THE UAM CORPUSTOOL: SOFTWARE FOR CORPUS ANNOTATION AND EXPLORATION

MICHAEL O'DONNELL
Universidad Autónoma de Madrid

ABSTRACT

This paper describes the capabilities of the UAM CorpusTool, software for the annotation of text corpora. The software allows the user to annotate a corpus of text files at a number of linguistic layers, which are defined by the user. For instance, one can annotate texts at the document layer (e.g., text type, writer characteristics, register, etc.), semantic-pragmatic levels, and at syntactic levels (e.g., clause, phrases, etc.). At each annotation layer, the user defines a hierarchy of tags appropriate for that layer, using a graphical tool. The user then annotates the text at each layer by swiping the text to indicate a segment, and then assigning features from the tag hierarchy at that layer. The paper also descibes some of the supporting functionalites of the software, including corpus search, automatic tagging based on lexical pattern matching, and production of statistics.

Keywords: Corpora, annotation tools, text markup.

RESUMEN

Este artículo describe las características del UAM CorpusTool, un software para la anotación de corpus de texto. Este software permite al usuario anotar un corpus de archivos de texto en distintos niveles lingüísticos previamente definidos por el usuario. Por ejemplo, uno puede anotar los textos a nivel de documento (Ej., tipo de texto, características del escritor, registro, etc.), en niveles semántico-pragmáticos, y en niveles sintácticos (Ej., cláusula, sintagmas, etc.). Utilizando una herramienta gráfica, el usuario define una jerarquía de etiquetas apropiadas para cada nivel de anotación. A continuación, el usuario anota el texto en cada nivel, seleccionando primero el texto para indicar un segmento, y asignándole características elegidas de

entre la jerarquía de etiquetas definidas para ese nivel. Este artículo describe también otras funcionalidades añadidas al software, tales como instrumento de búsqueda en el corpus, etiquetador automático basado en la correspondencia de patrones léxicos, y producción de informes estadísticos.
Keywords: Corpus, instrumentos de anotación, markup de texto.

1. Introduction

In recent years, there has been growing use of software to assist users in the annotation of text corpora. Part of this growth has been due to the increasing number of linguists interested in exploring linguistic patterns in text which cannot be explored with simple concordancers. Linguistic features which cannot yet be automatically tagged (e.g., semantic and pragmatic features) need to be identified by a human and good annotation software facilitates this task.

Additionally, the last 20 years has seen a growing interest in statistical-based language processing (e.g., machine translation, parsing, etc.). These systems typically require a training set, which is usually provided by human annotators. Human-annotated texts can also be used as 'golden standards', to facilitate the evaluation of such systems.

The *UAM CorpusTool* has been developed to address these needs. It is at base a system which allows a user to apply tags to segments of text. An interface presents a text, the user can swipe a segment of text, and is then prompted to select tags to assign to that segment (following a user-provided tag hierarchy).

For most needs, an annotation task requires more than one text file, and thus *CorpusTool* assumes that an annotation 'project' consists of a set of text files, each to be annotated with the same annotation scheme.

Annotation tasks also frequently require annotation at multiple 'layers', for instance, one might assign features to the document as a whole (e.g., text type, writer characteristics, register, etc.), or to segments within the text (at semantic-pragmatic layers, syntactic units (e.g., clause, phrases, etc.), or at a lexical level. *CorpusTool* allows the user to define any number of layers, and to provide a hierarchically-organised tagging scheme for each layer, using a graphical tool.

Given the complexity of dealing with collections of documents each annotated at a number of linguistic layers, some sort of 'project

management' functionality is required. The main interface of *CorpusTool* is thus not an annotation window as in most other tools, but a project management window, displaying the files included in the study, and the layers of annotation which the user has defined for these files. From this interface, the user can add or delete files, open files for annotation, and define the annotation layers to apply to these files.

While the central task of *CorpusTool* is annotation, it also provides other functionalities to support the user, such as cross-layer searching, semi-automatic tagging, production of statistical reports from the corpus, visualisation of the tagged corpus, inter-coder reliability statistics, etc. These functionalities will be explored below.

UAM CorpusTool is free, and works on Macintosh and Windows. It is perhaps the most user-friendly of the corpus annotation tools currently available, and is well-documented.

