Searching in the Penn Discourse Treebank Using the PML-Tree Query

Jiří Mírovský, Lucie Poláková, Jan Štěpánek
Charles University in Prague
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
(mirovsky|polakova|stepanek)@ufal.mff.cuni.cz

Abstract

The PML-Tree Query is a general, powerful and user-friendly system for querying richly linguistically annotated treebanks. The present paper shows how the PML-Tree Query can be used for searching for discourse relations in the Penn Discourse Treebank 2.0 mapped onto the syntactic annotation of the Penn Treebank.

Keywords: PML-Tree Query, Penn Discourse Treebank, searching in treebanks

1. Introduction

With increasing complexity of annotated treebanks, query tools and query languages also need to become more elaborate, in order to allow searching for and studying all phenomena annotated in the treebanks.[1] At the same time, it is important that the tools are user-friendly and the query languages, despite their inevitable complexity, remain easy to use.

The present paper shows a combination of leading representatives in two areas of NLP – discourse annotated treebanks on one side and query tools on the other, namely the Penn Discourse Treebank 2.0 (PDTB-2.0) and the PML-Tree Query (PML-TQ).

The PDTB-2.0, at the time of its publication in 2008, came with two possibilities to search in the data. First, there was the PDTB API describing in a hierarchy of Java classes the structure of the PDTB-2.0 data scheme and a skilled programmer could use it for searching in the data; for simple queries, there was a graphical interface as a part of the Java-based PDTB browser. Second, there was a conversion script for transforming the data from the native `.pdtb` format into a column format,[2] which could be used even by linguists without extensive programming skills to extract information about discourse relations using line-based tools such as `grep`, or spreadsheet programs. Yao et al. (2010) pointed out two shortcomings of the PDTB API, namely the impossibility to query the syntactic annotation of the Penn Treebank (PTB) and the discourse annotation of the PDTB-2.0 simultaneously, and the impossibility to incorporate and query additional annotation. They offered a solution based on their conversion of the PTB and the PDTB-2.0 data into an integrated XML format, and used XQuery – a general language for querying any XML data – for searching in the transformed data.

In our approach, we also transform the data of the PTB and the PDTB-2.0 into an XML format (PML, see Section 4.1). Unlike the previous approaches, for searching in the data we offer a powerful, yet user-friendly and graphically oriented system of the PML-Tree Query, a framework designed specifically for searching in richly linguistically annotated treebanks. Sections 2 and 3 shortly introduce the PDTB-2.0 and the PML-TQ, respectively. Section 4 describes the data format and annotation scheme used for encoding the data of the PTB and the PDTB-2.0 in the PML/PML-TQ. Section 5 demonstrates how to search in the PTB/PDTB-2.0 data in the PML-TQ. The Appendix provides examples of queries that can be used to generate summary tables from the appendices of the PDTB-2.0 annotation manual (Prasad et al., 2007).

2. The Penn Discourse Treebank

The Penn Discourse Treebank (PDTB; Miltsakaki et al., 2004) is one of the first and most influential corpus projects with discourse annotation. Its second version (PDTB-2.0; Prasad et al., 2008) comprises manual annotations of approx. 50 thousand English sentences of the Wall Street Journal portion of the Penn Treebank (PTB; Marcus et al., 1993). The (bottom-up) project focuses on marking discourse relations by primarily identifying their lexical cues – discourse connectives (explicit relations), the two discourse segments they connect (discourse arguments) and specifying a semantic type (sense) of such a relation. Where no explicit connective was present between two arguments (implicit relations), a connective best expressing the meaning of the relation was inserted. Also, other cues were marked: alternative lexicalizations of the connectives (AltLex), entity-based relations (EntRel), and places with no discourse relation at all (NoRel). An important part of the annotation is marking of attri-

[1] See e.g. a study on requirements that the complex annotation of the Prague Dependency Treebank (Hajič et al., 2006, Bejček et al., 2013) poses on search tools (Mírovský, 2009).

[2] The script (`convert.pl`) is a part of the PDTB-2.0 distribution and contains an error – it fails to transform second senses for AltLexes; the error can be fixed by changing the line 481 from "`if (@senseArray > 2) {`" to "`if (@senseArray > 1) {`".

bution (Prasad et al., 2006). The annotation was carried out on raw texts and then automatically mapped onto the constituency trees of the syntactic annotation of the Penn Treebank.

