View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Edinburgh Research Explorer

THE UNIVERSITY of EDINBURGH

Edinburgh Research Explorer

Using the NITE XML Toolkit on the Switchboard Corpus to study syntactic choice: a case study

Citation for published version:
Carletta, J, Dingare, S, Nissim, M & Nikitina, T 2004, Using the NITE XML Toolkit on the Switchboard Corpus to study syntactic choice: a case study. in Proceedings of The fourth international conference on Language Resources and Evaluation (LREC 2004). Fourth Language Resources and Evaluation Conference (LREC) 2004, Lisbon, Portugal, 26/05/04.

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Publisher's PDF, also known as Version of record

Published In:
Proceedings of The fourth international conference on Language Resources and Evaluation (LREC 2004)

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 05. Apr. 2019

Using the NITE XML Toolkit on the Switchboard Corpus to study syntactic choice: a case study

Jean Carletta*, Shipra Dingare*, Malvina Nissim*, Tatiana Nikitina†

*University of Edinburgh, HCRC Language Technology Group
2, Buccleuch Place, Edinburgh EH8 9LW, UK
J.Carletta,S.Dingare,M.Nissim@edinburgh.ac.uk

†Stanford University, Department of Linguistics
Margaret Jacks Hall, Building 460
Stanford CA 94305, USA
tann@stanford.edu

Abstract

The NITE XML Toolkit (NXT) provides library support for working with multimodal language corpora. We describe our experiences in using it to study discourse effects on syntactic choice using the parsed Switchboard Corpus as a starting point, as a case study for others who may wish to adopt similar techniques using NXT or one of the other libraries that are beginning to emerge. We discuss conversion into the NXT data format; automatic annotation of markables and of constituent length; hand-annotation of markables for animacy information structure, and coreferential links; and data analysis.

1. Introduction

The NITE XML Toolkit (Carletta et al., 2003) is a recent library that has the aim of making it easier to support multimedia corpus research, especially where several types of annotation are present on the same data. It is not particularly difficult to write tools with a specific corpus design and task in mind, or to allow a certain degree of tool configuration, such as tag renaming. However, this practice tends to make tool development expensive and result in suboptimal interfaces — small differences in layout or behaviour can greatly affect utility, especially where hand-annotation is involved. Arguably, it also restricts progress by encouraging the more corpus-oriented researchers to mark up linguistic phenomena and to perform analyses for which support already exists. Because NXT provides library routines from which annotation and analysis tools can be built, it is especially suited for new work on previously unsupported types of annotation. However, its very flexibility makes it harder to see how to use it than for other, more specific tools. We have been using it on the Switchboard Corpus as part of an ongoing study of syntactic choice in spoken dialogue. In this paper, we describe how we performed some common data processing tasks using NXT.

2. The research

Our research goal is to explore syntactic choice in the dialogues in order to provide guidelines by which natural language generators can choose among paraphrases, focusing on the choice of active versus passive construction, dative placement, and the use of prenominal genitives versus "of" phrases. The linguistic literature suggests that syntactic choice is influenced by the animacy of the entities referred to and their newness in the discourse (usually referred to as "information structure"). In addition, it has been suggested that the length of a syntactic constituent affects its placement. Thus our work is structured in terms of identifying discourse entities, or "markables"; annotating them for animacy, information structure, and coreference; adding constituent length to the existing syntactic annotation; for each type of syntactic choice, finding sentences that exhibit the two syntactic variants; and comparing the properties of discourse entities that occur in them.

For this work we chose the Switchboard Corpus (Godfrey, Holliman, & McDaniel, 1992), a collection of spontaneous telephone conversations between speakers of American English on predetermined topics, using the subset of around 650 dialogues that had been parsed by the Penn Treebank Project. The Linguistic Data Consortium distributes this subset as "Treebank 3", in a format amenable to query by tgrep.

