PDTB 1.0
Documentation of file formats

Nikhil Dinesh and Alan Lee
nikhild,aleewk@linc.cis.upenn.edu

1 Introduction

This document describes the format of the annotations in the first release by of the Penn Discourse Treebank (PDTB) and how they are linked to the Wall Street Journal corpus and Penn Treebank [1] annotations. Section 2 describes the default directory structure, and the mechanisms used to link the files. Section 3 describes the format of the PDTB annotation files. Section 4 gives an overview of the procedure used to link the PDTB annotations to the Penn Treebank. The reader is assumed to be familiar with what the Penn Discourse Treebank annotates, and is referred to the annotation guidelines available at `http://www.seas.upenn.edu/~pdtb` for details that are not discussed here.

2 Directory structure and Linking mechanisms

1. RAW refers to the Wall Street Journal raw text. RAW is assumed to be divided into 25 sections, each with at most 100 files. The directory RawRoot refers to the directory such that RawRoot/00/wsj_0003 is the RAW file for section 00, file 03.[1]

2. PDTB refers to the Penn Discourse Treebank. PdtbRoot refers to the directory such that PdtbRoot/00/wsj_0003.pdtb contains the PDTB annotations for RawRoot/00/wsj_0003.

3. PTB refers to the Penn Treebank [1]. PTB files are assumed to be in symbolic expression form, and PtbRoot/00/wsj_0003.mrg contains the parse trees for RawRoot/00/wsj_0003.

[1] RAW files are assumed to have only the linefeed (lf) as the newline character (as is used on Unix). Replacing this with carriage return followed by linefeed (crlf), as in Windows, will result in spans be incorrectly interpreted.

4. Given a PDTB file PdtbRoot/ij/wsj_ijkl.pdtb, the associated RAW file is RawRoot/ij/wsj_ijkl, and the associated PTB file is PtbRoot/ij/wsj_ijkl.

5. PDTB files are linked to RAW files using *spans*. A span $p..q$ denotes the string in the associated RAW file starting from character p (inclusive) to character q (exclusive). For example, given the string *hello*, the span 0..1 is h, and the span 1..3 is the string *el*. A *span list* is given by $p_1..q_1; p_2..q_2....; p_n..q_n$. We will assume that $q_i \leq p_{i+1}$ in this document. Given the string *hello* the span list $1..3; 4..5$ denotes *el o* (note that a space is added between spans).

6. PDTB files are linked to PTB files using *Gorn addresses*. A Gorn address $a_1, a_2, ... \ a_{n-1}, a_n$ denotes the a_nth child of the a_{n-1}th child of .... the a_2th child of the sentence number a_1 in the associated PTB file. Given a Gorn address $a_1, a_2, ... \ a_{n-1}, a_n$ - $T(a_1, a_2, ... \ a_n)$ denotes the subtree rooted at a_n. Given a PTB file with two sentences:

```
((S0 (A a) (B b)))
((S1 (C c) (D d)))
```

The Gorn address $0, 0$ refers to the node A. The Gorn address $1, 1, 0$ refers to the node d. $T(0, 0)$ denotes the subtree rooted at A, i.e., (Aa). $T(0, 1)$ denotes the subtree rooted at B, i.e., (Bb), and $T(1, 1, 0)$ denotes the subtree rooted at terminal d. Throughout this document, we will only need to refer to subtrees rooted at a particular node, and never to a node in isolation. So we simply write 0,0 to denote the subtree rooted at A, and we use the phrase *referring to a node* to mean *referring to the subtree rooted at the node.*[2]

Note that 0 refers to (the subtree rooted at) $S0$, and 1 refers to $S1$ and *not* to the TOP node inserted by several APIs. Let G_1, G_2 ...G_n be Gorn addresses, then a *Gorn address list* is given by $G_1; G_2...; G_n$. A Gorn address list of length n denotes n nodes. We will assume that G_i is not a prefix (denoting an ancestor) of G_j for all $1 \leq i, j \leq n$.