3. The PML-Tree Query

The PML-Tree Query (PML-TQ) is a powerful client–server system for searching in linguistically annotated treebanks (Pajas and Štěpánek, 2009). It was originally developed for searching in the Prague Dependency Treebank (PDT 2.0: Hajič et al., 2006, PDT 3.0: Bejček et al., 2013) but it is a general tool that can be used for any (both dependency and constituency) treebank encoded in the Prague Markup Language (Hana and Štěpánek, 2012).

The Prague Markup Language (PML)[3] is an abstract XML based format designed for annotation of linguistic corpora, especially treebanks. Data in the PML format can be browsed and edited in TrEd, a highly customizable tree editor (Pajas and Štěpánek, 2008).[4] The PML and TrEd, together with the PML-Tree Query system, form a general framework for treebank annotation and data processing. Virtually any treebank can be transformed into the PML, and once encoded in the PML, it can be browsed and edited in TrEd, and searched using the PML-TQ – see for example HamleDT (Zeman et al., 2014), a project of harmonizing 36 treebanks into a common data format and annotation scheme.[5]

The server part of the PML-TQ is implemented either as a relational database or as a system in btred, which is a command-line version of the tree editor TrEd. The client part uses either the tree editor TrEd along with the PML-TQ extension or a web browser.[6]

Queries in the PML-TQ can be created both in a textual form and (in the TrEd client) in a graphical environment with a visual creation of the query.[7] The query language allows to define properties of tree nodes and relations among them, inside or between sentences and also across layers of annotation; external resources such as lexicons can also be incorporated. Negation on the tree structure and Boolean expressions over the relations can be used. Results of the corpus search can be viewed along with the textual and tree context or processed with output filters to produce statistical tables.[8]

Possibilities of searching for discourse relations in the Prague Dependency Treebank 3.0 using the PML-TQ were first described in Mírovský et al. (2014), and later elaborated in more detail in Zikánová et al. (2015), where also the basics of the query language are explained. In the present paper, we show how to search – using the PML-TQ – for discourse relations in the Penn Discourse Treebank 2.0 mapped onto the syntactic trees of the Penn Treebank.

4. The PDTB-2.0 in the PML-TQ

4.1. The Data Format

As the PML is a general format for treebanks, a PML schema needs to be specified for any concrete data. It defines what types of nodes appear in the data, how they form a tree, what attributes can appear at a particular node type, which of them are obligatory or play a special role (like an identifier or an ordering attribute). Such a schema, which is an XML file too, has been defined for the PTB/PDTB-2.0 data. The following sample is a small part of the PTB/PDTB-2.0 PML schema, defining a tree node.

```
01  <type name="node.type">
02    <structure role="#NODE" name="node">
03      <member name="id" role="#ID" as_attribute="1"
         required="1">
04        <cdata format="ID"/>
05      </member>
06      <member name="type" required="1">
07        <choice>
08          <value>t</value>
09          <value>n</value>
10        </choice>
11      </member>
12      <member name="cat">
13      <alt type="cat.type"/>
14      </member>
15      <member name="functions">
16        <list ordered="1" type="function.type"/>
17      </member>
18      <member name="coindex.rf"><cdata
         format="PMLREF"/></member>
19      <member name="form"><cdata format="any"/>
        </member>
20      <member name="pos" type="postag.type"/>
21      <member name="order" role="#ORDER"><cdata
         format="nonNegativeInteger"/></member>
22      <member name="discourse">
23        <list ordered="0" type="discourse.type"/>
24      </member>
25      <member name="children" role="#CHILDNODES">
26        <list type="node.type" ordered="1"/>
27      </member>
28   </structure>
29  </type>
```

Line 2 says that this structure has a role of a tree node (`#NODE`). Lines 3 to 5 define an attribute carrying a corpus-wide unique identifier of the node (role `#ID`). Lines 6 to 11 define a distinction of non-terminal and terminal nodes. Lines 12 to 14 define an attribute for the syntactic category (its precise type cat.type to be defined later). Lines 15 to 17 define an attribute for a list of syntactic functions (again, its precise type to be defined later), and so on. Line 21 defines an ordering

[3] For an on-line documentation for the Prague Markup Language, see http://ufal.mff.cuni.cz/jazz/PML/.

[4] TrEd is written in Perl and can be easily customized for a desired purpose by extensions that are included into the system as modules. It is available from https://ufal.mff.cuni.cz/tred/ under the GPL – The General Public Licence.

