Stanford typed dependencies manual

Marie-Catherine de Marneffe and Christopher D. Manning

September 2008

1 Introduction

The Stanford typed dependencies representation was designed to provide a simple description of the grammatical relationships in a sentence that can easily be understood and effectively used by people without linguistic expertise who want to extract textual relations. In particular, rather than the phrase structure representations that have long dominated in the computational linguistic community, it represents all sentence relationships uniformly as typed dependency relations between pairs of words, such as "the subject of *distributes* is *Bell*." Our experience is that this simple, uniform representation is quite accessible to non-linguists thinking about tasks involving information extraction from text and is quite effective in relation extraction applications.

Here is an example. For the sentence:

> *Bell, based in Los Angeles, makes and distributes electronic, computer and building products.*

the Stanford Dependencies (SD) representation is:

> nsubj(makes-8, Bell-1)
> nsubj(distributes-10, Bell-1)
> partmod(Bell-1, based-3)
> nn(Angeles-6, Los-5)
> prep_in(based-3, Angeles-6)
> conj_and(makes-8, distributes-10)
> amod(products-16, electronic-11)
> conj_and(electronic-11, computer-13)
> amod(products-16, computer-13)
> conj_and(electronic-11, building-15)
> amod(products-16, building-15)
> dobj(makes-8, products-16)
> dobj(distributes-10, products-16)

This maps straightforwardly onto a directed graph representation, in which words in the sentence are nodes in the graph and grammatical relations are edge labels. Figure 1 gives the graph representation for the example sentence above.

This manual provides documentation about the set of dependencies defined for English. (There is also a Stanford Dependency representation available for Chinese, but it is not further

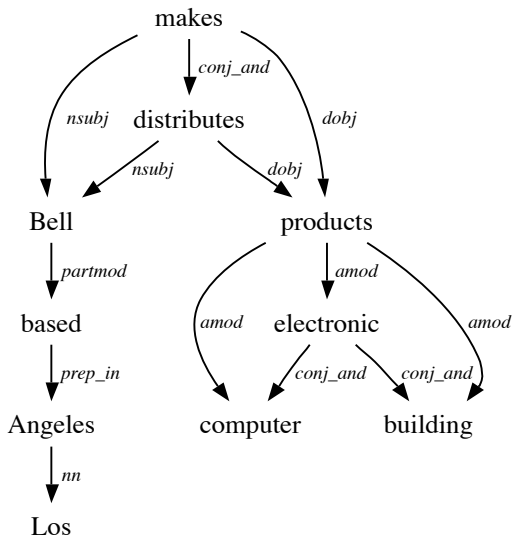

Figure 1: Graphical representation of the Stanford Dependencies for the sentence "Bell, based in Los Angeles, makes and distributes electronic, computer and building products."

discussed here.) The bulk of the manual defines the relations and the taxonomic hierarchy over them. This is then followed by some details of the software we have for generating Stanford Dependencies, a description of several variant dependency representations suitable in different contexts, and references to further discussion and use of the SD representation.

2 Definitions of the Stanford typed dependencies

The current representation contains 55 grammatical relations. The dependencies are all binary relations: a grammatical relation holds between a governor and a dependent. The grammatical relations are defined below, in alphabetical order according to the dependency abbreviated name (which appears in the parser output). The definitions make use of the Penn Treebank part-of-speech tags and phrasal labels.

***abbrev*: abbreviation modifier**
An abbreviation modifier of an NP is a parenthesized NP that serves to abbreviate the NP (or to define an abbreviation).

"The Australian Broadcasting Corporation (ABC)" *abbrev*(Corporation, ABC)

***acomp*: adjectival complement**
An adjectival complement of a VP is an adjectival phrase which functions as the complement (like an object of the verb); an adjectival complement of a clause is the adjectival complement of the VP which is the predicate of that clause.

"She looks very beautiful" *acomp*(looks, beautiful)

***advcl*: adverbial clause modifier**
An adverbial clause modifier of a VP is a clause modifying the verb (temporal clause, consequence, conditional clause, etc.).