3 PDTB File Format

Each PDTB file contains a list of relations. The following is the BNF (non-terminals start with lower case, terminals start with upper case and ϵ denotes the empty production):

[2]Note that a label is a property of a node, and there is no separate address for a node label.

```
relationList ::= relation relationList | relation

relation ::=
     Explicit explicitRelation
   | Implicit implicitRelation
   | AltLex altLexRelation
   | EntRel entityRelation
   | NoRel noRelation

explicitRelation ::= selection explicitRelationFeatures  sup arg arg sup

altLexRelation ::= selection altLexRelationFeatures sup arg arg sup

implicitRelation ::= inferenceSite implicitRelationFeatures sup arg arg sup

entityRelation ::= inferenceSite arg arg

noRelation ::= inferenceSite arg arg

arg ::=  Arg selection argFeatures

sup ::= Sup selection | ϵ

explicitRelationArg ::= Arg selection

explicitRelationFeatures ::=  Source Factuality Polarity ConnHead

altLexRelationFeatures ::=  Source Factuality Polarity SemanticClass

implicitRelationFeatures ::=
     Source Factuality Polarity Conn1 SemanticClass1 Conn2 SemanticClass2
   | Source Factuality Polarity Conn1 SemanticClass1

argFeatures ::= Source Factuality Polarity

selection ::= SpanList GornAddressList

inferenceSite ::= StringPosition SentenceNumber
```

Note that the terminals are types, not actual values. We now give examples of each type of relation. At most one terminal appears per line, and so the type is given to the right of appropriate lines in parentheses (the types do not appear in the files). The characters // are used to denote line breaks that do not appear in the actual file, but are used here for formatting. The lines starting with a # are comments.

```
------------------------------------------------------
____Explicit____                    (Explicit)
3672..3683                          (SpanList)
26,1,1,4,1,1,3,0;26,1,1,4,1,1,3,1 (GornAddressList)
#### Features ####
Ot                                  (Source)
Fact                                (Factuality)
Pos                                 (Polarity)
though                              (ConnHead)
###################
____Sup1____                        (Sup)
3595..3633                          (SpanList)
26,0;26,1,0;26,1,1,0;//
26,1,1,1;26,1,1,2;26,1,1,3;26,2    (GornAddressList)
###################
____Arg1____                        (Arg)
3635..3670                          (SpanList)
26,1,1,4,0;26,1,1,4,1,0;//
26,1,1,4,1,1,0;26,1,1,4,1,1,1;//
26,1,1,4,1,1,2                      (GornAddressList)
#### Features ####
Inh                                  (Source)
Null                                 (Factuality)
Pos                                  (Polarity)
###################
____Arg2____                        (Arg)
3684..3716
26,1,1,4,1,1,3,2                    (GornAddressList)
#### Features ####
Inh                                  (Source)
Null                                 (Factuality)
Pos                                  (Polarity)
###################
------------------------------------------------------
```

The possible values for terminals are as follows:

1. SpanList and GornAddressList have been explained in Section 2. The SpanList corresponds to selections made by the annotator, while the GornAddressList is computed programmatically given the SpanList. See Section 4 for a brief description.

2. The Source of the relation is Wr (for writer) or Ot (for other). The Source of an argument is Wr (for writer and different from the Source of the relation), Ot (for other and different from the Source of the relation), and Inh (for inherits the Source of the relation).

3. The Factuality of the relation is Fact (for factual) or NonFact (for non-factual). The Factuality of an argument is Fact (for factual and different from the Factuality of the relation), NonFact (for non-factual and different from the Factuality of the relation), and Inh (for inherits the Factuality of the relation).

4. The Polarity of the relations is Pos (for positive) or Neg (for negative).

5. ConnHead is the head of the connective. For example, the connective above is *even though* which is a modified form of the connective *though*.