[5] For treebanks that are publicly available for searching at the Prague PML-TQ server, see https://lindat.mff.cuni.cz/services/pmltq/#!/treebanks.

[6] There is also a command-line interface, capable to process a set of queries on a set of treebanks.

[7] Most parts of a query can be created by clicking on buttons, selecting values out of predefined lists, and dragging one node over another.

[8] A documentation for the PML-TQ can be found at http://ufal.mff.cuni.cz/pmltq/doc/pmltq_doc.html.

attribute (role `#ORDER`), which is used to order tree nodes horizontally. Lines 22 to 24 define an attribute for the discourse annotation, and lines 25 to 27 define an attribute carrying a list of children of the given node (role `#CHILDNODES`).

The following structure is a graphical representation of the part of the PML schema that defines a single discourse relation (referred to from line 23 of the previous listing):

```
discourse (Unordered list)
  type
  arg1 (Structure)
    text
    nodes.rf (Unordered list)
    terminals.rf (Unordered list)
    attribution (Structure)
      source
      source_lit
      type
      polarity
      determinacy
      text
      nodes.rf (Unordered list)
      terminals.rf (Unordered list)
  arg2 (Structure)
  sup1 (Structure)
  sup2 (Structure)
  sense
  sense2
  text
  head
  nodes.rf (Unordered list)
  terminals.rf (Unordered list)
  imp1_sense
  imp1_sense2
  imp1_text
  imp2_sense
  imp2_sense2
  imp2_text
  attribution (Structure)
  target_node.rf
  conn (Unordered list)
    type
    sem (Ordered list)
      order (1 or 2)
      sense
    head
    text
    nodes.rf (Unordered list)
    terminals.rf (Unordered list)
```

The attribute `discourse/type` (second line in the structure) contains one of the values `Explicit`, `AltLex`, `Implicit`, `EntRel` and `NoRel`. Structures `discourse/arg1`, `discouse/arg2`, `discourse/sup1` and `discourse/sup2` represent the arguments and supplementary texts for the discourse relation. Attributes of the argument 1 (Arg1) are listed under the branch of `discourse/arg1`, similar structures are used for the argument 2 (Arg2) and the supplementary texts (with the exception of attribution, which is not defined for supplementary texts). The surface form (rawtext) of the explicit connective with its possible modifications, or the alternative lexicalization is kept in the attribute `discourse/text`, the head of the connective is kept in the attribute `discourse/head`. Throughout the schema, the attribute `nodes.rf` contains a list of identifiers of non-terminals and terminals corresponding to the given text span, derived out of the gorn addresses from the PDTB-2.0 data, and the attribute `terminals.rf` represents a list of identifiers of terminals created as a projection of the nodes from the attribute `nodes.rf` (see Section 4.4 for a list of redundancies that have been added to the data during the transformation). The structure `attribution` is used for keeping attribution properties for the relation and the two arguments; its attributes reflect attributes and their values from the PDTB-2.0 data (see Section 4.4 for the description of the attribute `source_lit`). The potentially two senses of the discourse relation are kept in attributes `sense` and `sense2`. For implicit relations, the information about the potentially two connectives and, for each of them, potentially two senses is kept in attributes starting with `imp_`. The attribute `target_node.rf` contains an identifier of a representative node of Arg1 of the relation, it is a node selected out of the nodes in the list `nodes.rf` (more about this in Section 4.2). This flat representation of the connectives and their senses is supplemented by a structured version of this information, kept in the attribute `discourse/conn`. This redundancy was introduced to simplify the creation of some queries. The structured representation would be sufficient for all queries but the flat representation makes many queries simpler (more about redundancies in Section 4.4).

4.2. Arguments

After the transformation of the data into the PML format, each of the two arguments of a discourse relation is represented as a set of non-terminal and terminal nodes, derived from the gorn addresses of the original PDTB-2.0 data. The discourse relation itself is graphically represented as an oriented orange arrow going from Arg2 to Arg1 (see Figure 1). For allowing the arrow to start in a single node and end in (another) single node, a representative node for each argument was selected out of all nodes corresponding to the argument – it is the node (or the first of such nodes) that covers the largest span of terminals. Technically, all information about a discourse relation is kept in the attribute `discourse` at the representative node of Arg2 (where the arrow starts). As several discourse relations can start at a single node, the attribute `discourse` is a list of structures representing the individual relations.

