A Compact Interactive Visualization of Dependency Treebank Query Results

Chris Culy[†], Marco Passarotti[‡], Ulla König-Cardanobile[†]
[†]Universität Tübingen, [‡]Università Cattolica del Sacro Cuore
[†]Seminar für Sprachwissenschaft, Wilhelmstraße 19, Tübingen, Germany
[‡]CIRCSE Research Centre, Largo Gemelli 1, Milan, Italy
E-mail: christopher.culy@uni-tuebingen.de, marco.passarotti@unicatt.it, ulla.koenig@gmail.com

Abstract

One of the challenges of corpus querying is making sense of the results of a query, especially when a large number of results and linguistically annotated data are concerned. While the most widespread tools for querying syntactically annotated corpora tend to focus on single occurrences, one aspect that is not fully exploited yet in this area is that language is a complex system whose units are connected to each other at both microscopic (the single occurrences) and macroscopic level (the whole system itself). Assuming that language is a system, we describe a tool (using the DoubleTreeJS visualization) to visualize the results of querying dependency treebanks by forming a node from a single item type, and building a network in which the heads and the dependents of the central node are respectively the left and the right vertices of the tree, which are connected to the central node by dependency relations. One case study is presented, consisting in the exploitation of DoubleTreeJS for supporting one assumption in theoretical linguistics with evidence provided by the data of a dependency treebank of Medieval Latin.

Keywords: corpus query, dependency treebanks, visualization of results

1. Introduction and Motivation

One of the challenges of corpus querying is making sense of the results of a query, especially when a large number of results is concerned. Hard enough as this task is for text results, even without annotations, it is yet more difficult when results are syntactic structures, such as dependency constructions extracted from a treebank. Typically, results show the full dependency annotation of sentences, either one at a time or perhaps in a "carousel" of images, e.g. TrEd (Pajas & Štěpánek, 2008), Annis (Zeldes et al., 2009) and Tündra (Martens, 2012).

These approaches do not facilitate the comparison of the results, and especially of the specific context that was searched for. Indeed, one limit of the most widespread tools for querying syntactically annotated corpora is that they tend to focus on single occurrences (usually shown by tree-graphs), at best providing statistics reported in the form of lists and frequencies (or percentages). One aspect that is not fully exploited yet in this area is that language is a system whose units are connected to each other at both microscopic (the single occurrences) and macroscopic level (the whole system itself). In this respect, language can be viewed as a complex system by looking at different levels of analysis: morphology (in particular, word formation), syntax (relations between words in sentences) and semantics (for instance, compositional semantics).

Assuming that language is a system and focussing on syntactic analysis, a simple and efficient way to visualize the results of a query performed on a dependency-based treebank is to form a node from a single item type, and build a network in which the heads and the dependents of the central node are respectively the left and the right vertices of the tree, which are connected to the central node by dependency relations.

2. Contribution

In the domain of text queries, Word Tree (Wattenberg & Viègas, 2008) constructs a suffix tree (or a prefix tree, depending on user selection) from the results, and then provides an interactive way to explore that tree by expanding and collapsing nodes in the tree. In this way, the user can see and explore the various extended contexts of the hit term in a much more compact way than a KWIC index or concordance.

Although Wattenberg & Viègas (2008) refer to Word Tree as an "Interactive Concordance", one thing that concordances typically provide which Word Tree does not is two sided context. In a concordance the hit term is typically displayed with some context to the left of it and some context to the right of it. In contrast, Word Tree displays either the right context (suffix tree) or the left context (prefix tree), but not both. To address this issue, Culy & Lyding (2010) created Double Tree, an extension of the Word Tree concept to two sides. Since it is not the case that every right hand side corresponds to every left hand side (and vice versa), there needs to be some way to indicate which nodes on the right correspond to which nodes on the left and conversely. In Double Tree, when the user clicks on a node, the nodes on the opposite side of the tree with which it occurs are highlighted. Thus, in Figure 1, we can say that *and see whether* is a phrase in the original text, but *and see him* is not.

The key idea of our contribution is to realize that we can use Double Tree to visualize the results of querying syntactic corpora (in particular, dependency treebanks) by using the traversals of the syntactic structures as the data to be visualized, rather than the syntactic structures themselves. In other words, what we visualize are *paths* through the structures, which are in fact what the user is typically interested in.

