Thomas Aquinas in the TüNDRA: Integrating the *Index Thomisticus* Treebank into CLARIN-D

Scott Martens[1], Marco Passarotti[2]

[1]Eberhard Karls Universität Tübingen
Seminar für Sprachwissenschaft, Wilhelmstr. 19-23, Tübingen, Germany
[2]Università Cattolica del Sacro Cuore
CIRCSE Research Centre, Largo Gemelli 1, Milan, Italy
E-mail: scott.martens@uni-tuebingen.de, marco.passarotti@unicatt.it

Abstract

This paper describes the integration of the *Index Thomisticus* Treebank (IT-TB) into the web-based treebank search and visualization application TüNDRA (*Tübingen aNnotated Data Retrieval & Analysis*). TüNDRA was originally designed to provide access via the Internet to constituency treebanks and to tools for searching and visualizing them, as well as tabulating statistics about their contents. TüNDRA has now been extended to also provide full support for dependency treebanks with non-projective dependencies, in order to integrate the IT-TB and future treebanks with similar properties. These treebanks are queried using an adapted form of the TIGERSearch query language, which can search both hierarchical and sequential information in treebanks in a single query. As a web application, making the IT-TB accessible via TüNDRA makes the treebank and the tools to use of it available to a large community without having to distribute software and show users how to install it.

Keywords: treebank, dependency, web

1. Introduction

TüNDRA (*Tübingen aNnotated Data Retrieval & Analysis*) is a web-based treebank search and visualization platform. Integrating the *Index Thomisticus* Treebank (IT-TB) into TüNDRA has been a challenging task, because the IT-TB has features that require special consideration for search and visualization and that were not supported in the first release of TüNDRA (Martens, 2012). Specifically, the IT-TB uses a dependency-based annotation incompatible with TüNDRA's initial constituency approach. TüNDRA has been adapted to fully support dependency annotation, without making assumptions about projectivity.

2. The *Index Thomisticus* Treebank

The *Index Thomisticus* Treebank (McGillivray et al., 2009) is a Latin language treebank drawn from the contents of the *Index Thomisticus* (Busa, 1980). Presently, the size of the IT-TB is approximately 200,000 words (more than 11,000 sentences).

Begun in 1949, the *Index Thomisticus* was one of the first large corpus projects in computational linguistics and contains around 11 million tokens representing the complete works of Thomas Aquinas as well as texts by other authors related to Aquinas. The corpus has been morphologically tagged, lemmatized and manually corrected. The morphosyntactic disambiguation and syntactic annotation of the *Index Thomisticus* is the goal of the IT-TB, which is part of the wider *Lessico Tomistico Biculturale*, whose goal is the development of a lexicon of the works of Aquinas.

Latin is a highly inflected language with a broadly free word order. This means that equivalent syntactic structures may involve any of a number of word orders. For instance, the order of the words in a sentence like *ergo corrumpitur forma* ("form is thus corruputed[1]") could be changed into *ergo forma corrumpitur*, or *forma ergo corrumpitur*, without affecting the syntactic relations between its components.

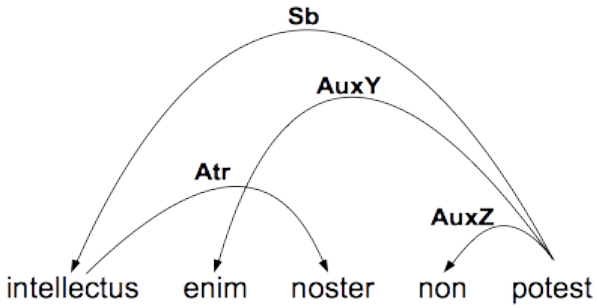

Figure 1: The dependencies of the phrase *intellectus enim noster non potest* as represented in the IT-TB. Arrows indicate the direction of the dependency (from head to dependent), and the arc labels are dependency types described in the IT-TB style guide (Bamman et al., 2007).

Furthermore, phrases in Latin and many similar languages are often non-projective (or discontinuous), i.e. not all the words between the first word in the phrase and the last form a part of that phrase (Marcus, 1965). For example, in the sentence *intellectus enim noster non potest una conceptione diversos modos perfectionis accipere* ("for our intellect cannot accept the concept of different kinds of perfection[2]"), the nominal phrase *intellectus noster* ("our intel-

[1]*Scriptum super Sententiis*, Liber 4, Distinctio 3, Quaestio 1, Articulus 2, Argumentum 2.

[2]*Scriptum super Sententiis*, Liber 1, Distinctio 2, Quaestio 1, Articulus 3, Solutio.