This paper provides an overview of the software, including project management, file annotation, its cross-layer searching capabilities, and the generation of statistical reports.

2. Data Handling

The data used by an annotation system can be divided into two parts: the content that is annotated, and the files used to store annotations and metadata such as annotation schemes.

2.1 Content Files

UAM CorpusTool is aimed at the annotation of text, it does not handle multimodal formats such as audio, video, images, multimodal documents, etc. CorpusTool is thus most comparable with text annotation tools such as *Gate* (Cunningham et al 2002), *MMAX-2* (Müller and Strube 2006), and *Knowtator* (Ogren 2006). *CorpusTool* currently accepts source texts only as plain text, it does not currently handle formats such as PDF, RTF, XML, etc.

Within these plain text files, *CorpusTool* accepts any character encoding (e.g., UTF-8), etc, and can thus handle texts in any writing system, e.g., Russian, Chinese, Arabic, etc. When a new text file

is incorporated into a project, *CorpusTool* offers the user a menu of encodings, with the selected encoding being the one which is most likely (given the bytes in the file, and a series of heuristics). The selected encoding can be changed later if the wrong one was selected.

While CorpusTool can process any encoding, there remain two problems in regards to display of some non-Western writing systems. Firstly, display of Arabic is left-to-right rather than right-to-left at present. Secondly, due to deficiencies in the GUI package used (Python/Tkinter), display of some writing systems is not yet supported under MacOSX, but this will be resolved in a pending release of Python.

2.2 Annotation and metadata storage

CorpusTool stores its annotation data using XML. As with most modern annotation tools, it uses 'stand-off annotation' (Thompson and McKelvie 1997), whereby the original text is left untouched, and annotation files refer to either character ranges in the original text, or ranges of tokens. Stand-off annotation provides far better support for projects with multiple annotation layers of the same text, or annotations by alternative users. Stand-off annotation allows for partially overlapping segments, not possible in normal XML.

```
<?xml version='1.0' encoding='utf-8'?>
<document>
  <segments>
    <segment id='1' start='158' end='176'
             features='participant;human' state='active'/>
    <segment id='2' start='207' end='214'
             features='participant;organisation;company'
             state='active'/>

 ...
  </segments>
</document>
```

Figure 1: Annotation Storage Example

CorpusTool maintains one XML file for each annotation layer of each text file. An example of one of these files is shown in Figure 1. The

format used is rather succinct in comparison with the mark-up used in most annotation tools, in order to reduce file size.

CorpusTool does not currently support any of the text encoding standards (e.g., TEI (Sperberg-McQueen and Burnard 1994), CES (Ide 1998), AIF (Bird et al 2000), or ISO TC37/SC4. This is on the agenda for the future, as users identify which of the many standards are important for their work. Import/export options will be provided.

3. The Project Window

The central concept in *CorpusTool* is a 'project', which consists of a corpus of text files, and a number of layers at which these text files are being annotated. Currently, only one type of annotation is supported, assigning features to segments of text.

The first step with the software is to define a project. One is led through this by a wizard, providing the name of the project, where to store it, and which text files to include (others can be added later).

Figure 2 shows the main window of *CorpusTool*, the *Project* window. The space at the top of the window shows that this project has three layers defined, one for assigning features to the document as a whole (the "Register" layer), one for assigning features to clause segments, and another for annotating NPs ("Participant" layer). Note that none of these layers are predefined: users define layers, and the feature scheme that are used to annotate segments at that layer. Users can add as many annotation layers as they need.