3. The NITE XML Toolkit

The core of NXT consists of two types of functionality: routines that load, access, manipulate, and save data according to a particular data model; and an engine that evaluates queries expressed in NXT's Query Language (NiteQL). Several groups plan library support with similar functionality, of which the Atlas project is perhaps the closest in style (*ATLAS Project*, 2000). Data handling functionality for at least one such library, the Annotation Graph Toolkit (AGTK), has already been released (*AGTK: Annotation Graph Toolkit*, n.d.). NXT differs from AGTK in two ways. First, its data model and query language are oriented towards those users who build descriptive analyses of heavily cross-annotated multimodal corpora in preparation for defining appropriate statistical models, and therefore it allows easy access to an expressive range of data relationships, at the expense of processing speed. Second, it supplements the data and query facilities with library routines for building displays and interfaces based on Java Swing. The libraries include a default top level interface that allows one to choose an observation (in this case, a dialogue) and a tool to run on it from those registered in the corpus metadata; audio and video players; a search interface for running queries and displaying the results; basic display layouts such as text areas and trees that synchronize with the data and with the media and search facilities; and standard utilities for things like opening an observation

and saving some annotation. These libraries are intended to make it possible to build tailored end user tools at low cost. There is also a default data display that is never as good as a tailored one but at least allows any corpus in the correct format to be viewed and searched without further programming.

Because NXT is designed for multimodal corpora, the data model can express (and the query language can test, and the interfaces can exploit) not just structural relationships but also temporal ones. However, this project is not using the signals or time-aligned transcriptions that are available in other Switchboard releases, and so we do not comment on these capabilities.

4. Conversion to NXT data format

NXT uses a stand-off XML data format that consists of several XML files that point to each other. We converted from the Penn Treebank bracketed format in which the Switchboard corpus is distributed, using TIGER, an XML-based tool for syntactic query that comes with a ready-made Switchboard converter (Tiger Project, n.d.), as an intermediate step. We did this so that we had a starting point that was easier to parse. (In retrospect, this was probably the wrong decision because NXT's XML is much more similar in structure to the Treebank format than to TIGER's graph-based XML, and because it meant we are not in complete control of the conversion process.) Conversion was performed using a set of XSL stylesheets, one to extract each of the multiple XML files associated with one dialogue. Writing the converter took three to four person-days, most of which was spent trying to understand the original data format.

In order to facilitate the kinds of processing which we have in mind for the data over the longer term, we divided our data into separate XML files representing the orthographic transcription; syntax; turn structure; disfluencies; and movement, or the relationship between traces and their sources. Transcription consists of a flat list of terminals: words, punctuation, traces, and so on. Syntax starts with a flat list of parses and works down through nonterminals, grounding in terminals (which are in the transcription file, but are referenced by pointers that indicate they are to be treated as if they were part of the tree itself). Turn structure is simply a flat list of turns that themselves contain parses as children, again via pointers into the syntax file. Yet another file couples reparanda and repairs into disfluencies by pointing to the appropriate nonterminals using named roles. A movement file similarly links sources with their target traces.

This representation sounds awkward, but it has advantages over the original arrangement of placing the information in a single tree structure, with co-indexing for the crossing links that are sometimes required for disfluency and movement. First, it makes it easier to query the crossing structures, since they are treated on a par with other structures within the data. Although this ease is not particularly important for the initial, syntactic data, it is crucial for a correct understanding of discourse phenomena such as coreference. Second, separating the tags into their various types makes it conceptually simpler to add data using external processes (part-of-speech taggers, named entity recognizers, and the like); each individual file is itself simple. Third, different people can change different data files at the same time without conflict, as long as neither edit the files they point to and both are able to lock complete paths of files pointing to the data they are revising. Fourth, a data set can be loaded whole or in part, speeding up some processing. The NITE XML Toolkit itself treats the data seamlessly no matter whether it is in one file or many.

5. Automatic annotation

We use automatic annotation to identify markables and add the length of syntactic constituents.

5.1. Markables

NXT provides a facility for automatic annotation based on query language matches for queries expressed in NiteQL. Once we had the published syntactic data translated into NXT format, we were able to use this facility to annotate markables corresponding to discourse entities.