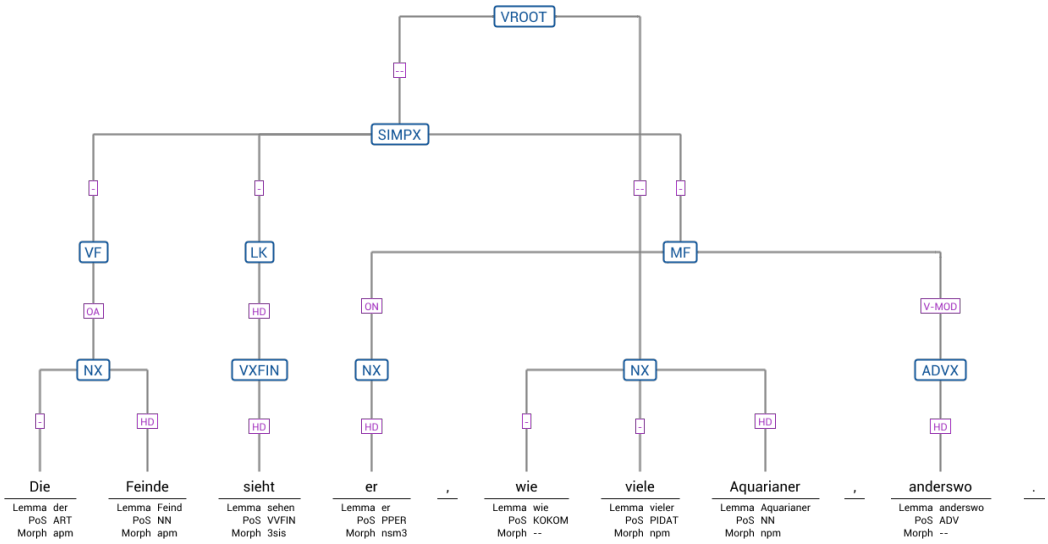

Figure 2: A non-projective hybrid constituency tree from the German language TüBa-D/Z: *Die Feinde sieht er, wie viele Aquarianer, anderswo.* ("He sees enemies, like many Aquarians, elsewhere"). The phrase *wie viele Aquarianer* ("like many Aquarians") is a non-projective element in this analysis of the sentence, coming in the middle of a phrase that does not dominate it.

lect") is discontinuous, as can be seen in Figure 1. Note that the arcs that indicate dependencies in this phrase cross, in contrast to constituency grammar which discourages or, in some linguistic theories and treebanks, forbids such crossing arcs.

This non-projectivity is an important property of dependency treebanks. The number of non-projective edges in dependency treebanks varies dramatically depending on the language and style of the underlying text. They are very rare in some languages: Havelka (2007) identifies 1.37% of edges in a Portuguese treebank as non-projective and 5.9% for a Dutch one. Other languages have very high rates - 15.15% of all edges are non-projective in one treebank of Ancient Greek texts (Mambrini and Passarotti, 2013). In Latin, non-projectivity appears to vary significantly depending on the author and time period of the text, from 3.24% in the IT-TB to 6.65% in the Latin Dependency Treebank, which includes around 55,000 words from Classical era texts (Passarotti and Ruffolo, 2010).

To encode these considerations into a treebank, the IT-TB uses a dependency grammar formalism, similar to that of the Prague Dependency Treebank of Czech (Hajič et al., 1999), that does not require projective syntactic relations. The IT-TB shares this formalism with other Latin treebanks, particularly those developed by the Perseus Digital Library (Bamman et al., 2007)[3].

3. Tools for constituency and projective treebanks

Free word order and non-projectivity pose a number of problems for treebank users and software developers. Although the roots of constituency treebanks are in generative grammar, where a non-projective tree would make little sense, projectivity and constituency are not necessarily synonymous. Some treebanks, for example the TüBA-D/Z treebank of German (see Figure 2), have a broadly constituency structure but are nonetheless non-projective. These kinds of treebanks are often labelled as *hybrid treebanks* to distance themselves from the association of strict projectivity with constituency.

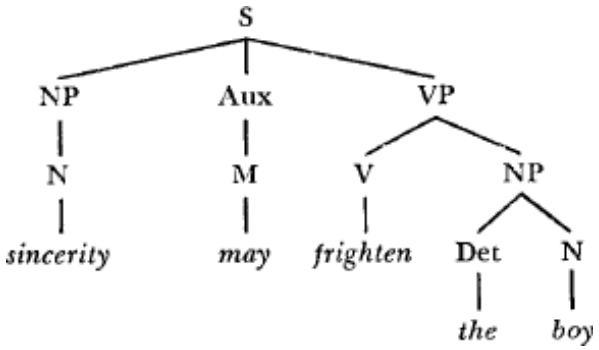

Figure 3: A projective constituency analysis of the sentence "Sincerity may frighten the boy". This analysis is taken from Chomsky (1965) and is representative of the early generative grammar tradition.

Search technologies that are effective for searching sequential information lend themselves much better to treebanks with purely projective tree structures, like the one in Figure 3 than to ones where search terms may appear in multiple orders. For example, non-treebank corpus search programs that are well-adapted to queries on sequences of annotated tokens, such as CQP (Christ and Schulze, 1995), can only search treebanks using the orders of tokens and constituents. It could be used to find appearances of *noster intellectus* from the example in Figure 1, or even *noster* followed by *intellectus* with some words or classes of words in between them, but would require an entirely separate query

[3] http://www.perseus.tufts.edu/.