"The accident happened as the night was falling"	*advcl*(happened, falling)
"If you know who did it, you should tell the teacher"	*advcl*(tell, know)

***advmod*: adverbial modifier**
An adverbial modifier of a word is a (non-clausal) RB or ADVP that serves to modify the meaning of the word.

"Genetically modified food"	*advmod*(modified, genetically)
"less often"	*advmod*(often, less)

***agent*: agent**
An agent is the complement of a passive verb which is introduced by the preposition "by" and does the action.

"The man has been killed by the police"	*agent*(killed, police)
"Effects caused by the protein are important"	*agent*(caused, protein)

***amod*: adjectival modifier**
An adjectival modifier of an NP is any adjectival phrase that serves to modify the meaning of the NP.

"Sam eats red meat"	*amod*(meat, red)

***appos*: appositional modifier**
An appositional modifier of an NP is an NP immediately to the right of the first NP that serves to define or modify that NP. It includes parenthesized examples.

"Sam, my brother"	*appos*(Sam, brother)
"Bill (John's cousin)"	*appos*(Bill, cousin)

***attr*: attributive**
An attributive is the complement of a copular verb such as "to be", "to seem", "to appear".

"What is that?"	*attr*(is, What)

***aux*: auxiliary**
An auxiliary of a clause is a non-main verb of the clause, e.g. modal auxiliary, "be" and "have" in a composed tense.

"Reagan has died"	*aux*(died, has)
"He should leave"	*aux*(leave, should)

***auxpass*: passive auxiliary**

A passive auxiliary of a clause is a non-main verb of the clause which contains the passive information.

"Kennedy has been killed"	*auxpass*(killed, been)
	aux(killed,has)
"Kennedy was/got killed"	*auxpass*(killed, was/got)

***cc*: coordination**

A coordination is the relation between an element of a conjunct and the coordinating conjunction word of the conjunct.

"Bill is big and honest"	*cc*(big, and)
"They either ski or snowboard"	*cc*(ski, or)

***ccomp*: clausal complement**

A clausal complement of a VP or an ADJP is a clause with internal subject which functions like an object of the verb or of the adjective; a clausal complement of a clause is the clausal complement of the VP or of the ADJP which is the predicate of that clause. Such clausal complements are usually finite (though there are occasional remnant English subjunctives).

"He says that you like to swim"	*ccomp*(says, like)
"I am certain that he did it"	*ccomp*(certain, did)

***complm*: complementizer**

A complementizer of a clausal complement (*ccomp*) is the word introducing it. It will be the subordinating conjunction "that" or "whether".

"He says that you like to swim"	*complm*(like, that)

***conj*: conjunct**

A conjunct is the relation between two elements connected by a coordinating conjunction, such as "and", "or", etc. We treat conjunctions asymmetrically: The head of the relation is the first conjunct and other conjunctions depend on it via the *conj* relation.

"Bill is big and honest"	*conj*(big, honest)
"They either ski or snowboard"	*conj*(ski, snowboard)

***cop*: copula**

A copula is the relation between the complement of a copular verb and the copular verb.

"Bill is big"	*cop*(big, is)
"Bill is an honest man"	*cop*(man, is)

***csubj*: clausal subject**
A clausal subject is a clausal syntactic subject of a clause, i.e. the subject is itself a clause. The governor of this relation might not always be a verb: when the verb is a copular verb, the root of the clause is the complement of the copular verb. In the two following examples, "what she said" is the subject.

"What she said makes sense"	*csubj*(makes, said)
"What she said is not true"	*csubj*(true, said)

***csubjpass*: clausal passive subject**
A clausal passive subject is a clausal syntactic subject of a passive clause. In the example below, "that she lied" is the subject.