Although as a first pass, discourse entities might be thought of as represented by noun phrases, there are in fact a number of arguable exceptions and idiosyncracies about the Switchboard data that make their identification more complicated. Our analysts preferred to omit adverbial noun phrases plus any that were labelled as locative or directive (or were dominated by adverbial phrases that were). Because this was spoken data, they also preferred to omit any data marked as disfluent ("unf") or within disfluent ("edited") constituents. In addition, they wished to consider possessive pronouns (words with part of speech "PRP$") to be markables in their own right if they were directly contained within markables.

A query that matches all markables except those that derive from possessive pronouns can be expressed in NiteQL as follows:

```
($n nt)(forall $up nt):
 (($n@cat == 'NP') or
  ($n@cat == 'WHNP')) and
 (not (($n@subcat ~ /.*ADV.*/) or
       ($n@subcat ~ /.*LOC.*/) or
       ($n@subcat ~ /.*DIR.*/) or
       ($n@subcat ~ /.*UNF.*/)))
 and ((($n != $up) and($up ^ $n)) ->
      ((not ($up@cat == 'EDITED')) and
       (not (($up@cat == 'ADVP') and
             (($up@subcat ~ /.*LOC.*/) or
              ($up@subcat ~ /.*DIR.*/))))))
```

In the query, subcategory conditions employ regular expressions (using the ~ operator) rather than exact string matches because occasionally one nonterminal had two subcategories that we had not bothered parsing into separate descriptors. The conditions involving `$up` which do not relate to its categorization simply restrict matches to nonterminal syntactic constituents that dominate (^) `$n` but are not `$n` itself. Once we had established the query, we were able to add markables to the data by running the AddIndex utility, specifying the query and the tag name "markable" at the command line. This utility adds new tags with the given name, one for each match, that point to whatever matched the first named variable in the query using a role named "at".

Once these markables had been added, it was a simple matter to find the possessive pronoun cases:

```
($w word)(exists $n nt)
(exists $m markable)(forall $up nt):
 ($w@pos = 'PRP$') and ($n ^ $w) and
 ($m >'at' $n) and
 (($up != $n) ->
  (not (($n ^ $up) and ($up ^ $w))))
```

It took discussion among the analysts to determine the exact constraints to express, in which they were aided by our ability to create HTML displays of the chosen markables in context using a simple XSLT stylesheet. A better option, available now but not at the time, is using the syntactic display and search menu available for data analysis (see section 7).

5.2. Constituent length

Our research requires a measure of constituent length in terms of number of words. This would most reasonably be represented using a new descriptor on the constituents themselves, but NXT only comes with command line utilities for adding new tags that point into a data set, not for adding attribute values. Building a utility that adds descriptors of a given name for matches to one query based simply on the frequency of results to another within the match would be easy, but anything more generic would be difficult to provide at the command line. For this reason, we chose to add constituent length using a stylesheet. This approach gives us a simple framework that can be reused, with access to the many and various calculation methods available in XSLT (arithmetic, filtering based on tag properties, sorting, counting, and so on).

6. Hand annotation

Our hand annotation for markables gives their animacy (based on Garretson, O'Connor, Skarabela, & Hogan, to appear) and information structure, including coreferential links (based on a scheme described elsewhere in this volume, Nissim, Dingare, Carletta, & Steedman, 2004).

Figure one, given at the end of the paper, shows our annotation tool for information structure and coreference; this is the most complex of our coding interfaces. Our information structure coding requires each markable to be given one of an enumerated list of status descriptors ("old", "mediated", and "new", along with buckets of various kinds) and optionally, a type subcategorization ("event", "general", and so on). In the data display, markables are shown in parentheses, with the nesting depth of the markable indicated by colour. Markables for which no information status coding has been added are indicated with a small round dot, and the status and type subcategory for the currently selected markable are highlighted with larger balls on the menus. Apart from these indicators, the existing coding is not displayed because the annotators felt that it hindered their reading of the text.