The following is a fragment of an implicit relation. Only the portions differing from explicit relations are shown:

```
------------------------------------------------------
____Implicit____                    (Implicit)
39                                  (StringPosition)
1                                   (SentenceNo)
#### Features ####
Wr                                  (Source)
Fact                                (Factuality)
Pos                                 (Polarity)
because                             (Conn1)
Cause                               (SemanticClass1)
##################
...Args and Sups as for Explicit Relations
```

The values for terminals, not explained above, are as follows:

1. StringPosition, and SentenceNo give the site of inference of the implicit connective. The StringPosition is the offset of the first character of Arg2 of the implicit relation and SentenceNo is the sentence number of Arg2.

2. Conn1 and SemanticClass1 corresponding to the connective, and the semantic class. These are required. Conn2 and SemanticClass2 are optional. Note that if Conn2 occurs SemanticClass2 will also occur. The set of possible values for Conn1 and Conn2 are the same, and so are the set of possible values for SemanticClass1 and SemanticClass2. A discussion of these can be found in the guidelines.

The following is a fragment of an AltLex relation (adjacent sentences related by a phrase which is not a connective):

```
------------------------------------------------------
____AltLex____                    (AltLex)
1080..1095                        (SpanList)
9,0,0;9,0,1,0;9,0,1,1,0           (GornAddressList)
#### Features ####
Wr                                (Source)
Fact                              (Factuality)
Pos                               (Polarity)
Consequence                       (SemanticClass)
##################
...Args and Sups as for Explicit Relations.
```

The SemanticClass terminal has the same set of possible values as SemanticClass1 for implicit relations.

EntRel and NoRel are different only in that no features appear for these types.

4 Computation of Gorn Addresses

We use τ to denote PTB nodes whose yield consists of only punctuation or traces. We use γ to denote clausal PTB nodes - these are nodes whose label starts with S, PRN (if it has a child whose label starts with S), or PP (if it has a child whose label starts with S). The arguments selected in the PDTB, in most cases, are intended to correspond to γ nodes. However, to reduce the overhead in annotation, annotators select spans in the RAW files. The question that arises is which τ nodes on the periphery of a selection need to be included. Consider the following (synthetic) scenario:

```
RAW:

John goes home early, when he is tired.

PTB:

((S
   (NP (NN John))
   (VP (V goes)
       (NP (NN home))
       (ADVP (RB early))
       (, ,)
       (SBAR (WHADVP_1 (RB when)
             (S
               (NP (PRP he))
               (VP (V is)
                   (ADJP (JJ tired))
                   (ADVP-TMP (-NONE- *T*-1))
               )
             )
       )
   )
   (. .)
))

PDTB:

Conn: 22..26 (when)
Arg1: 0..20  (John goes home early)
Arg2: 27..38 (he is tired)
```

In this case, if the minimal set of nodes correponding to the RAW selections were computed the τ nodes correponding to the comma, ADVP trace, and period would be excluded. These cases are handled in two steps:

1. *Span stretching*- The selections are stretched (in both directions) to include τ nodes on their periphery, as long as (i) no other lexical item is included, and (ii) all γ nodes are respected. The latter constraint ensures that a γ node is crossed or contained only if it is crossed or contained by the original selection. This

ensures that the τ node corresponding to the period is not included in Arg2, because to include this the $SBAR$ node (which is a γ node) would have to be crossed, and this is not crossed by the annotator's selection.

The GornAddressList, at the end of this phase, denotes the highest set of nodes that dominate the stretched span exactly. For Conn this would be $0, 1, 4, 0$. For Arg1: $0, 0; 0, 1, 0; 0, 1, 1; 0, 1, 2; 0, 1, 3$. For Arg2: $0, 1, 4, 1$. Note that the period with address $0, 2$ does not occur in any selection.

2. *Sibling inclusion*- For each node in Arg1, for each sibling which is a τ node which does not occur in Conn, Arg2, Sup1, or Sup2 add it to Arg1. And similarly for Arg2, Sup1 and Sup2 (in that order). At the end of this step the period is associated with Arg1.[3]

References

[1] Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz, "Building a large scale annotated corpus of English: the Penn Treebank," *Computational Linguistics*, vol. 19, 1993.

[3] To handle coordinated structure, we ensure that span stretching and sibling inclusion cannot cross a coordinating conjunction.