"That she lied was suspected by everyone"	*csubjpass*(suspected, lied)

***det*: determiner**
A determiner is the relation between the head of an NP and its determiner.

"The man is here"	*det*(man, the)
"Which book do you prefer?"	*det*(book, which)

***dobj*: direct object**
The direct object of a VP is the noun phrase which is the (accusative) object of the verb; the direct object of a clause is the direct object of the VP which is the predicate of that clause.

"She gave me a raise"	*dobj*(gave, raise)
"They win the lottery"	*dobj*(win, lottery)

***expl*: expletive**
This relation captures an existential "there". The main verb of the clause is the governor.

"There is a ghost in the room"	*expl*(is, There)

***infmod*: infinitival modifier**
An infinitival modifier of an NP is an infinitive that serves to modify the meaning of the NP.

"Points to establish are . . . "	*infmod*(points, establish)
"I don't have anything to say"	*infmod*(anything, say)

***iobj*: indirect object**
The indirect object of a VP is the noun phrase which is the (dative) object of the verb; the indirect object of a clause is the indirect object of the VP which is the predicate of that clause.

"She gave me a raise"	*iobj*(gave, me)

***mark*: marker**
A marker of an adverbial clausal complement (*advcl*) is the word introducing it. It will be a subordinating conjunction different from "that" or "whether": e.g. "because", "when", "although", etc.

"Forces engaged in fighting after insurgents attacked"	*mark*(attacked, after)

***measure*: measure-phrase modifier**
The measure-phrase modifier is the relation between the head of an ADJP/ADVP and the head of a measure-phrase modifying the ADJP/ADVP.

"The director is 65 years old"	*measure*(old, years)
"6 feet long"	*measure*(long, feet)

***neg*: negation modifier**
The negation modifier is the relation between a negation word and the word it modifies.

"Bill is not a scientist"	*neg*(scientist, not)
"Bill doesn't drive"	*neg*(drive, n't)

***nn*: noun compound modifier**
A noun compound modifier of an NP is any noun that serves to modify the head noun. (Note that in the current system for dependency extraction, all nouns modify the rightmost noun of the NP – there is no intelligent noun compound analysis. This is likely to be fixed once the Penn Treebank represents the branching structure of NPs.)

"Oil price futures"	*nn*(futures, oil)
	nn(futures, price)

***nsubj*: nominal subject**
A nominal subject is a noun phrase which is the syntactic subject of a clause. The governor of this relation might not always be a verb: when the verb is a copular verb, the root of the clause is the complement of the copular verb.

"Clinton defeated Dole"	*nsubj*(defeated, Clinton)
"The baby is cute"	*nsubj*(cute, baby)

***nsubjpass*: passive nominal subject**
A passive nominal subject is a noun phrase which is the syntactic subject of a passive clause.

"Dole was defeated by Clinton"	*nsubjpass*(defeated, Dole)

***num*: numeric modifier**
A numeric modifier of an NP is any number phrase that serves to modify the meaning of the NP.

"Sam eats 3 sheep" *num*(sheep, 3)

***number*: element of compound number**
An element of compound number is a part of a number phrase or currency amount.

"I lost $ 3.2 billion" *number*($, billion)

***parataxis*: parataxis**
The parataxis relation (from Greek for "place side by side") is a relation between the main verb of a clause and other sentential elements, such as a sentential parenthetical, a clause after a ":" or a ";".

"The guy, John said, left early in the morning" *parataxis*(left, said)

***partmod*: participial modifier**
A participial modifier of an NP or VP is a participial verb form that serves to modify the meaning of the NP or VP.

"Truffles picked during the spring are tasty" *partmod*(truffles, picked)
"Bill tried to shoot demonstrating his incompetence" *partmod*(shoot, demonstrating)

***pcomp*: prepositional complement**
The prepositional complement of a preposition is the head of a clause following the preposition.