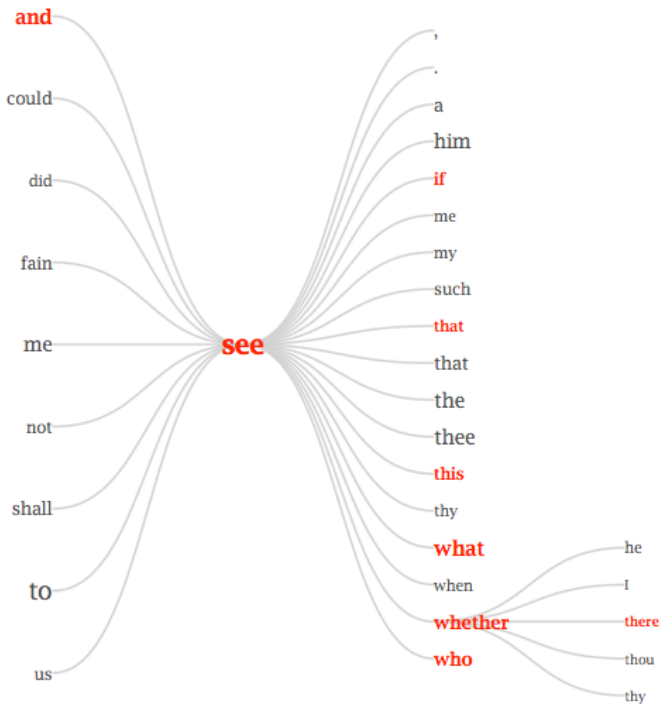

Figure 1: Double Tree with text

Figure 2: Double Tree with dependencies

KWIC

Click on a column heading to sort it, or reverse its sort.

Sort by DepRel Sort by Token Sort by Lemma Sort by POS

< > Showing results 1 to 6 of 6 results

-5	-4	-3	-2	-1	HIT	+1	+2	+3	+4	+5	
				root	**Obj**	Atr	AuxX				005.SCG*LB3.CP-9++7.N.-3.19-7.26-3: **assimilat definitiones significantur ,**
				root	**Obj**	Atr	AuxX				005.SCG*LB3.CP-9++7.N.-3.19-7.26-3: **assimilat definitiones significantur ,**
				root	**Obj**	Atr	Coord	Sb_Co			005.SCG*LB3.CP-9++7.N.-3.19-7.26-3: **assimilat definitiones significantur et formae**
				root	**Obj**	Atr	Coord	Sb_Co	Atr		005.SCG*LB3.CP-9++7.N.-3.19-7.26-3: **assimilat definitiones significantur et naturae rerum**
				root	**Obj**	Atr	AuxP	Adv			005.SCG*LB3.CP-9++7.N.-3.19-7.26-3: **assimilat definitiones significantur per quas**
root	Pred_Co	AuxP	Adv	Atr	**Obj**	Atr					005.SCG*LB1.CP-1++4.N.-2.17-4.20-3: **et collocamus in rebus cognoscimus definitiones quarum**

Figure 3: KWIC view with dependencies

For example, in Figure 2, we see the results of a query performed on a Medieval Latin dependency treebank (the *Index Thomisticus* Treebank, around 200,000 words in more than 11,000 sentences (Passarotti, 2011)). The query searches for those occurrences of the word *definitiones* ("definitions") where this is labeled with the dependency relation Obj (object: either direct or indirect). This means that in these contexts the word *definitiones* stands in an object relation with its heads in the tree.

Figure 2 shows that *definitiones* depends as an object on two different words in the treebank, which are reported as left-side vertices of the Double Tree: *assimilat* ("it absorbs") and *cognoscimus* ("we know")[1]. The size of the words in the vertices depends on the number of their occurrences in that specific syntactic position: the node of *assimilat* is bigger than that of *cognoscimus*, because *assimilat* occurs 5 times while *cognoscimus* just one (this is shown by moving the pointer on the word in the tree).

The right-side vertices of the Double Tree report the words that depend on the word *definitiones* (labeled with Obj) regardless of its head in the tree. These are *significantur* ("they are meant"), which occurs 5 times, and *quarum* ("of them", feminine), whose frequency is 1.

By clicking on one word, the words reported in the double tree that share at least one context with that word are highlighted in red. In figure 2, the words *assimilat*, *definitiones* and *significantur* are highlighted because *definitiones* heads the word *significantur* in all the 5 contexts where *definitiones* depends on *assimilat* as an object.

The Double Tree can be further expanded by clicking on one its vertices: the rightmost side of Figure 2 shows the nodes that depend on *significantur* along the dependency path *assimilat* - *definitiones* - *significantur*.