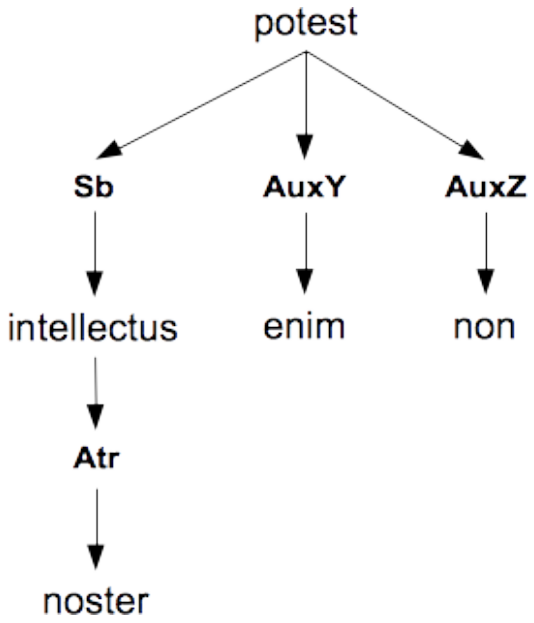

Figure 4: The same Latin phrase as Figure 1 restructured to match the Tgrep2 approach to tree data. Note that the order of the words is lost wherever it does not match the strictly projective tree structure, and the original sequence cannot be queried or retrieved.

to find instances of *intellectus noster*. There is no intuitive way, using those kinds of techniques, to only identify appearances of those two words with some specific syntactic dependency between them.

Tgrep2 (Rohde, 2005) has been designed to search projective constituency trees like the one in Figure 3. It does have the means to search hierarchical syntactic structures, but only where those structures are projective.

Tgrep2 could in principle be used to identify instances of *noster intellectus* and *intellectus noster* where they have a specific dependency. However, it can only do so by imposing projectivity on the elements of the sentence in a way that reorders the words. Figure 4 is an example of the kind of restructuring necessary to shoehorn a non-projective dependency structure into the kind of format Tgrep2 requires. For Latin and similar languages, this kind of restructuring necessarily means losing key information about the sequential structure of the underlying sentences.

Furthermore, because of its origins in searching constituency tree structures in which edges are unlabelled, Tgrep2 has no intuitive means of querying dependencies by their type. The information encoded in the edges of a dependency treebank can be forced into a constituency tree by adding extra tree nodes, as Figure 4 shows. Although this representation is formally equivalent to labelled edges, it is counter-intuitive and cluttered.

Similar problems with the formal assumptions underlying treebanks undermine the ability of software designed with constituency grammar assumptions to search and display treebanks like the IT-TB.

TIGERSearch (Lezius, 2002) is a widely used software suite for hybrid constituency treebanks and has support for both viewing and searching non-projective trees. Hybrid treebanks may have labelled and non-projective edges. Non-projective dependency trees can be converted into non-projective constituency trees of exactly the kind that TIGERSearch supports by assuming that each token with descendants is the head of a phrase of some kind, as displayed in Figure 5. While this kind of transformation is trivial enough from a formal standpoint, it is counter-intuitive for users familiar with dependency formalisms and drawing styles and adds superfluous information and unnecessary complexity to visualizations and queries. Furthermore, forcing users to introduce constituents into their treebanks defeats much of the purpose of doing dependency grammar. Good software should not impose unneeded structures on users's data.

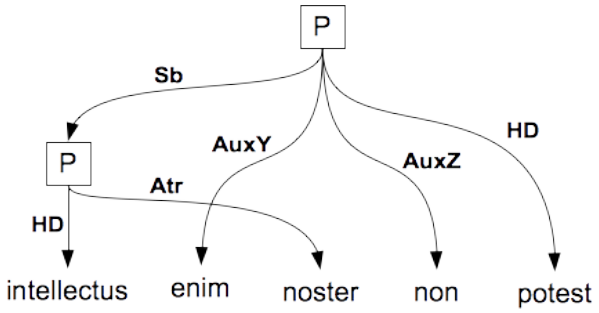

Figure 5: The same Latin phrase as Figure 1 restructured to match the TIGERSearch approach to tree data (similar for Figure 2). For every word with dependents, a dummy constituent (labelled *P* in this example) is inserted in its place, with the word reinserted as its head (marked with the edge label *HD*).

4. Tools for non-projective dependency treebanks

There are some tools that explicitly support non-projective dependency grammars.

ANNIS2 (Chiarcos et al., 2009) supports visualization and querying for dependency and constituency treebanks, as well as other annotated language data, via a web interface. However, users need to download and install ANNIS2 on their own servers, and convert their treebank data to its preferred format before using it.

The TrEd editor and its associated query language PML Tree Query, developed in the context of the Prague Dependency Treebank, supports dependency formalisms as well as other kinds of treebanks (Pajas and Štěpánek, 2009). PML Tree Query supports treebanks stored in formats compatible with the Prague Markup Language (PML), providing both textual and graphical representation of the queries.

TrEd can also be used as a web-based application. However, the graphical query interface is only available as a plug-in to the desktop version of TrEd, while the web interface only supports textual queries.

INESS (Rosén et al., 2012) is a web-based application that fully supports dependency treebanks both for visualization and querying, as well allowing users to visualize parallel treebanks, something that is currently not a feature of TüNDRA.

5. TüNDRA as a web application

TüNDRA is a web application developed as part of the CLARIN-D project[4] and accessible using CLARIN's single sign-on infrastructure. Academic users at participating universities are able to log into and use TüNDRA[5] with their existing institutional login credentials, and other academic users can request accounts from CLARIN[6].