"We have no information on whether users are at risk" *pcomp*(on, are)
"They heard about you missing classes" *pcomp*(about, missing)

***pobj*: object of a preposition**
The object of a preposition is the head of a noun phrase following the preposition. (The preposition in turn may be modifying a noun, verb, etc.) Unlike the Penn Treebank, we here define cases of VBG quasi-prepositions like "including", "concerning", etc. as instances of *pobj*. (The preposition can be called a FW for "pace", "versus", etc. It can also be called a CC – but we don't currently handle that and would need to distinguish from conjoined prepositions.)

"I sat on the chair" *pobj*(on, chair)

***poss*: possession modifier**
The possession modifier relation holds between the head of an NP and its possessive determiner, or a genitive 's complement.

"their offices" *poss*(offices, their)
"Bill's clothes" *poss*(clothes, Bill)

***possessive*: possessive modifier**
The possessive modifier relation appears between the head of an NP and the genitive 's.

"Bill's clothes" *possessive*(John, 's)

***preconj*: preconjunct**
A preconjunct is the relation between the head of an NP and a word that is part of a conjunction, an puts emphasis on it (e.g., "either", "both", "neither").

"Both the boys and the girls are here" *preconj*(boys, both)

***predet*: predeterminer**
A predeterminer is the relation between the head of an NP and a word that precedes and clarifies the use of the NP determiner.

"All the boys are here" *predet*(boys, all)

***prep/prepc*: prepositional modifier**
A prepositional modifier of a verb, adjective, or noun is any prepositional phrase that serves to modify the meaning of the verb, adjective, or noun. If the prepositional phrase is a clause, the relation is called *prepc* when collapsing takes place (see section 4).

"I saw a cat in a hat" *prep*(cat, in)
"I saw a cat with a telescope" *prep*(saw, with)
"He is responsible for meals" *prep*(responsible, for)

***prt*: phrasal verb particle**
The phrasal verb particle relation identifies a phrasal verb, and holds between the verb and its particle.

"They shut down the station" *prt*(shut, down)

***punct*: punctuation**
This is used for any piece of punctuation in a clause, if punctuation is being retained in the typed dependencies. By default, punctuation is not retained in the output.

"Go home!" *punct*(Go, !)

***purpcl*: purpose clause modifier**
A purpose clause modifier of a VP is a clause headed by "(in order) to" specifying a purpose. At present the system only recognizes ones that have "in order to" as otherwise the system is unable to distinguish from the surface representations between these and open clausal complements (*xcomp*). It can also recognize "to" clauses introduced by "be VBN".

"He talked to him in order to secure the account" *purpcl*(talked, secure)

***quantmod*: quantifier phrase modifier**
A quantifier modifier is an element modifying the head of a QP constituent.

"About 200 people came to the party" *quantmod*(200, About)

***rcmod*: relative clause modifier**
A relative clause modifier of an NP is a relative clause modifying the NP. The relation points from the head noun of the NP to the head of the relative clause, normally a verb.

"I saw the man you love" *rcmod*(man, love)
"I saw the book which you bought" *rcmod*(book,bought)

***ref*: referent**
A referent of the head of an NP is the relative word introducing the relative clause modifying the NP.

"I saw the book which you bought" *ref*(book, which)

***rel*: relative**
A relative of a relative clause is the head word of the WH-phrase introducing it.

"I saw the man who you love" *rel*(love, who)
"I saw the man whose wife you love" *rel*(love, wife)

***tmod*: temporal modifier**
A temporal modifier of a VP or an ADJP is any constituent that serves to modify the meaning of the VP or the ADJP by specifying a time; a temporal modifier of a clause is a temporal modifier of the VP which is the predicate of that clause.

"Last night, I swam in the pool" *tmod*(swam, night)

***xcomp*: open clausal complement**
An open clausal complement (*xcomp*) of a VP or an ADJP is a clausal complement without its own subject, whose reference is determined by an external subject. These complements are always non-finite. The name *xcomp* is borrowed from Lexical-Functional Grammar.

"He says that you like to swim" *xcomp*(like, swim)
"I am ready to leave" *xcomp*(ready, leave)

***xsubj*: controlling subject**
A controlling subject is the relation between the head of a open clausal complement (*xcomp*) and the external subject of that clause.