3. Technical Details

We have reimplemented Double Tree in Javascript using the D3 toolkit (Bostock et al., 2011). DoubleTreeJS[2] extends the original Double Tree with some capabilities that were mentioned as future work in Culy & Lyding (2010), such as the abilities to sort the branches by a variety of properties, to filter the branches by different properties, and to search the items in the Double Tree. Another particularly useful addition is the ability to trigger additional actions when using a modifier key while clicking on a node. In particular, we use *shift-click* to show a more extensive view of the item clicked on in the current context, using a KWIC style view of the original results. Since these results are of the clicked item in the context of the original hit, the number of results is smaller than the full results and they are easier to read as a whole. Figure 3 shows the KWIC view with dependencies resulting from shift-clicking on *Obj/definitiones* in Figure 2. From the KWIC view, the user can click on one example and see the dependency structure for the whole sentence, either as an arc diagram or as a hierarchical diagram, according to the settings chosen.

DoubleTreeJS is designed as a library which is independent from any particular data or data format. It is up to an application to supply DoubleTreeJS with the data to visualize. DoubleTreeJS then constructs its own internal data structures and the visualization. However, DoubleTreeJS also comes with several practical examples which use a data model which allows text to be read, provides simple regular expression-based querying, and allows query results to be stripped of context items that are not of interest (e.g., omitting function words, or showing nominal elements only).

To use DoubleTreeJS for dependency structures, we constructed a new example using a slightly modified version of the above data model. Basically, we simply have to provide DoubleTreeJS with the hits in the structure, along with the left contexts of the paths from the roots to the hits and the right contexts of the paths from the hits to the leaves, which we do as follows.

First, apart from the data model, we construct the list of all the paths from the roots to the leaves through the structures in the treebank. Then, we use a modified version of the data model to read in the paths. The modification, also relevant in other uses of the data model, limits the context of an item in a path to the path itself: unlike raw text, where we allow context to cross sentence boundaries, we do not want the context of a syntactic node to include nodes of words occurring in other sentences. All of the other operations of the tool, including the querying, the stripping of the results, the interaction in the visualization, follow immediately from the data model and from DoubleTreeJS itself. The only additional work was to supply the KWIC view (adapted from other examples), and the thumbnail view, which is specific to syntactic structures. The thumbnails use the ProD visualization (Culy et al., 2012)[3].

The data model can take advantage of whatever information is present in the input. In particular, it permits querying, via regular expressions, over whatever fields are present in the input, including forms, lemmas, parts of speech, etc., in addition to the dependencies. Figure 4 shows a schematic diagram of a sample application using DoubleTreeJS.

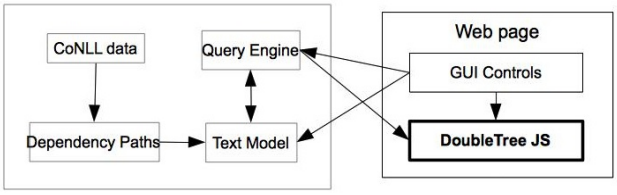

Figure 4: System diagram

[1] In the Double Tree reported in Figure 2, each word is preceded by the name of its dependency relation and followed by a part-of-speech tag (according to the *Index Thomisticus* Treebank annotation style). The dependency relations reported in Figure 2 are: Atr (attribute), AuxP (preposition), AuxX (punctuation), Coord (coordination), Obj (direct or indirect object). The part-of-speech tags are: 1 (nominal inflection), 3 (verbal inflection), 4 (uninflected), Punc (punctuation mark).

[2] Freely available at http://www.sfs.uni-tuebingen.de/~cculy/software/DoubleTreeJS/index.html.

[3] Freely available at http://www.sfs.uni-tuebingen.de/~cculy/software/ProD/index.html.