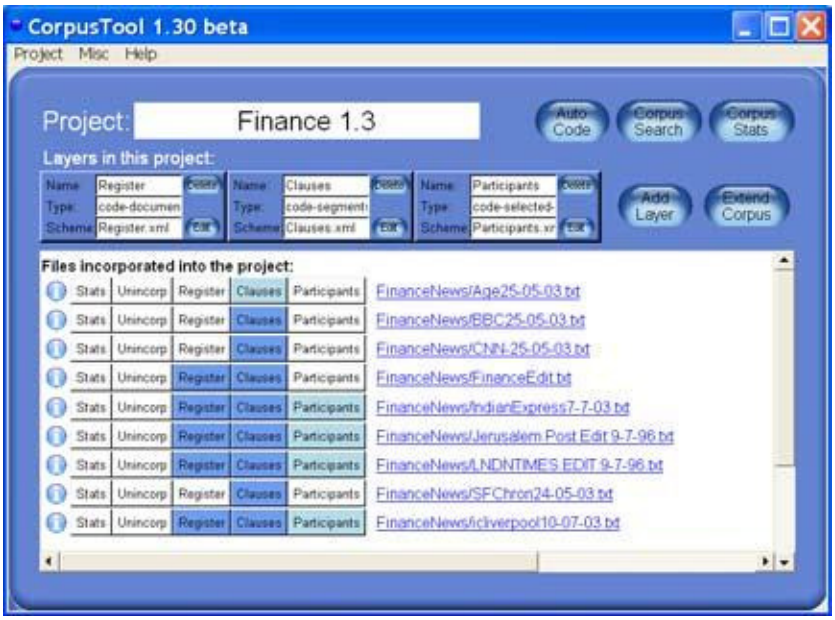

Figure 2: The Project Window

Below the Layer space, there is a space showing all the files so far incorporated into the project. Each of these files has a number of buttons:

- *The "i" button*: allows setting metadata for the text file, including encoding, language, and display font.
- *Stats*: this button opens a window giving, for the text, word counts, the number of sentences, and some information as to lexical density (if closed-class word lists are provided for the language).
- *Unincorp*: removes the file from the project.

Each of the remaining buttons corresponds to an annotation layer, and clicking on that button opens the annotation window for that file/ layer.

4. Scheme Editing

Clicking on the "Edit" button within each layer's box opens up a window which shows the hierarchical scheme of features which defines the layer (see Figure 3). One can edit this scheme by clicking on features to add further distinctions (including cross-classification), rename or delete a feature. Any change to the scheme will be propagated through the annotations, e.g., renaming a feature here will change the feature name in all files annotated at this layer. The graphical presentation of the scheme can be saved in various formats, for inclusion in publications and websites.

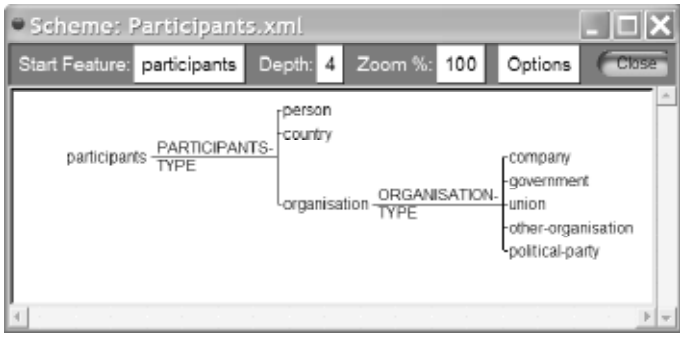

Figure 3: The Scheme Window

5. Annotation Windows

Currently, *CorpusTool* supports only one type of annotation: assigning features to segments. Future releases will support additional types of annotation, such as structuring (syntactic structure, document structure rhetorical structure, etc.), co-reference linking, and error analysis.

Figure 4 shows an annotation window for a file. The user creates a segment by swiping the mouse over the text. The start and end of a segment can be moved simply by dragging the end of the segment to the desired location. CorpusTool has no problem with overlapping segments. Segments can be embedded to any depth, and partial overlaps are allowed.