Coreference annotation, then, requires the annotator to link pairs of markables as anaphor and antecedent. In normal practice, one notices a coreference relation when one is processing the anaphor. For this reason, when a link is added, the tool assumes that the currently selected markable should be the anaphor and, if the user immediately selects another markable, assigns it to be the antecedent. When a link is selected, the antecedent is highlighted in pink, and the anaphor in grey.

Our animacy coding tool is similar in style but only allows the user to add a simple animacy descriptor (such as "human", or "organization") to each markable, again from an enumerated list. Because only one coding decision was needed for each markable, this interface moves to the next markable automatically

This arrangement of tools and the individual tool designs arose out of careful discussion between the tool developers and the end users, with a particular focus on minimizing mouse clicks. The annotation tools themselves were well-received. A separate "checking" mode for each tool displays the entire coding at a glance. NXT comes with a range of sample applications for varying interface designs that is growing all the time, and which should, along with current documentation efforts, cut development costs further.

7. Data analysis

In NXT, the query language is the key to data analysis. We cite queries at length in section 5.1 for the reader's inspection because in our experience, assuming users understand how a corpus has been encoded (the tag and attribute names and how tags relate to each other), they either understand queries readily or find them very difficult indeed.

NXT is designed so that when the search facility is called from an application that has a data display, if one selects a match or set of matches in the query result tree, the corresponding sections of the data display will also be highlighted. Although this facility is currently only implemented for data displays based on text areas, it is invaluable for writing and testing queries. We used this facility to develop, for instance, queries that select for different complex syntactic constructions, using a data display of the syntax trees themselves. NXT then includes command line utilities for counting query matches. It is also possible to save the query result tree, which is amenable to normal XML processing or which can without much difficulty be loaded back into NXT with the original data set, making it amenable to treatment by the query language.

Satisfaction with NXT for data analysis depended on what the individual needed to do with it. In brief, those who wrote queries that could not have been expressed in tgrep were happy with the system because they would have had no other way to perform their analysis short of programming, whereas other users felt that they would have been better off using tgrep on the original data format. Their dissatisfaction arose partly from a lack of familiarity with the query language, but also from its relative verboseness and its slowness.

Consider the query for "any noun phrase that does not dominate a verb phrase dominating a prepositional phrase." This is easily expressed in tgrep as "NP !< (VP < PP)". The same query in NXT, using our corpus representation, would be:

```
($np nt)(forall $vp nt)(forall $pp nt):
   (($np ^ $vp) && ($vp ^ $pp) && ($vp@cat==VP)
     && ($np@cat==NP)) -> ($pp@cat!=PP)
```

Although the query length would be halved if we were to represent np, vp, and pp as separate tags rather than nt (non-terminal) tags with category descriptions, this is still less succinct. Verbosity was a particular problem for queries involving immediate sisterhood and leftmost daughters, since these operators are not yet supported directly in the NXT query language.

Tgrep is so succinct because it is designed specifically for syntax trees. On the other hand, because it is special purpose, tgrep does not allow the full use of variables and boolean operators, and therefore cannot execute queries such as "return a noun phrase that dominates or is dominated by a verb phrase" in one search, or express relationships holding between more than two nodes in a tree, for example, "return a noun phrase which is the daughter of a verb phrase and is coindexed with the subject of that verb phrase". Tgrep also does not offer any full equivalent to NXT's existential and forall statements; consequently, it is not possible to search for "a verb phrase dominating only noun phrases" (except by enumerating all of the types not to dominate) or "the first NP daughter". These queries can be expressed directly when using NXT.

Tgrep is fast because queries are limited to individual sentences. NXT can express queries that require not just complete dialogues but complete corpora to be considered as one data set. At present, the NXT implementation does not analyze queries in order to minimize the data set it checks over. For this reason, we have found it essential to limit querying to one dialogue at a time, especially when using multiple instances of the forall and exists operators, and still at times to expect to use batch processing and indexing to save results we wish to re-use.