4.3. Genres of Documents

Additionally to the annotations released with the PDTB-2.0, we have enriched the data for the PML-TQ with another, separately annotated genre-related attribute `genre_ad`. For each document, it contains one of eight possible values according to the genre classification of the Wall Street Journal texts corresponding to the PTB texts in the ACL/DCI Corpus (errata,

essays, highlights, letters, news, notable and quotable, quaterly progress reports, wit, and short verse).[9] Section 5.4 below offers an example of how document genres can be used in the PMT-TQ search in combination with attributes from other annotation levels.

4.4. Adding Redundancy

To keep a changing data consistent, especially during an annotation phase, a well designed annotation format does not contain redundancies. However, to make queries in the PML-TQ simpler, we have added several redundancies to the transformed data of the PDTB-2.0, as the last step just before the transformation of the data into the database format. Namely:

- The genre of a document, although it is a property of the document,[10] has been copied to the technical root[11] of each tree (attribute `genre_ad`).
- The surface form of the sentence has been added to the root of each tree (attribute `sentence`), to allow simple search in the surface form of the text.
- For each discourse argument, the attribution source (attribute `source`) has been copied to another attribute (`source_lit`) and in this copy, the value `Inh` has been replaced with the real inherited attribution source (e.g. `Ot`).
- The list of respective terminal nodes has been added to each list of nodes (attribute `terminals.rf`).
- The representative node for Arg1 has been added to each discourse relation (attribute `target_node.rf`).
- As a supplement to the structured attribute for the semantic annotation of a discourse relation (attribute `conn`), a flat version of the semantic information has been added (mainly attributes `sense`, `sense2` and attributes starting with `imp_`).

There are other less obvious redundancies in the schema, for example the text of an argument can be derived from the list of nodes representing the argument and as such does not need to be explicitly stated in the data. This redundancy was already present in the original PDTB-2.0 data and significantly simplifies queries that work with the surface text.[12]

[9] Details on the classification can be found in Webber (2009) and in its online appendix comparing the PDTB genre set of Webber with the one from the ACL/DCI Corpus: http://www.let.rug.nl/~bplank/metadata/genre_files_updated.html.

[10] with very few exceptions of multi-genred documents in the PDTB-2.0, which we treat as single-genred (preferring the genre that is less frequent in the corpus)

[11] The technical root is not displayed in the trees. It is, however, available in the queries as a node of the type `root`.

[12] The additional information is also useful because of some inconsistencies between the text spans explicitly stated in the annotated PDTB-2.0 data and text spans one obtains by projecting the respective gorn addresses to the text.

5. The PML-Tree Query in Action

The basic idea of creating a query in the PML-TQ is such that a user defines (draws in the graphical client environment) a tree that should be found in a result tree as a subtree. The query is then processed by the server and the results are reported back to the client.

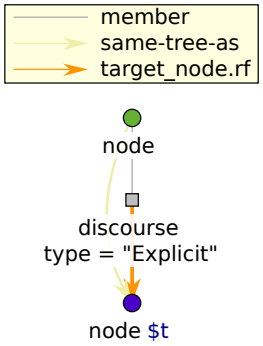
5.1. A Simple Example

Our first example shows how to search for a discourse relation. The query defines two nodes connected with a special `member` node that represents a discourse relation between arguments represented by the two nodes.[13] A required type of the discourse relation can be specified at the member node, in this example it is set to Explicit. The query also specifies that the start and target nodes of the relation (the representative nodes of Arg2 and Arg1) are from the same tree, i.e. it looks for an explicit intra-sentential discourse relation.

(1a) The textual form of the query:

```
node
  [ same-tree-as $t,
    member discourse
      [ type = "Explicit",
        target_node.rf node $t := [ ] ] ];
```

(1b) The graphical form of the query:



The following sentence represents one of the results of the query:

(1c) *The governor couldn't make it,* ***so*** *the lieutenant governor welcomed the special guests.* (PDTB-2.0)

Figure 1 below captures the graphical representation of the syntactic annotation of the sentence, along with the discourse relation represented by the thick orange arrow connecting roots of the two respective propositions. Spans of the arguments are highlighted by different background colours as groups of

[13] In the PML-TQ, member nodes are used to specify or mine properties of relations between nodes in cases where several relations of the given type can be defined at a single node, which is also the case of discourse relations in the PDTB-2.0.