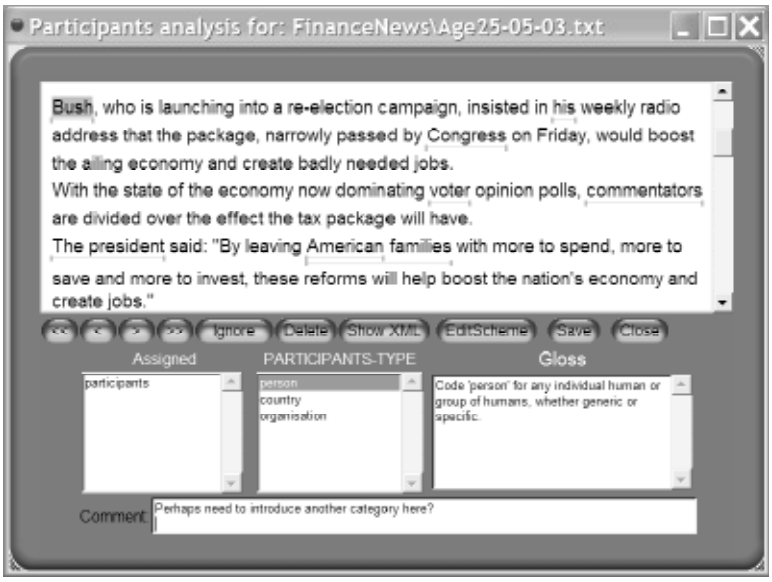

Figure 4: An annotation window

The space in the lower half of the window is used for assigning features to the currently selected segment (the features are those defined in the scheme editor for this layer). The features on the left are those already assigned, and those in the centre are the current choice. A feature is assigned by double-clicking on it. One can delete an assigned feature by double-clicking on the feature in the assigned feature box.

A 'gloss' can be associated with features in the coding scheme, helping the user decide which of the choices is most appropriate. The user can also associate a comment with each segment, for instance, as a note to fellow coders that the coding is in doubt.

6. Corpus Search

The *Corpus Search* window allows the user to search for instances in the annotated corpus which match some criteria. The search query is specified using a menu-driven widget. In the most basic mode, the user searches for segments tagged with a given feature, or feature combination ('X and Y'; 'X or Y'; 'X and not Y').

Searching across layers is possible: searching for segments ‘containing’ other segments, or searching for segments ‘in’ other segments. For instance:

```
clause containing person&subject in editorial&english
```

matches:

- segment tagged with ‘clause’
- which contain a segment tagged as ‘person’ and also tagged as ‘subject’
- which is in a document tagged as both ‘editorial’ and ‘english’.

The user can also search for segments containing strings, e.g., clause containing ‘after’. The same mechanism allows searching for lexical patterns. For instance, clause containing ‘be% @participle’ will match all segments tagged as clause which contain any inflection of the root ‘be’, followed by any participle verb (a basic pattern for passive clauses in English). Figure 5 shows the results of a search, displayed in KWIC table mode.

This type of search does not depend on POS tagging of the text, but rather uses dictionary lookup to see if a word matches a word-class or inflection. As such, sometimes false matches occur, but the need for POS tagging is avoided. Currently only a lexicon for English is provided, with 190,000 unique surface forms. A Spanish lexicon with 250,000 terms will soon be incorporated. Lexicons for other languages will be provided in the future.

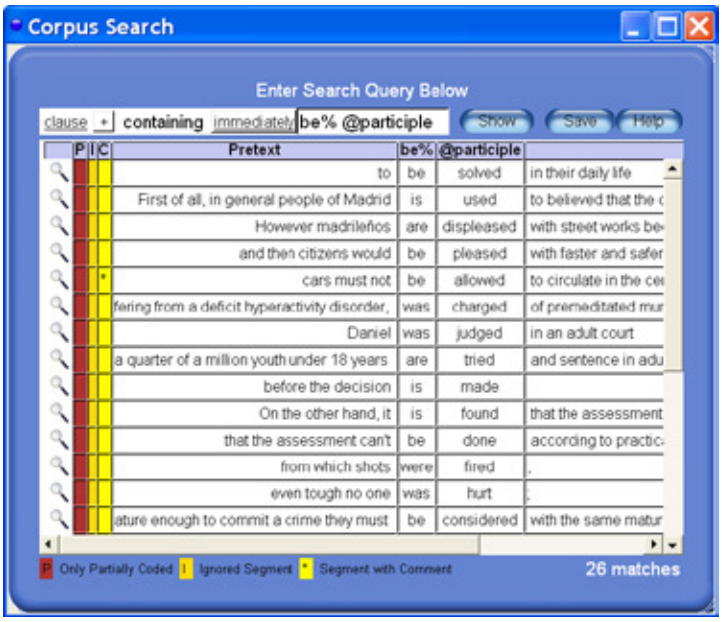

Figure 5: The Search Window

7. AUTO-CODING

In the previous section, we saw how, for English at least, concordance patterns can be used to find instances in the corpus. It would be useful if we could then assign a tag to the returned items, as a permanent indicator that they belong in this set. For instance, above we searched for passive clauses. it would be useful to tag all of the identified clauses as passive.