8. Acknowledgments

This work was carried out under funding from the European Commission (IST-2000-26095, *Natural Interactivity Tools Engineering*) and from Scottish Enterprise (The Edinburgh-Stanford Link, *Paraphrase analysis for improved generation*).

9. References

AGTK: Annotation Graph Toolkit. (n.d.). Retrieved 1 Mar, 2004, from http://agtk.sourceforge.net/.

ATLAS Project. (2000, 6 Feb 2003). Retrieved 1 Mar, 2004, from http://www.nist.gov/speech/atlas/.

Carletta, J., Evert, S., Heid, U., Kilgour, J., Robertson, J., & Voormann, H. (2003). The NITE XML Toolkit: flexible annotation for multi-modal language data. *Behavior Research Methods, Instruments, and Computers, 35*(3), 353-363.

Garretson, G., O'Connor, M. C., Skarabela, B., & Hogan, M. (to appear). *Coding practices used in the project Optimal Typology of Determiner Phrases*, from Boston University Noun Phrase Corpus website, http://npcorpus.bu.edu/.

Godfrey, J., Holliman, E., & McDaniel, J. (1992, March). *Switchboard: Telephone speech corpus for research development.* Paper presented at the International Conference on Acoustics, Speech, and Signal Processing, San Francisco, CA, USA.

Nissim, M., Dingare, S., Carletta, J., & Steedman, M. (2004). *An Annotation Scheme for Information Status in Dialogue.* Paper presented at the Fourth Language Resources and Evaluation Conference, Lisbon, Portugal.

Tiger Project. (n.d., 17 Nov 03). *Linguistic Interpretation of a German Corpus*. Retrieved 1 Mar, 2004, from http://www.ims.uni-stuttgart.de/projekte/TIGER/.

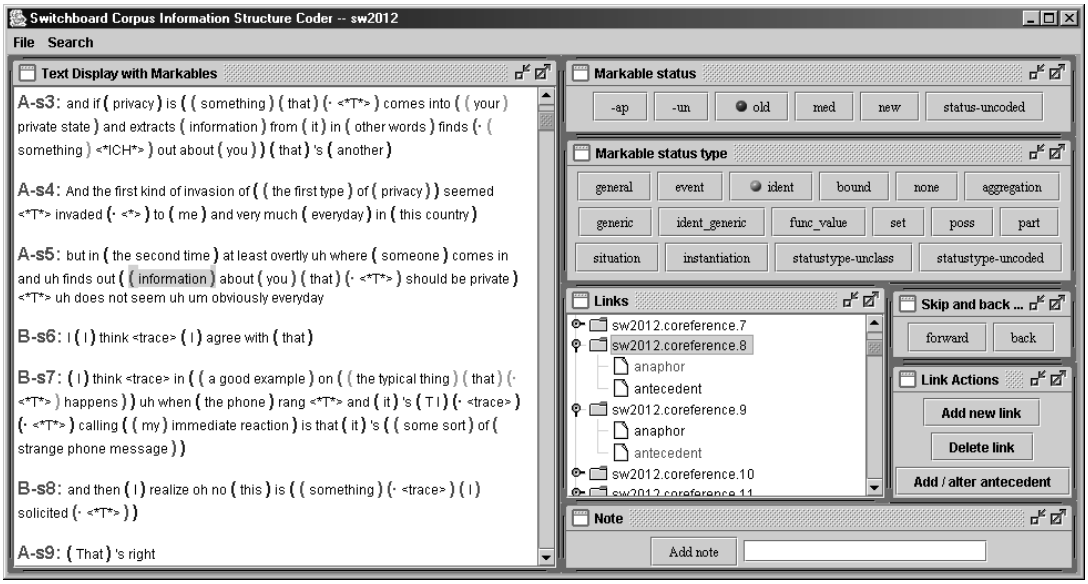

Figure 1: Screenshot of the information status coding tool.