually devoted to annotating corpora according to some predefined scheme. Before starting annotating something, however, one has to define *what* must be annotated, *i.e.* the Basic Unit of Annotation [Roman 2007] (also known as Elementary Discourse Unit [van der Vliet et al. 2011]). Although existing systems usually require the user to perform both unit segmentation and annotation altogether (*e.g.* [Orăsan 2003, Ogren 2006, O'Donnell 2008]), this approach conceals a dangerous confounding of variables.

Suppose, for example, that some researcher develops a very easy, intuitive and reliable annotation scheme. This scheme, however, must be applied to quite subjective units of annotation. Upon testing the scheme with some volunteers, the researcher notices a very low agreement amongst them. He then renders the scheme unfit for his needs, ruling it out of any further consideration. Nevertheless, the reason for so bad a result did not lie in the scheme itself, but instead in the low agreement about the segmentation procedure. For this reason, we believe both tasks should come separated in time, so as to avoid any misleading conclusions.

Splitting up texts into small units, however, is not an easy task. Although such units may sometimes deal with something more readily identified, such as words or sentences (*e.g.* [Beineke et al. 2004, Craggs and Wood 2004]), there are cases in which more elaborated (and sometimes more subjective) units are required, such as clauses (*e.g.* [Roman 2007, van der Vliet et al. 2011]) and text segments (*e.g.* [Rubin et al. 2004]), for example.

On this account, despite some efforts in the automatic identification of units like paragraphs (*e.g.* [Bolshakov and Gelbukh 2001]), morphemes (*e.g.* [Golcher 2006]), sets of structured data (*e.g.* [Zhao et al. 2008]) and even text segments, usually split according to their topic (*e.g.* [Beeferman et al. 1999, Kazantseva and Szpakowicz 2011, Kern and Granitzer 2009, Utiyama and Isahara 2001]), there still are units, such as clauses and dialogue acts [Reidsma et al. 2005], amongst others, that cannot be easily automatically determined. To deal with these, the only suitable method is to employ humans to identify which parts of the text belong to which unit [Varasai et al. 2008], usually with the aid of some annotation tool, in order to improve efficiency and accuracy.

In this paper, we introduce TSeg – a tool for text segmentation that allows for both manually defined and automatic segmentation of units[1]. TSeg was created in an attempt to take the identification of basic units of annotation out of the task of annotating these units, so that both problems, although related, could be analysed and assessed in isolation. Through a clear point-and-click interface, it was designed to provide a straightforward way to approach the problem, as it would be expected from annotation tools [Orăsan 2003]. Making the annotation task simpler, besides having an impact on the GUI's design [Reidsma et al. 2005], reduces the cognitive load on the user, preventing potential damage to the overall outcome of the annotation.

The rest of this paper is organised as follows. Section 2 presents an overview of current systems for text segmentation, highlighting their weaknesses and strengths. In Section 3 we describe TSeg in detail, from the user interface to the underlying XML data encoding. In this section we also make a comparison between TSeg's main features and the systems described in Section 2, justifying each of our choices. Finally, in Section 4, we present our conclusions and avenues for future work.

2. Related Work

Current tools for text segmentation range from web applications (*e.g.* [Verhagen 2010]) to stand-alone systems (*e.g.* [Maeda et al. 2008]). Our system – TSeg – belongs to the second group, *i.e.* a stand-alone tool. The reason for this choice lies in that, although representing an interesting alternative, web applications depend on web connections, something that might not be readily available at the time of the annotation, thereby reducing the convenience for annotators to carry out the task [Birnbaum 2004].

Differently from TSeg, none of the existing systems seems to deal with embedded independent units, *i.e.* units that, however defined within another unit, must be treated as if they were two different (and sometimes totally independent) units. This is an important requirement when one deals with some linguistic constructions, such as clauses, for example, in which one clause may be placed inside another, but without being part of it (even though they might be semantically or syntactically related). In such cases, annotation tools must make this independence clear and, although many systems do allow for embedded and overlapping units (*e.g.* [O'Donnell 2008, Verhagen 2010]), independence still seems to be out of their toolboxes.

Another difference between TSeg and other applications for the same task lies in the way TSeg approaches the problem. Within some systems (*e.g.* [Verhagen 2010,

[1]Currently, only words, sentences and paragraphs can be automatically identified.