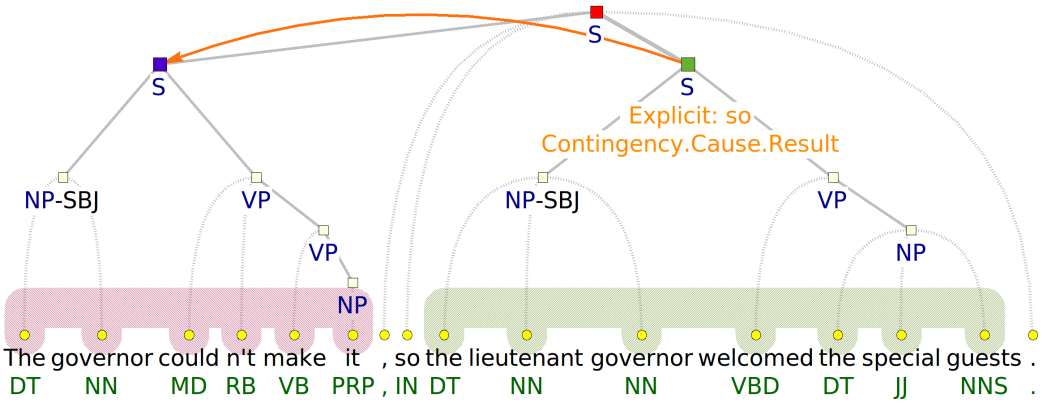


Figure 1: A result tree for the query 1a/1b

terminal nodes. Additional information – sense assigned to the discourse relation (in this case Contingency.Cause.Result) and the connective (so) – is displayed also in orange at the start node of the relation.

5.2. Output Filters

Results of queries in the PML-TQ can be further processed using so-called output filters. Thanks to an output filter, a result of a query does not consist of individual matching positions in the searched data but of a tabular summary of all matching positions, as specified by the output filter. In Example 2, we modify the previous query by adding another member node representing an explicit connective (`member conn`) and yet another member node representing each of potentially two senses assigned to the connective (`member sem`), naming the last member node `$s` and adding an output filter (the last line with the prefix `>>`). To save space, from now on we only show textual forms of the queries.

(2) The textual form of the query:

```
node
  [ same-tree-as $t,
    member discourse
      [ type = "Explicit", target_node.rf node $t := [ ],
        member conn
          [ member sem $s := [ ] ] ] ];
>> for $s.sense give $1,count() sort by $2 desc
```

The query searches for all explicit intra-sentential discourse relations in the data and – thanks to the output filter – produces a distribution table (see Table 1) of the senses of these relations, sorted in the descending order by the number of occurrences (only a few first lines are printed here to save space). If there are two senses assigned to a connective, each of them is counted separately in this query. For examples of queries with output filters where two senses for one connective are counted in combination, see the Appendix.

5.3. Querying Syntactic Trees

Example 3 combines the annotation of discourse relations with the syntactic annotation of the Penn Treebank. It searches for discourse relations with at least one of the arguments realized by a declarative clause with inverted subject–verb order (the subject follows the tensed verb or modal). Such clauses are represented by a node with the syntactic category SINV in the PTB annotation.

Sense	Count
Expansion.Conjunction	2 431
Contingency.Cause.Reason	1 475
Temporal.Synchrony	1 424
Temporal.Asynchronous.Succession	1 041
Comparison.Contrast	923
Contingency.Condition.Hypothetical	767
Temporal.Asynchronous.Precedence	731
Comparison.Contrast.Juxtaposition	591
Contingency.Cause.Result	444
...	

Table 1: (Selected) results of the query 2

(3) The textual form of the query:

```
node
  [ (cat = "SINV" or $t.cat = "SINV"),
    member discourse $d :=
      [ target_node.rf node $t := [ ] ] ];
```

If we add an output filter:

```
>> for $d.type give $1,count() sort by $2 desc
```

...we get a distribution of relation types for the results of the query from Example 3, see Table 2.

5.4. Querying Genres

Example 4 combines the annotation of discourse relations with the annotation of document genres. It searches for all senses annotated at all discourse relations in the data and produces distributions of the four semantic classes for each individual genre.