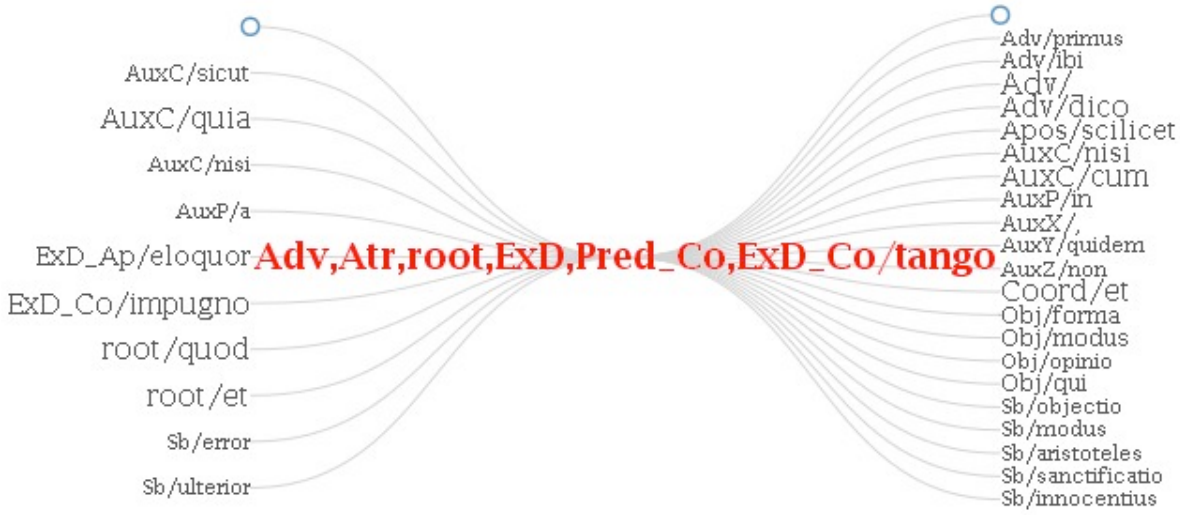

Figure 5: The Double Tree of *tango*

4. Case Study

In this section, we describe a case study consisting in the exploitation of DoubleTreeJS for supporting one assumption in theoretical linguistics with evidence provided by corpus data. The purpose of this case study is not to provide a full analysis of data, but to give an example of how DoubleTreeJS can be used for accessing a corpus for linguistic research.

It is a well-known fact that verbs play a central role in dependency-based description of the syntactic structure of sentences. The root of a dependency tree is the predicate of the main clause of the sentence (usually a verb) and the remainder of the sentence depends on the root. Also, this aspect is strictly related to the basic idea of frame semantics (Fillmore, 1982): knowledge of a particular sense of a verb (and of other parts of speech, as well) requires the knowledge of a number of entities evoked in the situation designated by that verb. These entities are both obligatory and optional complements, respectively named 'arguments' and 'adjuncts'[4].

The entities are collected in a 'frame'. Knowledge of the specific sense of a verb thus involves the knowledge of the specific frame of that verb, i.e. of the entities that are combined with that verb. "The capacity a verb (or noun, etc.) has for combining with particular patterns of other sentence constituents" (Allerton, 1982: 2) is called the 'valency' of that verb. Valency is usually defined as the number of arguments of a word, but Éech et al. (2010) have recently brought good arguments for not distinguishing between obligatory and not obligatory complements, claiming that all complements are ruled by a similar mechanism of selection, called by the authors 'full valency'.

Although valency can refer to different parts of speech (verbs, nouns, adjectives and, at some extent, adverbs), scholars have mainly focused their attention on verbs. This is due to the fact that verb is the most valency-capable part of speech.

In a dependency tree, the node of a complement depends on the node of the word that it modifies. As verbs are the part of speech most able to select complements, it may be expected that in a dependency-based treebank they have a high number of dependent nodes and, conversely, a small number of head nodes. Instead, other parts of speech should behave differently. In particular, we expect that (a) nouns should have a similar number of heads and dependents and (b) adjectives and adverbs should have more heads than dependents.

DoubleTreeJS can be used to confirm (or refute) these intuition-based assumptions. In terms of DoubleTreeJS, the above assumptions mean the following:

- a Double Tree centred on a verb should have more right branches (dependents) than left ones (heads);
- a Double Tree centred on a noun should have more or less the same number of right and left branches;
- a Double Tree centred on an adjective or an adverb should have more left branches.

In our case study, we took one lemma of the IT-TB as representative for each of the parts of speech concerned and built its corresponding Double Tree[5].

The first lemma discussed is the verb *tango* ("to touch"). Figure 5 presents its Double Tree[6]. Figure 5 clearly shows that the Double Tree centred on *tango* has much more right branches (dependent-nodes) than left ones (head-nodes). Most of the left nodes are verbs (*eloquor*, *impugno*: these are the cases where *tango* heads a subordinate clause) and "bridging" function words, like subordinative conjunctions (*sicut*, *quia*, *nisi*, *quod*)[7].