O'Donnell 2008]), users have to worry about the management of multiple layers, keeping track not only of the specific layer with which they are working, but also the overall layer structure. This extra complexity naturally increases the cognitive load on the annotators, with some possible negative effects on the annotation outcome.

TSeg, on the other hand, has the user focus on a single task – the text segmentation itself. Although multiple-layered systems may also be used to this end, their allowance for both segmentation and annotation through the same interface makes them more complicate to use. The problem, in this case, is again the amount of information the user is presented at the time of the segmentation, whereby the mere knowledge that there are other layers and annotation schemes involved might draw the user's attention away from the segmentation task once in a while, decreasing the segmentation accuracy.

Since, within TSeg's data model, segmentation and segment annotation are taken to be two absolutely distinct problems, there is no need to overload the user with many annotation layers or other details external to the segmentation task. In this case, annotation of segments should be performed by a different tool, better suited to the applied annotation scheme. TSeg's output could then be used as input data to such tools as MMAX2 [Müller and Strube 2006], for example, which takes an already segmented text and adds some annotations to it in a stand-off manner, whereby the data to be annotated are kept in a separate XML file, while annotations are stored in separate documents, somehow linked to the base document [Ide and Brew 2000].

3. TSeg: A Tool for Segmenting Texts

TSeg's main purpose is to provide researchers with a tool for the task of text segmentation into basic units of annotation, *i.e.* the identification (either manual or automatic) of text spans representing the smallest units to which some (possibly more than one) annotation scheme should be applied. Designed mainly as a Computational Linguistics tool, TSeg also allows for the automatic identification of (and segmentation of the source text into) some standard units, namely words, sentences and paragraphs. Other more elaborated units, such as clauses, speech acts and the like, must be manually determined by the user.

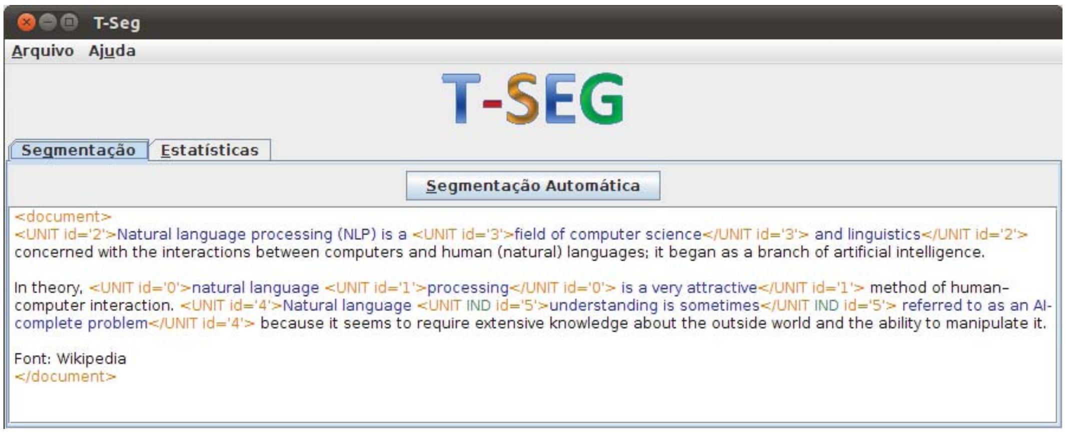
This focus on segmentation, with no regards to segment classification, makes TSeg simpler than other systems that deliver both tasks through the same interface, and which allow for multi-layered annotations (*e.g.* [O'Donnell 2008, Verhagen 2010]). Within TSeg, annotation is thought to be better executed by more scheme-specific tools designed for the annotation effort at hand. TSeg takes then a stand-off approach to the problem, delivering the base data – an XML file – to be linked by annotations built with more specialised tools (*e.g.* [Müller and Strube 2006]).

Finally, the fact that TSeg comes as a stand-alone application, makes it suitable for use whenever and wherever annotators find it more convenient. Output files from different annotators might then be collected by the researcher and analysed, with the aid of some specialised software, for inter-annotator agreement and other statistics. Also, since TSeg was developed in Java, it can be run in different platforms, thereby increasing its portability amongst annotators. In what follows, we will take a closer look at TSeg's usage and its data model, making a parallel, whenever necessary, to existing annotation systems.