CorpusTool includes an interface for specifying coding rules of the form:

```
select <feature> if contains <pattern>
```

Figure 6 shows this window after finding all matches with the search pattern. In this window, each hit includes a check button. The user reads through the list of hits, unchecking any false matches. Once finished, they can click the "Code Selected" button, and all checked hits are assigned the 'passive' feature.

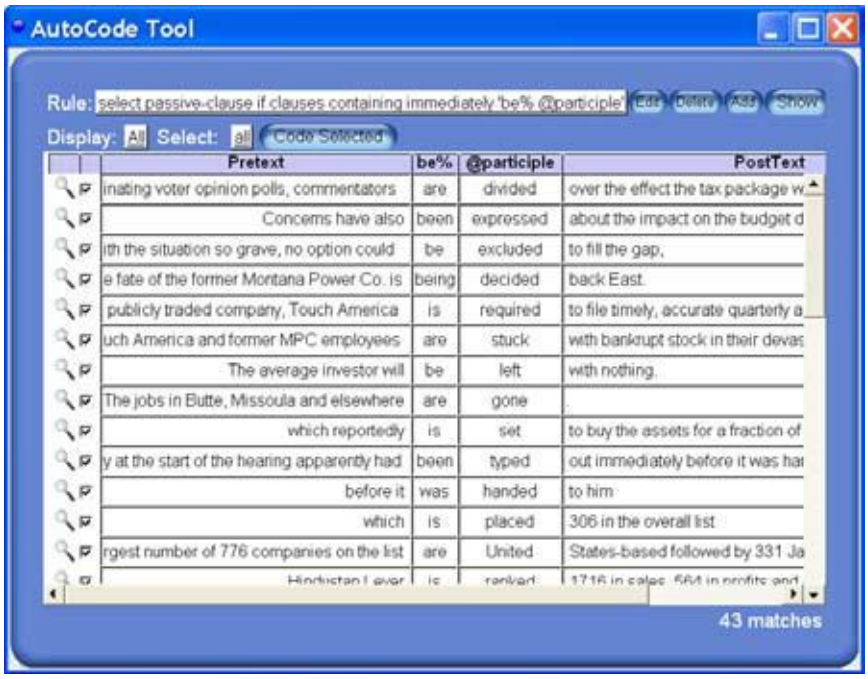

Figure 6: The Autocode Window

Once passives are coded, the user can specify a rule like the following to code all remaining clauses as 'active':

```
select 'active' if clause and not passive
```

Auto-coding in this way takes a common feature of concordancing tools such as Wordsmith and makes it into a powerful tagging feature. I have used lexical patterns to identify syntactic features such as clause modality, primary tense, perfective and progressive aspect, infinitive, gerund and participle clauses, etc. Since the dictionary also classifies verbs in terms of Process Type (cf. Halliday 1985), we can also autocode clauses as material, mental, verbal, behavioural or relational. However, given the ambiguity of verbs, many false matches are presented and need to be eliminated.

It is clear that not all syntactic patterns can be identified, or be identified reliably, in terms of lexical patterns. This feature should be seen as a partial bridge between the needs of large corpora studies

requiring automatically tagged corpora, and the need to perform studies of features at a level above POS, where full parsing is not an option.

8. Statistics

Most typically, annotation tools depend on external software for statistical analysis of the corpus text and annotations. However, there are arguments for in-house statistical analysis: the software already has access to all the data (text and annotations), and knows the structure of this data.

UAM CorpusTool provides various statistical analyses of the corpus. This includes:

- *Descriptive statistics* of feature tagging of the corpus, or of a sub-corpus defined in the same manner as in Corpus Search.
- *Contrastive statistics* of feature tagging of any two subsets of the corpus, e.g., we might contrast the nature of participants which appear in editorials vs. those used in front-page news.
- *General Text Statistics* provides details such as word-counts, average segment length, lexical density, pronominal usage, etc., in any specified set of segments in the corpus. This can also be done contrastively.
- *Word Propensity* provides a listing of the words which are strongly associated with a given sub-corpus in comparison with the rest of the corpus, and a measure of the degree of association.

In all cases, statistics can be saved to tabbed-delimited text files for import into external statistical processing packages (e.g., Excel, SPSS). There is an option to export statistics aggregated per corpus file (one row per file), to support file-based analyses, such as document clustering, or PCA.