Offering treebanks access via a web application is an application of the *Software as a Service* paradigm (SIIA, 2001) and has a number of advantages over traditional desktop software:

- A web application is accessible from any compatible browser, on any computer with an adequate Internet connection. There is no particular technical expertise required.
- New features, changes, and bug fixes are available to users as soon as they are implemented. Tools made available as web applications can be in a constant state of improvement without interfering with users and their activities.
- Treebank size and access speeds are not bounded by the limited memory and storage of desktop computers.
- Users do not have to import treebanks from various treebank data formats, and problems of software compatibility should never arise for treebanks available via TüNDRA, since developers must resolve any incompatibility before making the treebank publicly available.

This last point in particular distinguishes TüNDRA from other treebank software projects. TüNDRA is not just a service for the users of treebanks, but also for treebank developers. Introducing a new treebank into TüNDRA involves CLARIN-D taking charge of the treebank and making sure it is correctly processed and displayed, regardless of its formal assumptions or encoding format.

The *Software as a Service* paradigm has come under criticism, however, for the way that it takes control over data and its presentation away from users and data owners[7]. TüNDRA's service model is unlike that of stand-alone software: it is primarily a digital publishing platform for treebanks, and users do not have the autonomy to install any treebank they like and use it with TüNDRA. The user experience is more tightly controlled by the service provider, who bears the burden of responding to user interests and needs.

TüNDRA is integrated with CLARIN-Ds WebLicht language tool chaining environment (Hinrichs et al., 2010), enabling users to create treebanks from texts using a variety of automated parsing tools. Integration with WebLicht provides users with a limited ability to upload small to moderate sized treebanks in WebLicht's internal XML data format[8] and use them immediately in TüNDRA.

Web applications also pose digital rights management problems. While the IT-TB is available under a very liberal Creative Commons license, other widely used treebanks with very restrictive licensing provisions may be completely incompatible with a web publishing model. TüNDRA's integration with the CLARIN-D user authentication infrastructure makes it possible to restrict access to treebanks to authorized users, but installing into TüNDRA a corpus with a restrictive license still requires special permission from treebank providers. Site-licensed treebanks are only compatible with locally operated software.

However, the largest single benefit for web application development is the ability to introduce entirely new areas of functionality into TüNDRA without having to distribute software or interfere with the existing user base. TüNDRA required new functionality to support dependency treebanks, a development motivated by a request to support the IT-TB. This new functionality was rolled out simultaneously with the IT-TB, as soon as it was ready for users.

6. TüNDRA functionality

TüNDRA allows users to view each sentence in an installed treebank, and to move from one sentence to the next or jump to a sentence by its number in the treebank. Figure 6 shows a sentence from the IT-TB as displayed in TüNDRA: *sed forma est creatura* ("but form is a creature[9]"). The on-screen controls provide display and navigation functions: moving to the next or previous tree, jumping to other parts of the treebank, zooming, shrinking and panning the tree display, as well as downloading rendered trees in SVG, PNG and JPEG formats for reuse in other contexts.

Multiple visualization formats are available, providing different perspectives on individual trees and corresponding to different traditions in tree drawing. This is in contrast to ANNIS and INESS, which also provide browser visualizations of treebanks, but do not give users a choice of visualization style. TrEd allows users to define visualizations using stylesheet specifications described in its documentation.

Searching in treebanks is available by typing a query into the input box on the main TüNDRA page, at the top left of Figure 6. TüNDRA uses a query language adapted from the widely used TIGERSearch software (Lezius, 2002). TIGERSearch query language separates searching on sequential relationships in treebanks from hierarchical relationships, allowing queries that contain both. It is therefore well-suited to non-projective treebanks and has been adapted for INESS and ANNIS as well. Query results are displayed by highlighting the treebank elements that match the query. Users can browse within query results.

TüNDRA also provides consolidated statistics for query results, a feature not currently available in INESS or ANNIS. These consolidated results are not customizable, as

[4] `http://de.clarin.eu/`.

[5] Accessible with CLARIN login at: `https://weblicht.sfs.uni-tuebingen.de/Tundra/`

[6] `https://user.clarin.eu/user`.

[7] See Stallman (2010) for one well-known example.

[8] `http://weblicht.sfs.uni-tuebingen.de/weblichtwiki/index.php/The_TCF_Format`.

[9] *Scriptum super Sententiis*, Liber 1, Distinctio 8, Quaestio 5, Articulus 1, Argumentum 1.