Type	Count
Implicit	365
EntRel	310
Explicit	104
AltLex	18
NoRel	8

Table 2: Results of the query 3 with the output filter

(4) The textual form of the query:

```
root $r :=
  [ descendant node
    [ member discourse
      [ member conn
          [ member sem $s := [ ] ] ] ] ];
>> for $r.genre_ad,match($s.sense,'^[^.]+')
    give $1,$2,count() sort by $1,$3 desc
>> give $1,$2,ceil($3 * 100 div sum($3 over $1)) & '%'
```

The query also shows how one line of an output filter can be put after another, further processing the output of the previous line. The first line of the output filter in Example 4 (split over two lines here) produces a temporary table with three columns: the genre, the semantic class (cut out of the whole sense label using a regular expression) and the number of co-occurrences of this genre with this class in the query results. The second line of the output filter is applied on this temporary table, coppies the first and the second column, counts the percentage of the occurrences of the semantic classes for each genre and adds the sign %. Table 3 shows results of the query for two first genres in the alphabetical order, i.e. errata and essay.

Genre	Class	Freq.
errata	Comparison	65%
errata	Contingency	18%
errata	Temporal	12%
errata	Expansion	6%
essay	Expansion	42%
essay	Contingency	25%
essay	Comparison	21%
essay	Temporal	14%
...		

Table 3: (Selected) results of the query 4

6. Conclusion

Without a powerful and at the same time simple and intuitive search tool, any annotated treebank of non-trivial size would only be of limited use. We have demonstrated how the PML-Tree Query system can be used for searching for individual discourse relations in the PDTB-2.0, as well as for creating tabular summaries of all occurrences of the searched phenomenon in the data. In the examples, we have combined the annotation of discourse relations with the annotation of syntactic trees and with the annotation of genres of documents. We hope that it is quite clear that adding any other annotation available for the Penn Treebank data would be very straightforward, as long as it can be reasonably mapped onto the syntactic trees of the PTB.

For further information about the transformation of the PTB and the PDTB-2.0 data into the PML/PML-TQ format and searching in the data, please visit the dedicated web page https://ufal.mff.cuni.cz/pdtb.

Acknowledgment

The authors gratefully acknowledge support from the Ministry of Education, Youth and Sports of the Czech Republic (project LH14011: Multilingual Corpus Annotation as a Support for Language Technologies, and project LM2015071: LINDAT/CLARIN). This work has been using language resources and tools developed, stored and distributed by the LINDAT/CLARIN project LM2015071.

Appendix: Example Queries

The following queries can be used to produce exact coppies[14] of the Appendices A to H of the PDTB-2.0 Annotation Manual.[15] Just reproducing the tables would of course not be of much consequence. The main benefit of the possibility to generate each of these tables with a single PML-TQ query is when the queries are modified to search only for a desired subset of discourse relations, restricted for example by a genre of the document or syntactic constructions involved in the relations or any other set of features needed for a given linguistic study.

Appendix A of the PDTB-2.0 Annotation Manual: Explicit connectives and their senses

```
node
  [ member discourse
    [ type = "Explicit",
      member conn $c :=
        [ member sem $s := [ ] ] ] ];
>> give $c,lower($c.head),match($s.sense,'[^\.]+$')
>> give distinct $1,$2,concat($3,'/' over $1 sort by $3)
>> for $2,$3 give $1,$2,count()
>> for $1,$2,$3 give $1,$2 & ' (' & $3 & ')',
    sum($3 over $1) sort by $1,$2
>> give distinct $1,concat($2,', ' over $1),$3 sort by $1
```

A few first lines of the result of the query are shown in Table 4.

Appendix B of the PDTB-2.0 Annotation Manual: Senses and their associated explicit connectives

The query is identical to the previous one with the exception of the third line of the output filter, where there are switched references to columns 2 and 3 from the previous line of the filter.

[14] without the headings of the columns and with some difference in the capitalization of senses

[15] https://www.seas.upenn.edu/~pdtb/PDTBAPI/pdtb-annotation-manual.pdf

Connective	Senses	Total
accordingly	Result (5)	5
additionally	Conjunction (7)	7
after	Expectation (2), Expectation/Succession (1), Reason/Succession (50), Specification/Succession (1), Succession (523)	577
afterward	Precedence (11)	11
also	Conjunction (1733), Conjunction/Synchrony (2), List (10), Specification (1)	1746
...		