3.1. Data Model

Despite the efforts in search for text encoding standards for language resources (*e.g.* [Przepiórkowski and Bański 2009, Verhagen 2010]), none of them has so far emerged as the *de facto* standard for text annotation. There is, however, some understanding that, amongst other things, the data representation format should preferably comply with XML, making it more succinct and independent of the software used to create it [Müller and Strube 2006, O'Donnell 2008]. There also seems to be a tendency to ensure that new annotations can be added to datasets without interfering with already existing ones – *i.e.* there is a preference for stand-off annotations (*e.g.* [Müller and Strube 2006, O'Donnell 2008, Verhagen 2010]).

Along these lines, TSeg's data model comprises a text file annotated with XML tags, representing the segments defined by the user. The main document starts with a <document> tag, with its corresponding </document> coming at the end of the file. Segments are represented as text spans between <UNIT> and </UNIT> tags (akin to the <word> and </word> tags, in [Müller and Strube 2006]), as illustrated in Figure 1. Every UNIT tag has, in turn, a corresponding identification number, so it can be linked by other files in a stand-off manner.



```
<document>
<UNIT id='2'>Natural language processing (NLP) is a <UNIT id='3'>field of computer science</UNIT id='3'> and linguistics</UNIT id='2'>
concerned with the interactions between computers and human (natural) languages; it began as a branch of artificial intelligence.

In theory, <UNIT id='0'>natural language <UNIT id='1'>processing</UNIT id='0'> is a very attractive</UNIT id='1'> method of human-
computer interaction. <UNIT id='4'>Natural language <UNIT IND id='5'>understanding is sometimes</UNIT IND id='5'> referred to as an AI-
complete problem</UNIT id='4'> because it seems to require extensive knowledge about the outside world and the ability to manipulate it.

Font: Wikipedia
</document>
```

Figure 1. XML data encoding.

Along with its identification, some units may also contain the word "IND", as shown in the figure, meaning that this is an independent embedded unit. Independent embedded segments are to be used whenever some text span belongs to one segment, but not to the longer segment that holds it. In a sense, such units may be understood as representing a discontinuity in the host segment – some sort of interruption or flow break. Under this model, TSeg covers all aspects elicited by [Reidsma et al. 2005] for segmentation, to wit:

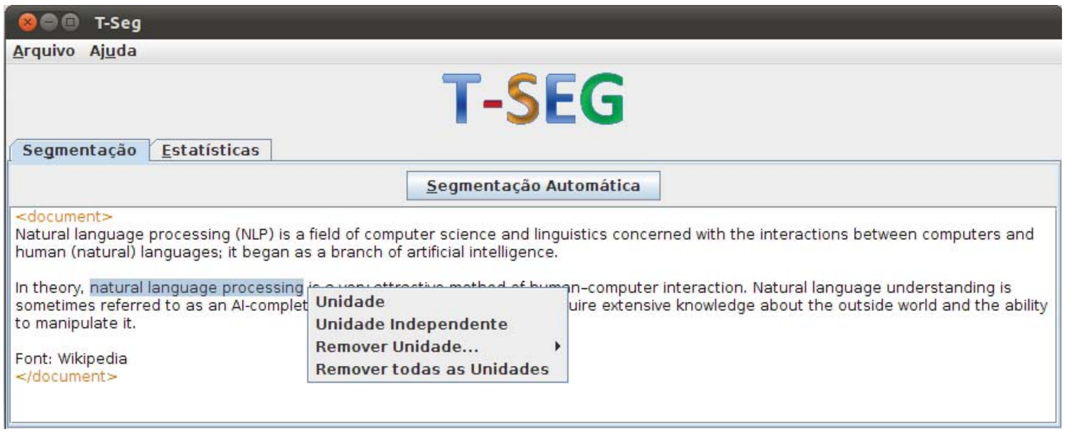
- Overlapping parts: TSeg allows for overlapping segments, in which two or more segments share the same text span. In Figure 1, this is illustrated by segments 1 and 0. Segments may also be embedded to any depth. In fact, embedded segments are but segments in which one is entirely overlapped by another.
- Interleaving parts: Interleaving segments mean that two or more segments, although independent, have their parts partially ordered in an interleaved way. In

TSeg, this is achieved by defining multiple segments, and making their identification number the same.