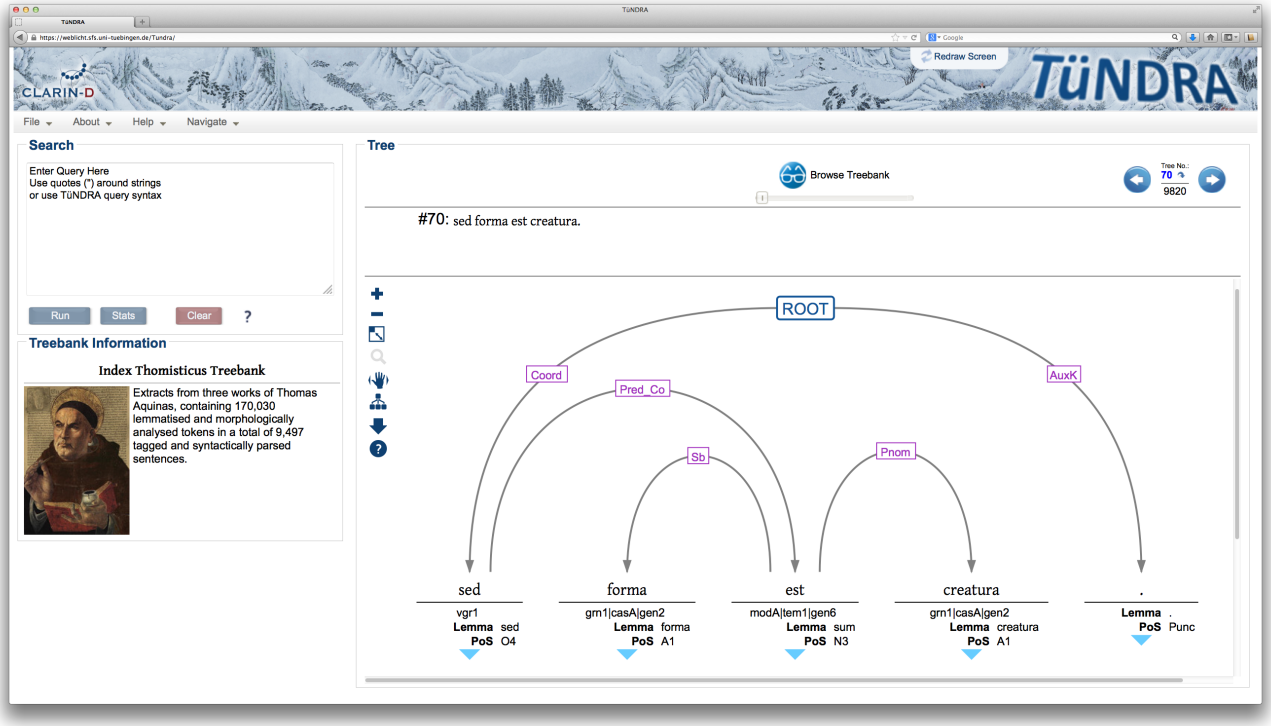

Figure 6: A short sentence from the IT-TB displayed in TüNDRA, using a Word Grammar style visualization.

they are in TrEd, but users can download the raw result data for further analysis in spreadsheets or other software of their own.

For more detailed information about the current functioning of TüNDRA, see Martens (2013) and the TüNDRA website.

7. Adapting TüNDRA to Dependency Treebanks

Adding support for the IT-TB involved interventions in two areas:

- Tree visualisations that are natural for dependency treebanks.
- Adapting TüNDRA's existing query engine to dependency grammars.

In both cases, the existing software and framework were extended to provide for the IT-TB. TüNDRA uses a number of standard protocols and off-the-shelf development libraries to simplify the development process.

Treebanks are stored in an extensible XML format devised for TüNDRA, and managed using an adapted version of the open source BaseX XML database software[10]. Dependency treebanks are encoded using a schema developed specifically for them.

All tree visualizations use W3C standard *Scalable Vector Graphics* (SVG), which are supported by all major internet browsers. Adding new visualizations for dependency

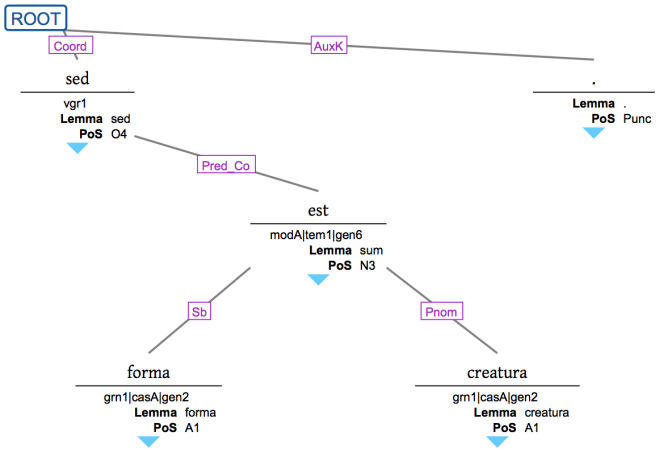

Figure 7: The same sentence as Figure 6, but using a TrEd-style rendering.

treebanks was primarily a matter of developing new routines for transforming TüNDRA's XML tree format into the desired SVG form.

Since the TIGERSearch query language already supports non-projective trees, adapting the query language itself to dependency treebanks was not very complicated.

Internally, on the TüNDRA server, queries are translated into XQuery, a W3C standard for interacting with XML documents. XQuery is supported in BaseX, which executes the translated query against the stored treebank in XML form.

TüNDRA offers full Unicode support through the use of treebank-specific webfonts, another W3C standard technol-

[10] http://basex.org.

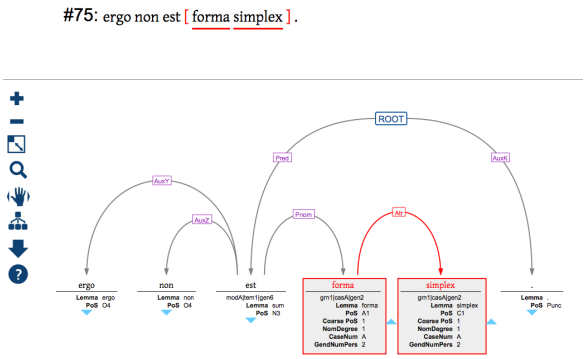

Figure 8: A query match for ”forma simplex”.