Figure 6 shows the Double Tree of the noun *diversitas* ("difference"). In Figure 6, the number of left and right branches is similar. In particular, it is worth noting that most of the left nodes are verbs (*sum*, *sequor*, *consequor*

[4] Different terminology is used in different traditions. For instance, 'actants' (arguments) and 'circonstants' (adjuncts) are the terms introduced by Tesnière (Tesnière, 1959), while 'inner participants' (arguments) and 'free modifications' (adjuncts) are those used in Functional Generative Description (Sgall et al., 1986).

[5] Of a number of lemmas extracted randomly from the IT-TB, the ones discussed here are among those whose Double Tree fits the size of the page in the paper.

[6] In the Double Trees reported here, branches are ordered alphabetically by dependency relation.

[7] In the IT-TB style, lemma "_" (appearing, for instance, in Adv/_) is assigned to abbreviations.

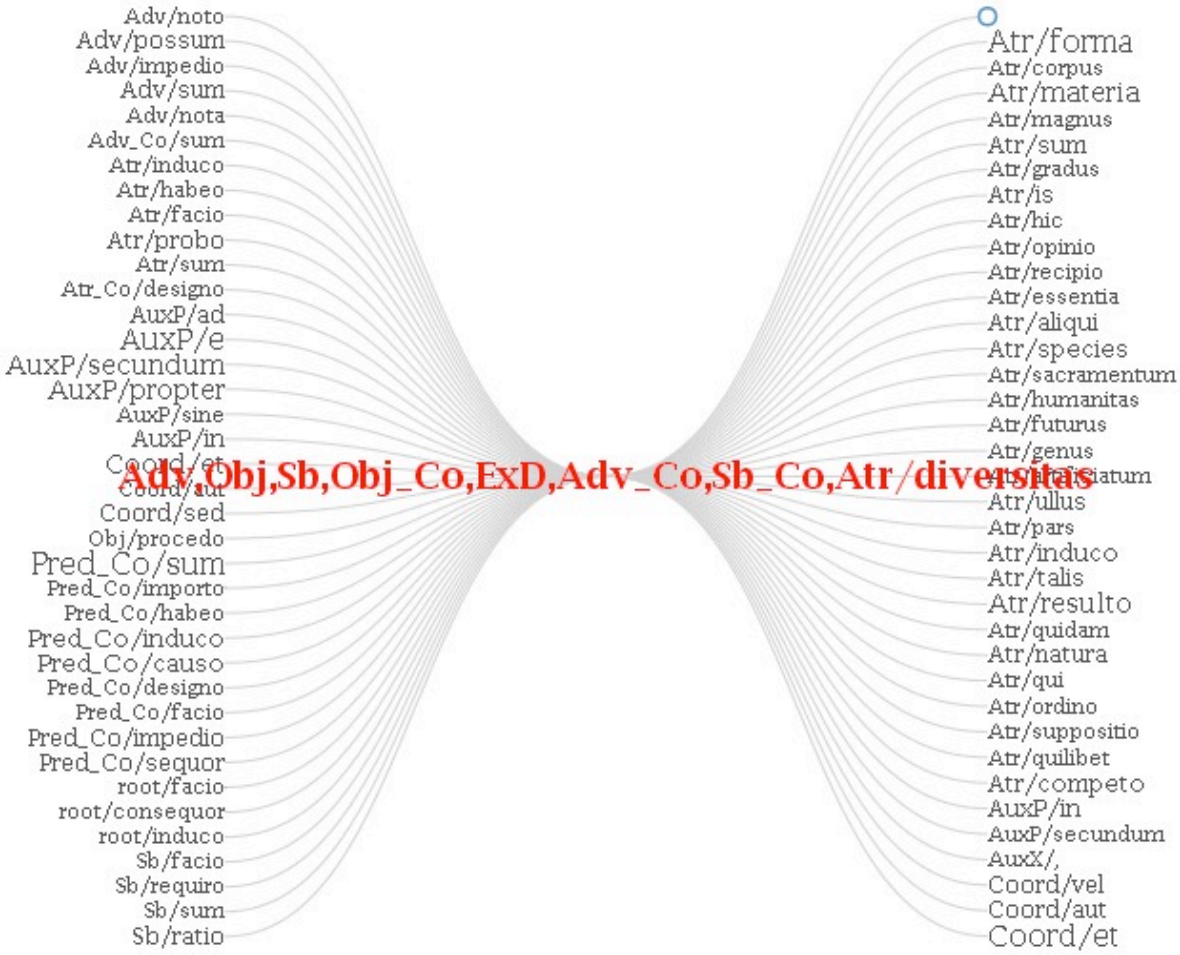

Figure 6: The Double Tree of *diversitas*

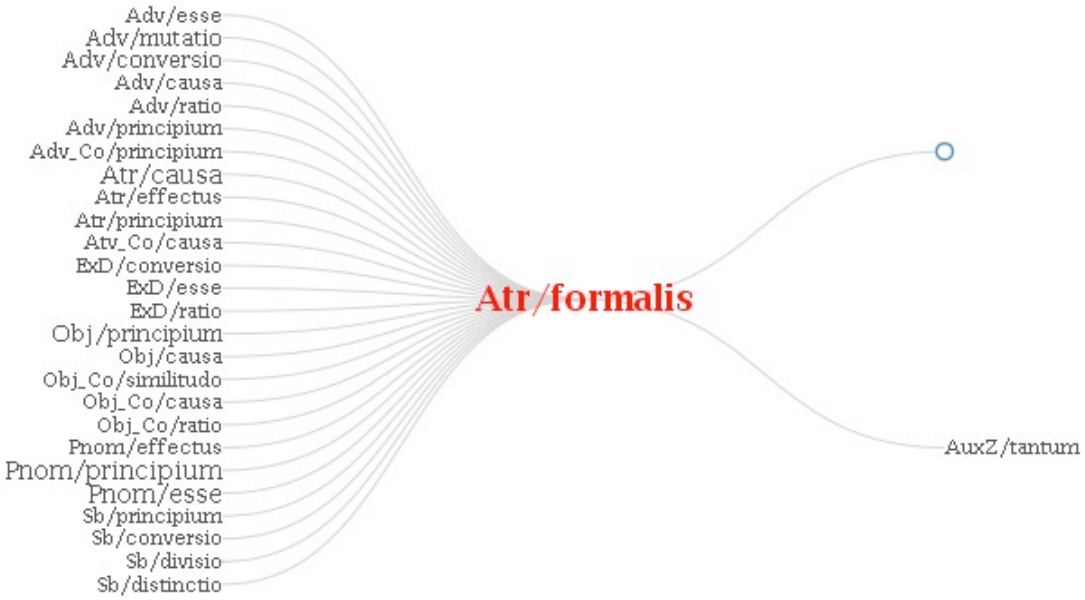

Figure 7: The Double Tree of *formalis*

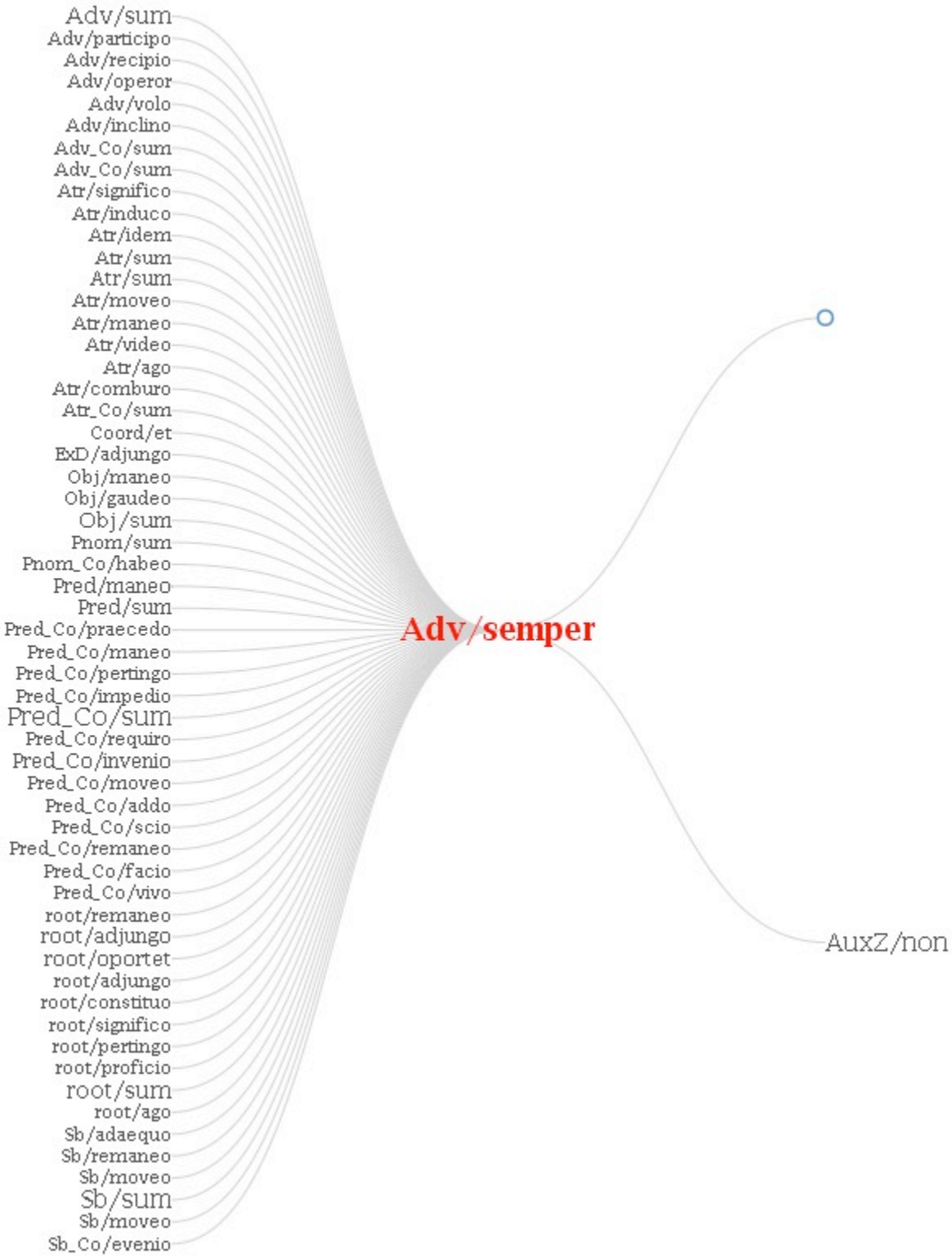

Figure 8: The Double Tree of *semper*

and all the other lemmas ending in *-o*) and most of the right ones are labelled with the dependency relation Atr (attributes). This is consistent with the fact that nouns tend to modify verbs and nouns (i.e. to be headed by verbs and nouns) and to be modified by attributes represented by nouns (mostly in genitive case), agreeing adjectives and relative clauses (headed by verbs in the annotation style of the IT-TB). That one noun is more dependent on verbs than on nouns is related to its semantic value.