- Discontinuous parts: With TSeg, discontinuous segments may be determined either by defining one segment and then creating an embedded independent segment within it (*i.e.*, taking the text span belonging to the independent segment out of the host segment), or defining separate segments, and assigning them the same identification number.
- Input coverage: The segmentation may or may not cover the entire input text, *i.e.* all spans may belong to some unit (usually as a result of automatic segmentation), or some may be left aside, depending on the user's choice.
- Segment size: Segments may differ in size, at the user's will, going from single words to arbitrary text spans.

3.2. Usage

TSeg takes as input a raw text file. Upon loading the file into the system, the user may then edit it, by using the mouse to select the desired text span, right-clicking on it, and selecting the appropriate action. In this case, actions can be define a Unit ("Unidade"); define an Independent Unit ("Unidade Independente"); Remove a unit ("Remover Unidade..."), should the user be willing to delete any existing unit; and Remove all units ("Remover todas as unidades"), as shown in Figure 2.



<document>
Natural language processing (NLP) is a field of computer science and linguistics concerned with the interactions between computers and human (natural) languages; it began as a branch of artificial intelligence.

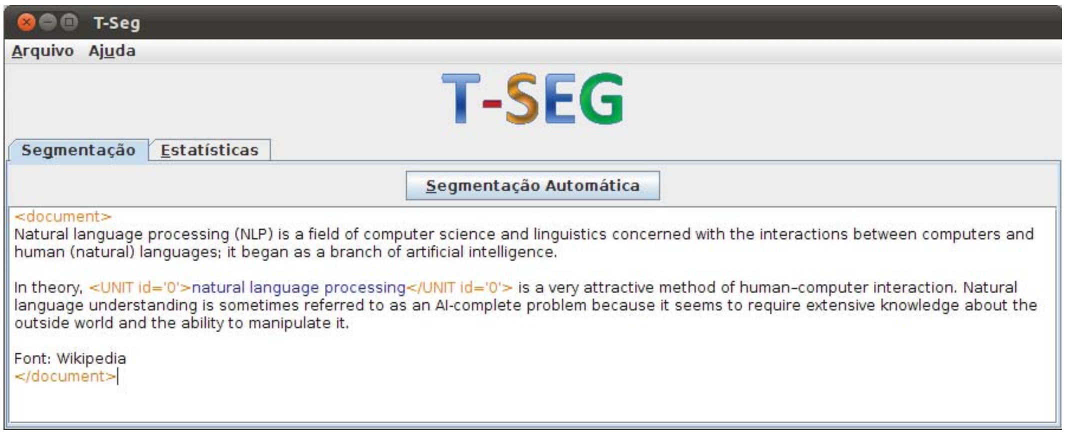
In theory, natural language processing [illegible]n-computer interaction. Natural language understanding is sometimes referred to as an AI-complet [illegible] uire extensive knowledge about the outside world and the ability to manipulate it.

Font: Wikipedia
</document>

Figure 2. Defining a unit in TSeg.

Once defined a unit, TSeg will add the appropriate XML coding to the source text (Figure 3). For the ease of visualisation, XML tags, unmarked text, and annotated spans are shown in different colours[2].

[2]In its current version, TSeg shows the user all XML tags. In a future version we will make this transparent to annotators, hiding all codification details.



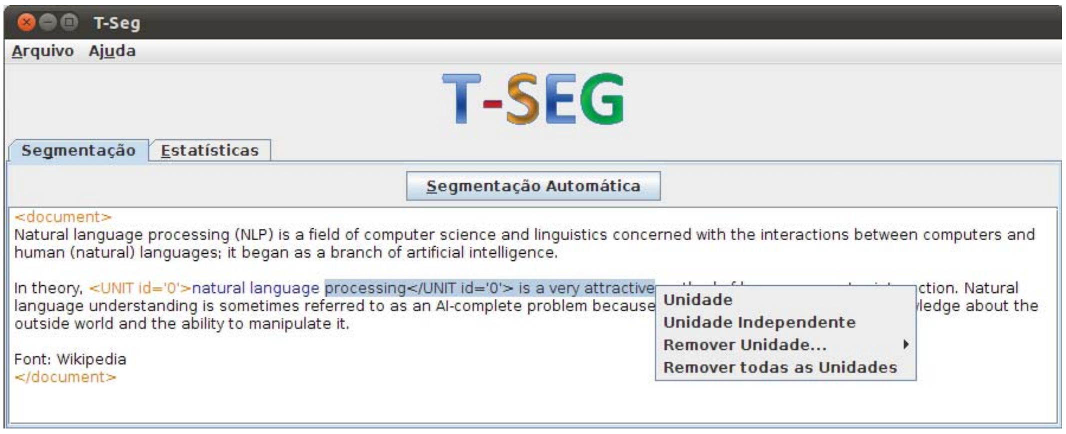
```
<document>
Natural language processing (NLP) is a field of computer science and linguistics concerned with the interactions between computers and
human (natural) languages; it began as a branch of artificial intelligence.

In theory, <UNIT id='0'>natural language processing</UNIT id='0'> is a very attractive method of human-computer interaction. Natural
language understanding is sometimes referred to as an AI-complete problem because it seems to require extensive knowledge about the
outside world and the ability to manipulate it.

Font: Wikipedia
</document>
```