"Tom likes to eat fish" *xsubj*(eat, Tom)

3 Hierarchy of typed dependencies

The grammatical relations defined in the above section stand in a hierarchy. The most generic grammatical relation, dependent (*dep*), will be used when a more precise relation in the hierarchy does not exist or cannot be retrieved by the system.

dep - dependent
- *aux* - auxiliary
  - *auxpass* - passive auxiliary
  - *cop* - copula
- *arg* - argument
  - *agent* - agent
  - *comp* - complement
    - *acomp* - adjectival complement
    - *attr* - attributive
    - *ccomp* - clausal complement with internal subject
    - *xcomp* - clausal complement with external subject
    - *compl* - complementizer
    - *obj* - object
      - *dobj* - direct object
      - *iobj* - indirect object
      - *pobj* - object of preposition
    - *mark* - marker (word introducing an *advcl*)
    - *rel* - relative (word introducing a *rcmod*)
  - *subj* - subject
    - *nsubj* - nominal subject
      - *nsubjpass* - passive nominal subject
    - *csubj* - clausal subject
      - *csubjpass* - passive clausal subject
- *cc* - coordination
- *conj* - conjunct
- *expl* - expletive (expletive “there”)
- *mod* - modifier
  - *abbrev* - abbreviation modifier
  - *amod* - adjectival modifier
  - *appos* - appositional modifier
  - *advcl* - adverbial clause modifier
  - *purpcl* - purpose clause modifier
  - *det* - determiner
  - *predet* - predeterminer
  - *preconj* - preconjunct
  - *infmod* - infinitival modifier
  - *partmod* - participial modifier
  - *advmod* - adverbial modifier
    - *neg* - negation modifier

rcmod - relative clause modifier
quantmod - quantifier modifier
tmod - temporal modifier
measure - measure-phrase modifier
nn - noun compound modifier
num - numeric modifier
number - element of compound number
prep - prepositional modifier
poss - possession modifier
possessive - possessive modifier (*'s*)
prt - phrasal verb particle
parataxis - parataxis
punct - punctuation
ref - referent
sdep - semantic dependent
xsubj - controlling subject

4 Different styles of dependency representation

Four variants of the typed dependency representation are available in the dependency extraction system provided with the Stanford parser. The representations follow the same format: a dependency is written as *abbreviated_relation_name*(governor, dependent) where the governor and the dependent are words in the sentence to which the word number in the sentence is append. The differences are that they range from a more surface-oriented representation, where each token appears as a dependent in a tree, to a more semantically interpreted representation where certain word relationships, such as prepositions, are represented as dependencies, and the set of dependencies becomes a possibly cyclic graph.

Basic

The basic typed dependencies use the dependencies defined in section 2, and form a tree structure. Each word in the sentence (except the head of the sentence) is the dependent of one other word. For the sentence, "Bell, a company which is based in LA, makes and distributes computer products", the basic typed dependencies will be:

```
nsubj(makes-11, Bell-1)
det(company-4, a-3)
appos(Bell-1, company-4)
rel(based-7, which-5)
auxpass(based-7, is-6)
rcmod(company-4, based-7)
prep(based-7, in-8)
pobj(in-8, LA-9)
cc(makes-11, and-12)
```

```
conj(makes-11, distributes-13)
nn(products-15, computer-14)
dobj(makes-11, products-15)
```

Collapsed dependencies

In the collapsed representation, additional dependencies are considered, even ones that break the tree structure (turning the dependency structure into a *directed graph*. So in the above example, the following relations will be added:

```
ref(company-4, which-5)
nsubjpass(based-7, which-5)
```

These relations do not appear in the basic representation since they create a cycle with the `rcmod` and `rel` relations. Relations that break the tree structure are the ones taking into account elements from relative clauses and their antecedents (as shown in this example), as well as the controlling (*xsubj*) relations.