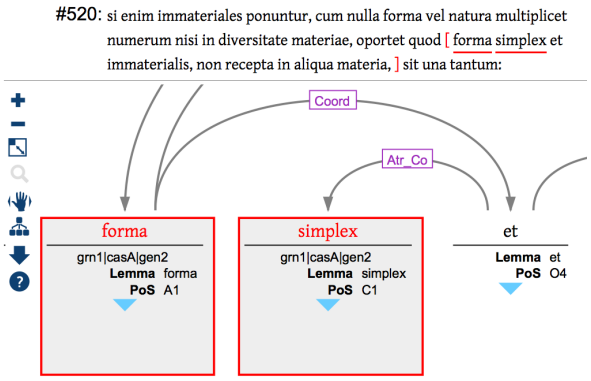

Figure 9: A query match where sequence and hierarchy do not match.

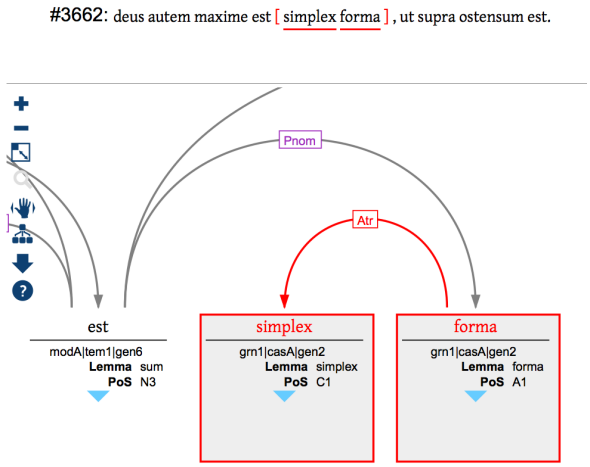

Figure 10: Dependency where word order is reversed.

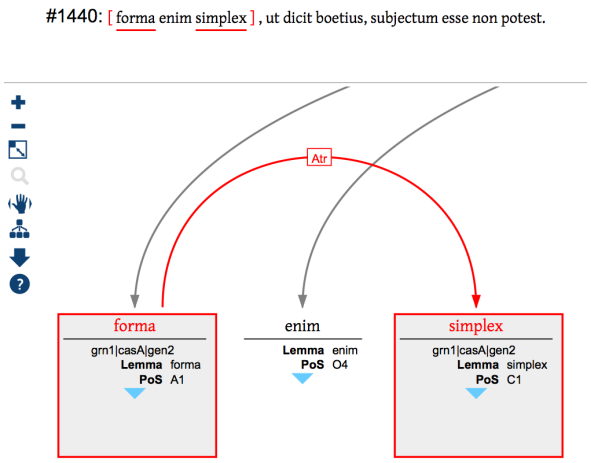

Figure 11: Dependency with an intervening word.

ogy. Even when a user's desktop system is not configured to support a particular language, TüNDRA provides the user with a font to correctly display texts.

7.1. Dependency tree visualization

TüNDRA provides two visualization styles for dependency trees. The first, shown in Figure 6, draws on the visualization style most associated with Word Grammar (Hudson, 1984). Words are ordered linearly, like on a page, and dependencies are represented as arcs above the words. Figure 7 shows an alternative visualization, based on the approach used in TrEd. The nodes of the tree are arranged horizontally according to surface word order, but their vertical placement reflects their place in the dependency hierarchy. In both visualizations, lemmas and parts-of-speech are shown to users while the full set of morphological features is obscured to prevent screen clutter, but can be optionally viewed by clicking the blue arrow beneath each token.

7.2. Searching dependency treebanks

TüNDRA's adapted TIGERSearch query language already supports the separation of hierarchical and sequential relations necessary to query dependency trees, and has been adapted to support the requirements of the IT-TB and similar treebanks. The query language does not assume, as it would for a constituency treebank, that all tokens are terminal nodes in the tree.

To search for the token sequence *forma simplex* (“simple form”) in the IT-TB, enter the following query into the query window (upper left of Figure 6):

```
[token="forma"] . [token="simplex"]
```

This matches any node in the treebank with a `token` attribute matching the string *forma* when followed in surface word order (indicated by “.”) by a node with a `token` attribute matching the string *simplex*. The result is a tree with the relevant elements highlighted in red, as in Figure 8.

This query will only match instances of *forma simplex* and will exclude any instance of *simplex forma*, *formae simplices*, or any other ordering or inflection. It will, however, match all instances where the words *forma simplex* happen to be adjacent, even when there is no direct syntactic dependency between them, as in Figure 9[11].