Figure 3. An already defined unit.

Overlapping units are defined the same way: by selecting a text span (with part of it already inside another unit) and turning it into a different unit, as shown in Figures 4 and 5. Embedded and embedded independent units follow the same process, except that for the embedded independent ones the user should choose the *Independent Unit* ("Unidade Independente") option, instead of *Unit* ("Unidade").



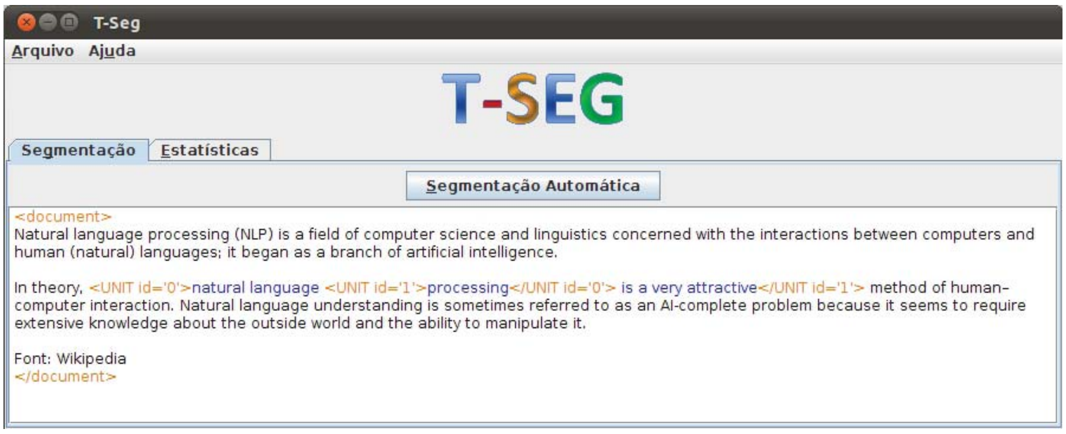
```
<document>
Natural language processing (NLP) is a field of computer science and linguistics concerned with the interactions between computers and
human (natural) languages; it began as a branch of artificial intelligence.

In theory, <UNIT id='0'>natural language processing</UNIT id='0'> is a very attractive                         ction. Natural
language understanding is sometimes referred to as an AI-complete problem because                          ledge about the
outside world and the ability to manipulate it.

Font: Wikipedia
</document>
```

Figure 4. Selecting an overlapping span.

To have the input text automatically segmented, the user must click on the *Automatic Segmentation* ("Segmentação Automática") button, choosing the type of unit that should be applied to the text (Figure 6). Currently, TSeg provides only three options: words ("Palavras"), sentences ("Sentenças") and paragraphs ("Parágrafos"). For sentences, punctuation marks commonly used to end them, such as full stops, question marks, or exclamation marks, are used as separators. Paragraphs boundaries comprise the end of a sentence along with a new line. Finally, word boundaries comprise blank characters, such as tabulator, empty space etc. Figure 7 shows a sample text already segmented in sentences.