Table 4: Appendix A of the PDTB-2.0 annotation manual, produced by a PML-TQ query

```
node
  [ member discourse
    [ type = "Explicit",
      member conn $c :=
        [ member sem $s := [ ] ] ] ];
>> give $c,lower($c.head),match($s.sense,'[^\.]+$')
>> give distinct $1,$2,concat($3,'/' over $1 sort by $3)
>> for $3,$2 give $1,$2,count()
>> for $1,$2,$3 give $1,$2 & ' (' & $3 & ')',
    sum($3 over $1) sort by $1,$2
>> give distinct $1,concat($2,', ' over $1),$3 sort by $1
```

Appendix C of the PDTB-2.0 Annotation Manual: Modified forms and variants of explicit connectives

```
node
  [ member discourse $d :=
    [ type = "Explicit" ] ];
>> for lower($d.head),lower($d.text) give $1,$2,count()
>> for $1,$2,$3 give $1,$2 & ' (' & $3 & ')',
    sum($3 over $1) sort by $1,$2
>> give distinct $1,concat($2,', ' over $1),$3 sort by $1
```

Appendix D of the PDTB-2.0 Annotation Manual: Implicit connectives and their senses

The query is identical to the respective query for explicit connectives with the exception of specifying `type = "Implicit"` instead of `"Explicit"`, and using attribute `text` instead of `head` in the first line of the output filter.

```
node
  [ member discourse
    [ type = "Implicit",
      member conn $c :=
        [ member sem $s := [ ] ] ] ];
>> give $c,lower($c.text),match($s.sense,'[^\.]+$')
>> give distinct $1,$2,concat($3,'/' over $1 sort by $3)
>> for $2,$3 give $1,$2,count()
>> for $1,$2,$3 give $1,$2 & ' (' & $3 & ')',
    sum($3 over $1) sort by $1,$2
>> give distinct $1,concat($2,', ' over $1),$3 sort by $1
```

Appendix E of the PDTB-2.0 Annotation Manual: Senses and their associated implicit connectives

The query is identical to the previous one with the exception of the third line of the output filter, where there are switched references to columns 2 and 3 from the previous line of the filter.

```
node
  [ member discourse
    [ type = "Implicit",
      member conn $c :=
        [ member sem $s := [ ] ] ] ];
>> give $c,lower($c.text),match($s.sense,'[^\.]+$')
>> give distinct $1,$2,concat($3,'/' over $1 sort by $3)
>> for $2,$3 give $1,$2,count()
>> for $1,$2,$3 give $1,$2 & ' (' & $3 & ')',
    sum($3 over $1) sort by $1,$2
>> give distinct $1,concat($2,', ' over $1),$3 sort by $1
```

Appendix F of the PDTB-2.0 Annotation Manual: AltLex senses

```
node
  [ member discourse
    [ type = "AltLex",
      member conn $c :=
        [ member sem $s := [ ] ] ] ];
>> give $c,match($s.sense,'[^\.]+$')
>> give distinct $1,concat($2,'/' over $1 sort by $2)
>> for $2 give $1,count() sort by $1
```

Appendix G of the PDTB-2.0 Annotation Manual: Attribution features for explicit relations and their arguments

```
node
  [ member discourse $d :=
    [ type = "Explicit" ] ];
>> give $d.attribution/source & '.' & $d.attribution/type
& '.' & $d.attribution/polarity & '.' & $d.attribution/
determinacy,$d.arg1/attribution/source & '.' & $d.arg1/
attribution/type & '.' & $d.arg1/attribution/polarity
& '.' & $d.arg1/attribution/determinacy,$d.arg2/
attribution/source & '.' & $d.arg2/attribution/type & '.'
& $d.arg2/attribution/polarity & '.' & $d.arg2/
attribution/determinacy
>> for $1,$2,$3 give $1,$2,$3,count() sort by $1,$2,$3
```

Appendix H of the PDTB-2.0 Annotation Manual: Attribution features for implicit relations and AltLexes, and their arguments [16]

```
node
  [ member discourse $d :=
    [ type in {"Implicit","AltLex"} ] ];
>> give $d.attribution/source & '.' & $d.attribution/type
& '.' & $d.attribution/polarity & '.' & $d.attribution/
determinacy,$d.arg1/attribution/source & '.' & $d.arg1/
attribution/type & '.' & $d.arg1/attribution/polarity
& '.' & $d.arg1/attribution/determinacy,$d.arg2/
attribution/source & '.' & $d.arg2/attribution/type & '.'
& $d.arg2/attribution/polarity & '.' & $d.arg2/
attribution/determinacy
>> for $1,$2,$3 give $1,$2,$3,count() sort by $1,$2,$3
```

Please note that for the last two queries, linebreaks in the attribute paths (e.g. `arg2/attribution/source`) need to be removed before the query can be processed by the PML-Tree Query server.