To query for any appearance of *forma* followed by

[11] In Figure 9, the word *simplex* does not depend directly on *forma* because the former is part of a coordinated construction (*simplex et immaterialis*, “simple and immaterial”). According to

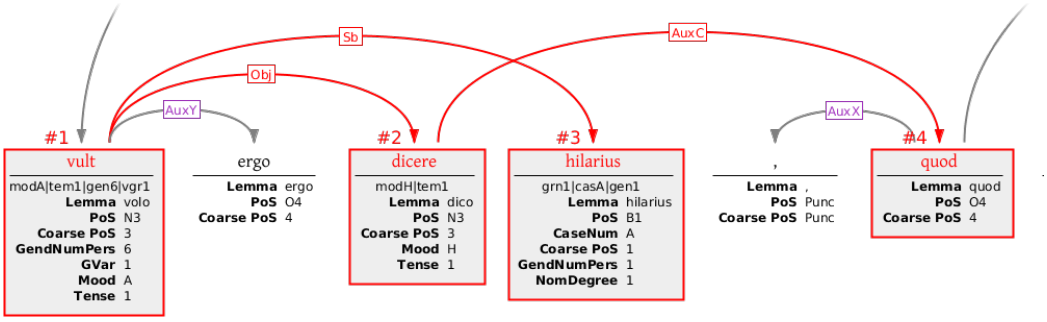

Figure 12: A right-headed well-nested subtree from the IT-TB: *vult ergo dicere hilarius, quod...* ("hence, Hilarius wishes to say that..."), from *Scriptum super Sententiis*, Liber 1, Distinctio 5, Quaestio 3, Articulus 1, Expositio.

simplex regardless of declension, users can query for lemmas directly:

```
[lemma="forma"] .  [lemma="simplex"]
```

Users can query for any combination of node features by joining them inside square brackets with AND (`&`) and OR (`|`) operators.

To query for any instance of *forma* with the word *simplex* as a direct dependent, use the ">" operator:

```
[token="forma"] > [token="simplex"]
```

This query will find *forma* and *simplex* only where joined by a hierarchical dependency, without regard for surface word order or proximity in the sentence, matching trees like the ones in Figures 10 and 11.

By separating hierarchy and surface order entirely, it is also possible to query for trees that satisfy particular hierarchical structures and orders at the same time. To find trees where any declension of *simplex* is dependent on any declension of *forma*, but where *simplex* immediately precedes *forma*, variable labels (indicated by the `#` sign) must be introduced into the query:

```
#1:[lemma="forma"] > #2:[lemma="simplex"]
& #2 .  #1
```

Queries may specify any number of nodes and variable labels in any relation. Relations can query over hierarchical or sequential relations at any distance. Unicode regular expression matching is supported on all feature labels, and a node can have any number of features expressible as Unicode strings.

7.3. Identifying well-nested structures

Since handling non-projectivity is such an essential practical element of dependency treebank tool development, it is important to highlight the ability to perform searches that specifically target this phenomenon. For example, the notion of *well-nestedness* is a relaxed notion of syntactic projectivity relevant to languages with relatively free word order. A tree or subtree[12] is said to be well-nested if, for any pair of edges that cross, the parent node of one of the two edges dominates the parent of the other[13].

The following query searches for non-projective, right-headed, well-nested subtrees in the IT-TB using TüNDRA:

```
#1 .* #2 .* #3 .* #4
   & #1 > #3 & #2 > #4
   & #1 >* #2
```

Figure 12 is an example of a match for this query as displayed in TüNDRA. Note that TüNDRA displays which tokens match which variable names in order to simplify visual inspection and interpretation. The edge from #2 (*dicere*) to #4 (*quod*) crosses the edge from #1 (*vult*) to #3 (*hilarius*), making the subtree rooted in #1 (*vult*) clearly non-projective. It is nonetheless *well-nested* because the root of one of the two edges (#1) dominates the root of the other (#2), and it is *right-headed* because the rightmost of the two roots (#2) is dominated by the leftmost (#1).

This query identifies exactly such subtrees because:

1. The term `#1 .* #2 .* #3 .* #4` searches for four distinct nodes in a sequence that may or may not include intervening nodes.
2. The term `#1 > #3 & #2 > #4` restricts the search to instances where the first node directly dominates the third and the second node directly dominates the fourth, ensuring that their edges cross non-projectively.
3. The term `#1 >* #2` further restricts the search to instances where the first node dominates the second, although possibly not directly.

This kind of search is interesting only for dependency treebanks and highlights the importance of tools adapted specifically for them. The ability to separately query node sequence and node hierarchy, and then join them in one query, makes this possible.

the annotation style of the IT-TB, there is no direct dependency in this context, but other approaches differ.

[12] A *subtree* is defined as the entire set of nodes and edges descending from some specific node, i.e. a *bottom-up subtree* in the terminology of Luccio et al. (2001).

[13] This definition follows Dubusmann & Kuhlmann (2010) and Havelka (2007), but note the slightly different definition in Bodirsky et al. (2005).

8. Conclusion

TüNDRA offers a web application solution to visualizing and searching in treebanks, including dependency treebanks with complex morphologies and non-projective tree structures. Integrating the IT-TB into TüNDRA makes it much more accessible to researchers, and adapting TüNDRA to the IT-TB has been very productive in expanding its functionality for future treebank projects involving dependency formalisms and classical languages. This adaptation makes TüNDRA an accessible platform for other treebanks with problems similar to those of the IT-TB: non-projectivity, free word order and complex morphology. Although unnecessary for support of the IT-TB, TüNDRA's use of server-provided fonts also makes it a solution for treebanks with writing schemes that may be difficult to display using other solutions, like many historical treebanks and resources for less widely supported languages.