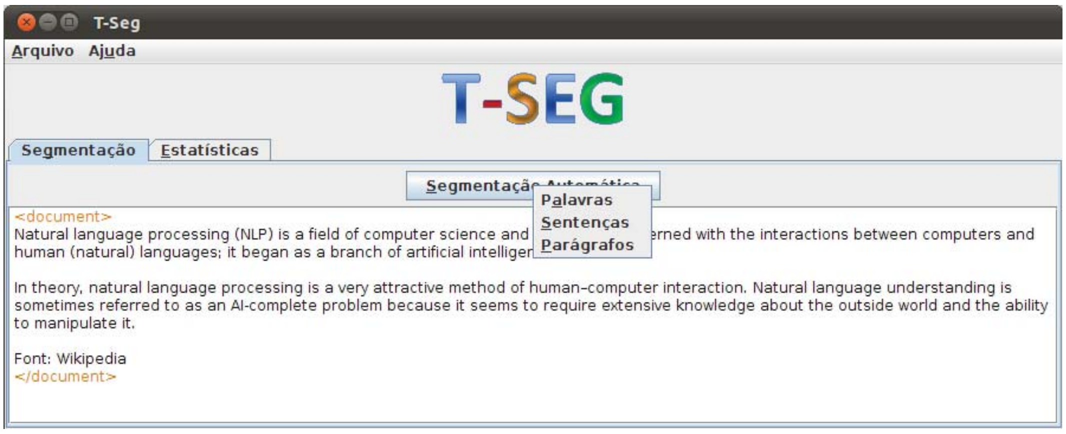


```
<document>
Natural language processing (NLP) is a field of computer science and linguistics concerned with the interactions between computers and
human (natural) languages; it began as a branch of artificial intelligence.

In theory, <UNIT id='0'>natural language <UNIT id='1'>processing</UNIT id='0'> is a very attractive</UNIT id='1'> method of human-
computer interaction. Natural language understanding is sometimes referred to as an AI-complete problem because it seems to require
extensive knowledge about the outside world and the ability to manipulate it.

Font: Wikipedia
</document>
```

Figure 5. Defining an overlapped unit.

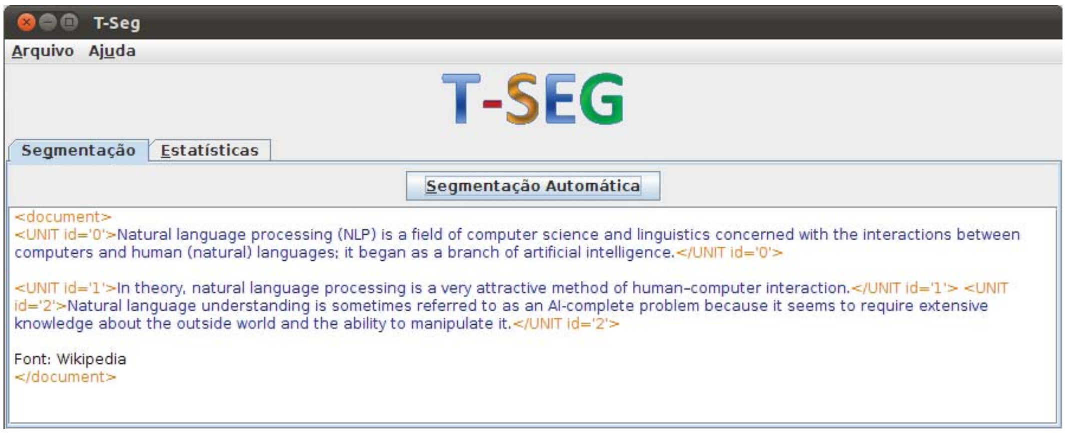


```
<document>
Natural language processing (NLP) is a field of computer science and                erned with the interactions between computers and
human (natural) languages; it began as a branch of artificial intelliger

In theory, natural language processing is a very attractive method of human-computer interaction. Natural language understanding is
sometimes referred to as an AI-complete problem because it seems to require extensive knowledge about the outside world and the ability
to manipulate it.

Font: Wikipedia
</document>
```

Figure 6. Choices for automatic segmentation.