Figure 7 reports the Double Tree of the adjective *formalis* ("formal")[8]. This Double Tree features almost only left branches. There is just one right node, which is filled with an adverb (*tantum*: "only")[9]. All the left nodes are nouns, reflecting that adjectives tend to modify (i.e. to be dependent on) nouns.

Figure 8 is the Double Tree of the adverb *semper* ("always")[10]. Like with the adjective *formalis*, the Double Tree of *semper* has almost only left branches (the only right node being that of the negation *non*). All the left nodes but two are verbs. Namely, they are:

(a) one occurrence of the pronoun *idem* ("same"; form: *eodem*), used with attributive function in the clause *semper eodem modo se habentes* ("[they] being always in the same way");

(b) one occurrence of the coordinating conjunction *et* ("and"). In this case, *semper* is an adverbial modification shared by two coordinated predicates and, thus, dependent on the node of the coordinating element (*et*).
The full sentence here concerned is *semper enim honorabilius est agens patiente, et principium, scilicet activum, [honorabilius est] materia* ("The agent is always more honorable than the patient, and the principle, i.e. the active one, [is more honorable] than the matter"). In the dependency tree of this sentence in the IT-TB, the conjunction *et* heads (i.e. coordinates) both the predicate of the first clause (*est*) and all the words of the second clause that would depend on the elliptical predicate of it, if this was present. *Semper* depends on *et* because it modifies both the present and the missing predicates. Further information on the IT-TB annotation style can be found in Bamman et al. (2007).

5. Conclusion and Future Work

We have presented an innovative interactive tool to help make sense of query results for syntactic structures by reusing an existing system for visualization with different data: paths through syntactic structures instead of strings of textual items. While we have illustrated dependency structures, also constituent structures, or indeed any graph that can be represented by traversals, could be visualized using this technique. The visualization itself needed no modification, and the modification to the data model was simple. In addition, all the scenarios illustrated by the DoubleTreeJS examples (e.g. comparison of two Double Trees) are easily done also for syntactic structures.

Although we have used a simple data model and a simple query engine, it would also be possible to use DoubleTreeJS with other data models and/or other search engines. DoubleTreeJS is just a visualization library, and as such it does not depend on any particular data model or search engine. In other words, every component of the system reported in Figure 4 could be replaced by a component with similar functionality. Indeed, we have used DoubleTreeJS to visualize the results of text queries in Sketch Engine (Kilgariff et al., 2004). This was done directly, i.e. without using the sample data model we have used here. In the near future, we hope to be able to use DoubleTreeJS to visualize the results produced by the XQuery-based TüNDRA treebank querying tool (Martens, 2012).