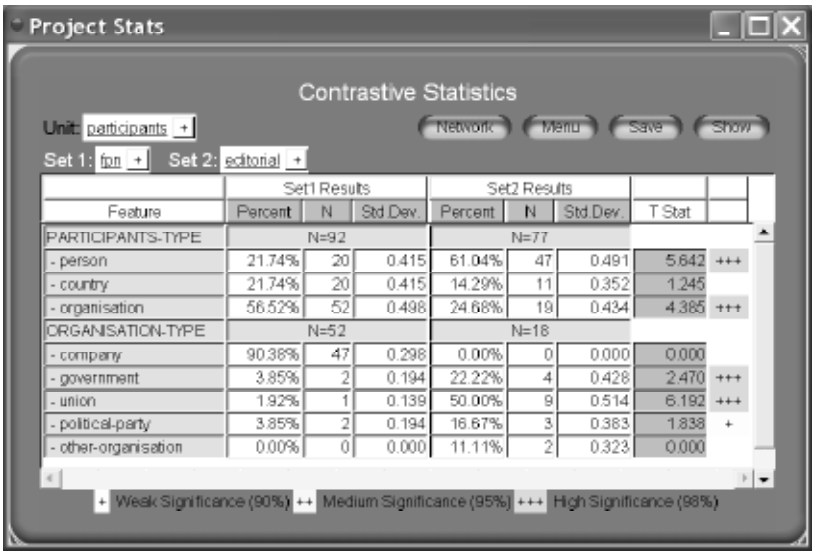

Figure 7: One of the Statistics Windows

9. Conclusion

This paper has summarised some of the features included within UAM CorpusTool, free software for the multi-layer annotation of a corpus of text files. In contrast to many of the available annotation tools, the interface is easy to learn and easy to use. Different tools target different user needs, and no tool can address all needs. This tool is aimed particularly at those wishing to perform linguistic studies, and thus provides on-board search facilities and statistical reporting. Currently no structural annotation is supported (e.g., treebank annotation, rhetorical structure, co-reference linking, etc.), although these will be added in the future.

The author is currently developing features to support multi-coder interaction with the tool. There exists a tool to measure inter-coder agreement between a pair of coders. This is currently limited to segmentation agreement, but will be extended to coder agreement in the near future.

In the longer term, it is planned to support storage of project files online, allowing multiple users to work on project files at the same time,

controlling access to avoid two people working on the same file at the same time.

The software is undergoing continual development, in response to user feedback. As of the end of January, 2008, a year after its first release, CorpusTool has been downloaded 1350 times, to 780 unique CPUs.

References

Bird, S., D. Day, J. Garofolo, J. Henderson, C. Laprun and M. Liberman 2000. 'ATLAS: A Flexible and Extensible Architecture for Linguistic Annotation', *LREC 2000*, Athens: 1699-1706.

Cunningham, H., D. Maynard, K. Bontcheva and V. Tablan 2002. 'GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications', 40th Meeting of the *Association for Computational Linguistics (ACL'02)*. Philadelphia, July 2002.

Halliday, M.A.K. 1985. *An Introduction to Functional Grammar*. London: Edward Arnold.

Ide, N. 1998. 'Corpus Encoding Standard: SGML Guidelines for Encoding Linguistic Corpora', *Proceedings of the First International Language Resources and Evaluation Conference,* Granada, Spain, 463-70.

Müller, C. and M. Strube 2006. 'Multi-Level Annotation of Linguistic Data with MMAX2', in S. Braun, K. Kohn, J. Mukherjee (eds.) *Corpus Technology and Language Pedagogy. New Resources, New Tools, New Methods (English Corpus Linguistics, Vol.3)*. Frankfurt: Peter Lang. 197-214.

Ogren, P.V. 2006. 'Knowtator: a plug-in for creating training and evaluation data sets for biomedical natural language systems', *Proceedings of the 9th International Protégé Conference*. 73–76.

Sperberg-McQueen, C. M. and L. Burnard 1994. *TEI Guidelines for Electronic Text Encoding and Interchange (P3)*. Chicago and Oxford: ACH/ACL/ALLC Text Encoding Initiative.

Thompson, H. and D. McKelvie 1997. "Hyperlink semantics for

standoff markup of read-only documents". *Proceedings of SGML Europe'97*.