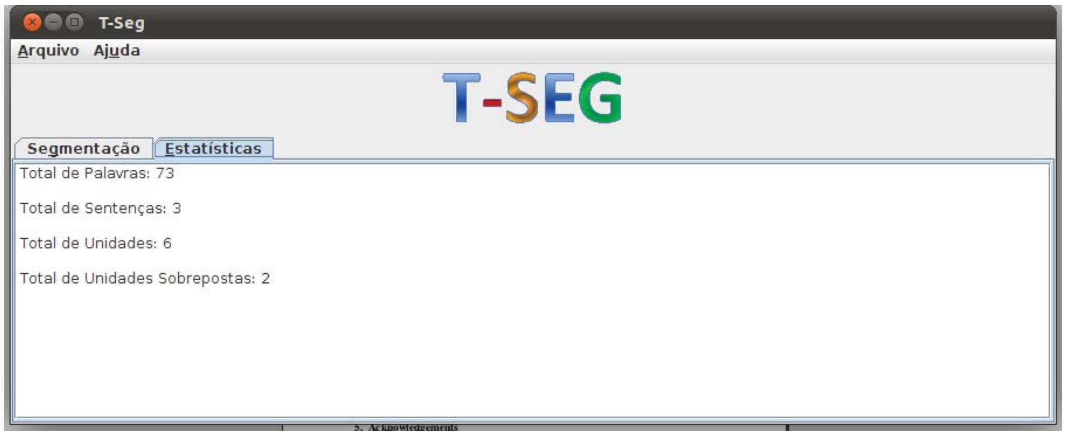
```
<document>
<UNIT id='0'>Natural language processing (NLP) is a field of computer science and linguistics concerned with the interactions between
computers and human (natural) languages; it began as a branch of artificial intelligence.</UNIT id='0'>

<UNIT id='1'>In theory, natural language processing is a very attractive method of human-computer interaction.</UNIT id='1'> <UNIT
id='2'>Natural language understanding is sometimes referred to as an AI-complete problem because it seems to require extensive
knowledge about the outside world and the ability to manipulate it.</UNIT id='2'>

Font: Wikipedia
</document>
```

Figure 7. Text segmented in sentences.

Finally, and following [O'Donnell 2008], when clicking on the *Statistics* ("Estatísticas") tab, the user is shown some general statistics. These are the number of words ("Total de Palavras"), sentences ("Total de Sentenças"), units ("Total de Unidades") – either automatically or manually defined segments – and the amount of overlapping (and embedded) units ("Total de Unidades Sobrepostas"). Figure 8 show the statistics for Figure 1.



Total de Palavras: 73

Total de Sentenças: 3

Total de Unidades: 6

Total de Unidades Sobrepostas: 2

Figure 8. Document statistics in TSeg: Number of words, Number of sentences, Number of units and Number of overlapped units.

4. Conclusion

In this paper we have described TSeg – a tool for text segmentation. Primarily designed for stand-off annotations, TSeg complements a number of existing annotation tools (*e.g.* [Müller and Strube 2006]), which depend on already segmented texts to go on with their tasks. In separating source data segmentation from annotation, TSeg also contributes to a better assessment of annotation schemes, by removing any confounding that the annotator's choice for segments might introduce into the application of the annotation scheme.

Although developed without multi-user support, TSeg's output files from different users may be input to specialised statistical software packages, for inter-annotator agreement and other related statistics. With such results at hand, researchers might ensure that their annotation schemes would be applied on solid grounds, *i.e.*, that results of the application of their annotation schemes would not suffer from any misinterpretation annotators might have about the segment they were annotating.

Alongside the annotation of the source text, TSeg also provides some basic statistics, such as the total amount of words, sentences, units (*i.e.* segments) and overlapping units. These give the researcher better grips on the properties of the text with which he is dealing, to the extent that, by knowing the amount of overlapping segments, for example, he can have an idea on what level of confounding to expect from the text annotation, should the available categories be mutually exclusive.

As for follow-ups, we intend to hide all codification details away from the user (as done in [Orăsan 2003, Müller and Strube 2006]), making the source text clearer at the program's interface. Next, we will carry out some usability tests with the tool, so as to measure how hard it is to use it. Also, and since there are some arguments for local statistical analyses (*e.g.* [O'Donnell 2008, Verhagen 2010]), we intend to develop a management module to TSeg, giving it the power to do some multi-user statistical tests, such as inter-annotator agreement, for example. Finally, it would be interesting delivering TSeg in different languages too (something that can be achieved via Java's internationalisation facilities).

5. Acknowledgements

The development of this tool was supported by the Campo Limpo Paulista Faculty, through its Undergraduate Scholarship Program.

References

Beeferman, D., Berger, A., and Lafferty, J. (1999). Statistical models for text segmentation. *Machine Learning*, 34:177—-210.

Beineke, P., Hastie, T., Manning, C., and Vaithyanathan, S. (2004). An exploration of sentiment summarization. In *AAAI Spring Symposium: Exploring Attitude and Affect in Text: Theories and Applications*, Stanford, USA. Technical Report SS-04-07.

Birnbaum, M. H. (2004). Human research and data collection via the internet. *Annual Review of Psychology*, 55:803–832.