An anonymous referee suggested that DoubleTreeJS could also be helpful in "debugging" parses, i.e. checking for anomalies in input data. We fully agree, and think that DoubleTreeJS would be a useful part of a broader application that might also allow editing the underlying dependency data when an error is found, in addition to incorporating a more sophisticated query engine.

Finally, assuming that one of the main interests in querying treebanks is to compare different constructions as well as the syntactic behaviour of different (categories of) words, in the near future we foresee to enhance DoubleTreeJS with a visualization facility that permits to highlight the shared and non-shared features of two or more words/constructions in a dependency treebank.

6. Acknowledgements

The research of the first author was supported by the Federal Ministry for Education and Research (BMBF) as part of the grant CLARIN-D. Thanks to Markus Dickinson for valuable feedback.

7. References

Allerton, D.J. (1982). *Valency of the English verb*. London: Academic Press.

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

Bostock, M., Ogievetsky, V. and Heer, J. (2011). D3: Data-Driven Documents. *IEEE Trans. Visualization & Comp. Graphics* (Proc. InfoVis), 17(12), pp. 2301--2309.

Culy, C., Dima, C. and Dima, E. (2012). Through the Looking Glass: Two Approaches to Visualizing Linguistic Syntax Trees. In *Proceedings of the 16th International Conference on Information Visualisation IV2012*, *July 11-13, 2012, Montpellier, France*. pp. 214--219.

Culy, C. and Lyding, V. (2010). Double Tree: An

[8] For reasons of space, this Double Tree is limited to the occurrences of *formalis* labelled with the dependency relation Atr only.

[9] The other right node appearing in figure 8 is not filled with any lemma, as it represents those occurrences of *formalis* having no dependent nodes in the tree.

[10] Again for reasons of space, this Double Tree is limited to the occurrences of *semper* labelled with the dependency relation Adv (adverbial) only.

Advanced KWIC Visualization for Expert Users. In *Information Visualization, Proceedings of IV 2010, 2010 14th International Conference Information Visualization*. London, UK, pp. 98--103.

Éech, R., Pajas, P. and Maéutek, J. (2010). Full valency. Verb valency without distinguishing complements and adjuncts. *Journal of Quantitative Linguistics*, 17, pp. 291--302.

Fillmore, C. (1982). Frame semantics. In The Linguistic Society of Korea (ed.), *Linguistics in the Morning Calm*. Seoul: Hanshin Publishing Co., pp. 111--137.

Kilgarriff, A., Rychly, P., Smrz, P. and Tugwell, D. (2004). The Sketch Engine. In *Proceedings of EURALEX 2004, Lorient, France*. pp 105--116. Available from http://www.sketchengine.co.uk.

Martens, S. (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 Conference, Hamburg,* pp. 43--50. Available from http://www.clarin-d.de/images/workshops/proceedingssoasforthehumanities.pdf.

Pajas, P. and Štěpánek, J. (2008). Recent advances in a feature-rich framework for treebank annotation. In *Proceedings of the 22nd International Conference on Computational Linguistics*, *August 18-22, 2008, Manchester, United Kingdom*. pp. 673--680.

Passarotti, M. (2011). Language Resources. The State of the Art of Latin and the *Index Thomisticus* Treebank Project. In M.-S. Ortola (ed.), *Corpus anciens et Bases de données*, «ALIENTO. Échanges sapientiels en Méditerranée», N°2. Nancy: Presses universitaires de Nancy, pp. 301--320.

Sgall, P., Hajicová, E. and Panevová, J. (1986). *The Meaning of the Sentence in its Semantic and Pragmatic Aspects*. Dordrecht NL: D. Reidel.

Tesnière, L. (1959). *Éléments de syntaxe structurale*. Paris, France: Editions Klincksieck.

Wattenberg, M. and Viégas, F.B. (2008). The word tree, an interactive visual concordance. *IEEE Trans. on Visualization and Computer Graphics*, 14(6), pp. 1221--1228.

Zeldes, A., Ritz, J., Lüdeling, A. and Chiarcos, C. (2009). ANNIS: A Search Tool for Multi-Layer Annotated Corpora. In *Proceedings of Corpus Linguistics 2009, July 20--23, Liverpool, UK*. Availabe from http://ucrel.lancs.ac.uk/publications/cl2009/358_FullPaper.doc.