[16] For a reason yet unknown to the authors of the present paper, some numbers produced by this query differ from the numbers in the Appendix H of the PDTB-2.0 annotation manual. We believe our numbers to be correct, as they are slightly higher in all cases different from the Appendix H and they add up to a correct total number of implicit relations (16 053; implicit relations with two connectives are counted as one (not two)) and AltLexes (624) in the PDTB-2.0 data (in total 16 677).

References

Bejček, E., Hajičová, E., Hajič, J., Jínová, P., Kettnerová, V., Kolářová, V., Mikulová, M., Mírovský, J., Nedoluzhko, A., Panevová, J., Poláková, L., Ševčíková, M., Štěpánek, J., and Zikánová, Š. (2013). Prague Dependency Treebank 3.0. Data/software.

Hajič, J., Panevová, J., Hajičová, E., Sgall, P., Pajas, P., Štěpánek, J., Havelka, J., Mikulová, M., Žabokrtský, Z., Ševčíková-Razímová, M., and Urešová, Z. (2006). Prague Dependency Treebank 2.0. Data/software.

Hana, J. and Štěpánek, J. (2012). Prague Markup Language Framework. In *Proceedings of the Sixth Linguistic Annotation Workshop*, pages 12–21, Stroudsburg. Association for Computational Linguistics, Association for Computational Linguistics.

Marcus, M., Santorini, B., and Ann, M. M. (1993). Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330.

Miltsakaki, E., Prasad, R., Joshi, A., and Webber, B. (2004). The Penn Discourse Treebank. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon. European Language Resources Association.

Mírovský, J., Jínová, P., and Poláková, L. (2014). Discourse Relations in the Prague Dependency Treebank 3.0. In Lamia Tounsi et al., editors, *The 25th International Conference on Computational Linguistics, Proceedings of the Conference System Demonstrations*, pages 34–38, Dublin. Dublin City University.

Mírovský, J. (2009). *Searching in the Prague Dependency Treebank*. Charles University in Prague, Faculty of Mathematics and Physics, Institute of Formal and Applied Linguistics, Prague.

Pajas, P. and Štěpánek, J. (2008). Recent Advances in a Feature-Rich Framework for Treebank Annotation. In Donia Scott et al., editors, *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 673–680, Manchester. The Coling 2008 Organizing Committee.

Pajas, P. and Štěpánek, J. (2009). System for Querying Syntactically Annotated Corpora. In Gary Lee et al., editors, *Proceedings of the ACL–IJCNLP 2009 Software Demonstrations*, pages 33–36, Suntec. Association for Computational Linguistics.

Prasad, R., Dinesh, N., Lee, A., Joshi, A., and Webber, B. (2006). Annotating Attribution in the Penn Discoure Treebank. In *Proceedings of the COLING/ACL Workshop on Sentiment and Subjectivity in Text*, pages 31–38, Sydney. Association for Computational Linguistics.

Prasad, R., Miltsakaki, E., Dinesh, N., Lee, A., Joshi, A., Robaldo, L., and Webber, B. (2007). *The Penn Discourse Treebank 2.0 Annotation Manual*. Number IRCS-08-01, Technical Report. Institute for Research in Cognitive Science, University of Pennsylvania, Philadelphia.

Prasad, R., Dinesh, N., Lee, A., Miltsakaki, E., Robaldo, L., Joshi, A., and Webber, B. (2008). The Penn Discourse Treebank 2.0. In Nicoletta Calzolari, et al., editors, *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, pages 2961–2968, Marrakech. European Language Resources Association.

Webber, B. (2009). Genre Distinctions for Discourse in the Penn TreeBank. In Keh-Yih Su, et al., editors, *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 674–682, Suntec. Association for Computational Linguistics.

Yao, X., Borisova, I., and Alam, M. (2010). PDTB XML: the XMLization of the Penn Discourse TreeBank 2.0. In Nicoletta Calzolari, et al., editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association.

Zeman, D., Dušek, O., Mareček, D., Popel, M., Ramasamy, L., Štěpánek, J., Žabokrtský, Z., and Hajič, J. (2014). HamleDT: Harmonized Multi-Language Dependency Treebank. *Language Resources and Evaluation*, 48(4):601–637.

Zikánová, Š., Hajičová, E., Hladká, B., Jínová, P., Mírovský, J., Nedoluzhko, A., Poláková, L., Rysová, K., Rysová, M., and Václ, J. (2015). *Discourse and Coherence. From the Sentence Structure to Relations in Text*. Studies in Computational and Theoretical Linguistics. ÚFAL, Praha, Czechia.