Bolshakov, I. A. and Gelbukh, A. F. (2001). Text segmentation into paragraphs based on local text cohesion. In *Proceedings of the 4th International Conference on Text, Speech and Dialogue (TSD '01)*, pages 158–166, Zelezna Ruda, Czech Republic.

Craggs, R. and Wood, M. M. (2004). A two dimensional annotation scheme for emotion in dialogue. In *AAAI Spring Symposium: Exploring Attitude and Affect in Text: Theories and Applications*, Stanford, USA. Technical Report SS-04-07.

Golcher, F. (2006). Statistical text segmentation with partial structure analysis. In *Proceedings of 8th Conference on Natural Language Processing (KONVENS 2006)*, pages 44–51, Konstanz, Denmark.

Ide, N. and Brew, C. (2000). Requirements, tools, and architectures for annotated corpora. In *Proceedings of Data Architectures and Software Support for Large Corpora*, pages 1–5, Paris, France. European Language Resources Association.

Kazantseva, A. and Szpakowicz, S. (2011). Linear text segmentation using affinity propagation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)*, pages 284—-293, Edinburgh, Scotland, UK.

Kern, R. and Granitzer, M. (2009). Efficient linear text segmentation based on information retrieval techniques. In *Proceedings of the International Conference on Management of Emergent Digital EcoSystems (MEDES '09)*, pages 167–171, Lyon, France.

Maeda, K., Lee, H., Medero, S., Medero, J., Parker, R., and Strassel, S. (2008). Annotation tool development for large-scale corpus creation projects at the linguistic data consortium. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco.

Müller, C. and Strube, M. (2006). Multi-level annotation of linguistic data with MMAX2. In Braun, S., Kohn, K., and Mukherjee, J., editors, *Corpus Technology and Language Pedagogy: New Resources, New Tools, New Methods*, pages 197–214. Peter Lang, Frankfurt a.M., Germany.

O'Donnell, M. (2008). The uam corpustool: software for corpus annotation and exploration. In *Proceedings of the XXVI Congreso de AESLA*, Almeria, Spain.

Ogren, P. V. (2006). Knowtator: A plug-in for creating training and evaluation data sets for biomedical natural language systems. In *Proceedings of the 9th International Protégé Conference*, Stanford, USA.

Orăsan, C. (2003). Palinka: A highly customisable tool for discourse annotation. In *Proceedings of the 4th SIGdial Workshop on Discourse and Dialog*, pages 39—43, Sapporo, Japan.

Przepiórkowski, A. and Bański, P. (2009). Which xml standards for multilevel corpus annotation? In *Proceedings of the 4th Language and Technology Conference, LTC 2009, Poznan*, pages 400–411, Poznan, Poland.

Reidsma, D., sa Jovanović, N., and Hofs, D. (2005). Designing annotation tools based on properties of annotation problems. In *Measuring Behavior 2005, 5th International Conference on Methods and Techniques in Behavioral Research*.

Roman, N. T. (2007). *Emoção e a Sumarização Automática de Diálogos*. PhD thesis, Instituto de Computação – Universidade Estadual de Campinas, Campinas, São Paulo.

Rubin, V., Stanton, J., and Liddy, E. (2004). Discerning emotions in texts. In *AAAI Spring Symposium: Exploring Attitude and Affect in Text: Theories and Applications*, Stanford, USA. Technical Report SS-04-07.

Utiyama, M. and Isahara, H. (2001). A statistical model for domain-independent text segmentation. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics (ACL '01)*, Toulouse, France.

van der Vliet, N., Berzlánovich, I., Bouma, G., Egg, M., and Redeker, G. (2011). Building a discourse-annotated dutch text corpus. *Bochumer Linguistische Arbeitsberichte*, 3:157–171. ISSN: 2190-0949.

Varasai, P., Pechsiri, C., Sukvari, T., Satayamas, V., and Kawtrakul, A. (2008). Building an annotated corpus for text summarization and question answering. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco.

Verhagen, M. (2010). The brandeis annotation tool. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, pages 3638–3643, Valletta, Malta.

Zhao, C., Mahmud, J., and Ramakrishnan, I. (2008). Exploiting structured reference data for unsupervised text segmentation with conditional random fields. In *Proceedings of the SIAM International Conference on Data Mining (SMD08)*, Atlanta, USA.