A key element of dependency treebank support is the ability to encode, display and query non-projective trees in a transparent way. TüNDRA supports two approaches to displaying these kinds of trees, based on already existing traditions in drawing them, and queries them by separating hierarchical relations from sequential ones. As the example of *well-nestedness* in Section 7.3. shows, this ability makes it possible to identify structures where hierarchical and sequential information interact in productive ways.

9. References

D. Bamman, M. Passarotti, G. Crane, and Raynaud S. 2007. *Guidelines for the Syntactic Annotation of Latin Treebanks*. Tufts University Digital Library. URL: http://hdl.handle.net/10427/42683.

M. Bodirsky, M. Kuhlmann, and M. Möhl. 2005. Well-nested drawings as models of syntactic structure. In *Proceedings of the 10th Conference on Formal Grammar and 9th Meeting on Mathematics of Language*, pages 195–203, Stanford, CA, USA. CSLI Publications.

R. Busa. 1980. *Index Thomisticus*. Frommann-Holzboog, Stuttgart.

C. Chiarcos, S. Dipper, M. Götze, U. Leser, A. Lüdeling, J. Ritz, and M. Stede. 2009. A flexible framework for integrating annotations from different tools and tagsets. *Traitement Automatique des Langues*, 49(2).

N. Chomsky. 1965. *Aspects of the theory of syntax*. MIT Press.

O. Christ and B. Schulze, 1995. *Ein flexibles und modulares Anfragesystem für Textcorpora*, pages 121–133. Niemeyer, Tübingen.

R. Debusmann and M. Kuhlmann, 2010. *Dependency grammar: Classification and exploration*, pages 365–388. Springer, Berlin and Heidelberg.

J. Hajič, J. Panevová, E. Buráňová, Z. Urešová, and A. Bémová. 1999. *Annotations at Analytical Level - Instructions for annotators*. Charles University, Institute of Formal and Applied Linguistics. URL: http://ufal.mff. cuni. cz/pdt2.0/doc/manuals/en/a-layer/pdf/a-man-en.pdf.

J. Havelka. 2007. Beyond projectivity: Multilingual evaluation of constraints and measures on nonprojective structures. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 608–615, Prague.

E. Hinrichs, M. Hinrichs, and T. Zastrow. 2010. Weblicht: web-based lrt services for german. In *Proceedings of the ACL 2010 System Demonstrations (ACLDemos Í0)*, pages 25–29, Stroudsburg, PA, USA.

R. Hudson. 1984. *Word grammar*. Blackwell, Oxford.

W. Lezius. 2002. Tigersearch - ein suchwerkzeug für baumbanken. In *Proceedings der 6. Konferenz zur Verarbeitung natürlicher Sprache (KONVENS 2002)*, Saarbrücken.

F. Luccio, A. Enriquez, P. Rieumont, and L. Pagli. 2001. Exact rooted subtree matching in sublinear time. Technical Report TR-01-14, Università di Pisa.

F. Mambrini and M. Passarotti. 2013. Non-projectivity in the ancient greek dependency treebank. In *Proceedings of the Second International Conference on Dependency Linguistics (DepLing 2013)*, pages 177–186, Prague.

S. Marcus. 1965. Sur la notion de projectivité. *Mathematical Logic Quarterly*, 11(2):181–192.

S. Martens. 2012. Tündra: Tigersearch-style treebank querying as an xquery-based web service. In *Proceedings of the joint CLARIN-D/DARIAH Workshop 'Service-oriented Architectures (SOAs) for the Humanities: Solutions and Impacts', Digital Humanities 2012*, Hamburg.

S. Martens. 2013. Tündra: A web application for treebank search and visualization. In *Proceedings of The Twelfth Workshop on Treebanks and Linguistic Theories (TLT12)*, Sofia.

B. McGillivray, M. Passarotti, and P. Ruffolo. 2009. The index thomisticus treebank project: Annotation, parsing and valency lexicon. *Traitement Automatique des Langues*, 50(2).

P. Pajas and J. Štěpánek. 2009. System for querying syntactically annotated corpora. In *Proceedings of the ACL-IJCNLP 2009 Software Demonstrations*, Singapore.

M. Passarotti and P. Ruffolo. 2010. Parsing the index thomisticus treebank – some preliminary results. In *Latin Linguistics Today. Akten des 15. Internationalen Kolloquiums zur Lateinischen Linguistik*, pages 714–725, Innsbruck. Innsbrucker Beiträge zur Sprachwissenschaft.

D. Rohde. 2005. *TGrep2 User Manual v. 1.15*. MIT. URL: http://tedlab.mit.edu/ dr/Tgrep2/tgrep2.pdf.

V. Rosén, K. De Smedt, P. Meurer, and Helge Dyvik. 2012. An open infrastructure for advanced treebanking. In *Proceedings of the META-RESEARCH Workshop on Advanced Treebanking at LREC 2012*, pages 22–29, Istanbul.

SIIA. 2001. Software as a service: Strategic backgrounder. Technical report, Software & Information Industry Association, Washington, DC. URL: http://www.siia.net/estore/pubs/SSB-01.pdf.

R. Stallman. 2010. What does that server really serve? *Boston Review*, March. Online review. URL: http://www.bostonreview.net/richard-stallman-free-software-DRM.