Moreover dependencies involving prepositions, conjuncts as well as information about the referent of relative clauses are collapsed to get direct dependencies between content words. This "collapsing" is often useful in simplifying patterns in relation extraction applications. For instance, the dependencies involving the preposition "in" in the above example will be collapsed into one single relation:

```
prep(based-7, in-8)
pobj(in-8, LA-9)
```

will become

```
prep_in(based-7, LA-9)
```

The same happens for dependencies involving conjunction:

```
cc(makes-11, and-12)
conj(makes-11, distributes-13)
```

become

```
conj_and(makes-11, distributes-13)
```

The information about the antecedent of the relative clause (`ref(company-4, which-5)`) will serve to expand the following dependency:

```
nsubjpass(based-7, which-5)
```

becomes

```
nsubjpass(based-7, company-4)
```

In the end the collapsed dependencies that the system gives you for the sentence are:

```
nsubj(makes-11, Bell-1)
det(company-4, a-3)
appos(Bell-1, company-4)
nsubjpass(based-7, company-4)
rel(based-7, which-5)
auxpass(based-7, is-6)
```

```
rcmod(company-4, based-7)
prep_in(based-7, LA-9)
conj_and(makes-11, distributes-13)
nn(products-15, computer-14)
dobj(makes-11, products-15)
```

Note that, in some cases, collapsing relations introduces a slight alteration of the semantics of the sentence. This is the case with PP conjunction as in "Bill went over the river and through the woods": the two prepositions "over" and "through" are conjoined and governed by the verb "went". To avoid disjoint subgraphs when collapsing the relations (preposition and conjunction), examples like this are transformed into VP coordination, which requires making a copy of the word "went" (marked by an apostrophe in the system's output). This gives the following representation, which corresponds to a sentence like "Bill went over the river and went through the woods":

prep_over(went-2, river-5)
prep_through(went-2', woods-10)
conj_and(went-2, went-2')

Collapsed dependencies with propagation of conjunct dependencies

When there is a conjunction, you can also get propagation of the dependencies involving the conjuncts. In the sentence here, this propagation will add two dependencies to the collapsed representation; due to the conjunction between the verbs "makes" and "distributes", the subject and object relations that exist on the first conjunct ("makes") will be propagated to the second conjunct ("distributes"):

```
nsubj(distributes-13, Bell-1)
dobj(distributes-13, products-15)
```

Since this representation is an extension of the collapsed dependencies, it does not guarantee a tree structure.

Collapsed dependencies preserving a tree structure

In this representation, dependencies which do not preserve the tree structure are omitted. As explained above, this concerns relations between elements of a relative clause and its antecedent, as well as the controlling subject relation (*xsubj*). This also does not allow propagation of conjunct dependencies. In our example, the dependencies in this representation will be:

```
nsubj(makes-11, Bell-1)
det(company-4, a-3)
appos(Bell-1, company-4)
rel(based-7, which-5)
auxpass(based-7, is-6)
rcmod(company-4, based-7)
prep_in(based-7, LA-9)
conj_and(makes-11, distributes-13)
nn(products-15, computer-14)
dobj(makes-11, products-15)
```

The table below shows the dependencies for the 4 variants, to facilitate comparison:

basic	**collapsed**	**propagation**	**collapsed tree**
nsubj(makes, Bell)	nsubj(makes, Bell)	nsubj(makes, Bell)	nsubj(makes, Bell)
		nsubj(distributes, Bell)	
det(company, a)	det(company, a)	det(company, a)	det(company, a)
appos(Bell, company)	appos(Bell, company)	appos(Bell, company)	appos(Bell, company)
rel(based, which)	rel(based, which)	rel(based, which)	rel(based, which)
auxpass(based, is)	auxpass(based-7, is)	auxpass(based, is)	auxpass(based, is-)
rcmod(company, based)	rcmod(company, based)	rcmod(company, based)	rcmod(company, based)
	nsubjpass(based, company)	nsubjpass(based, company)	
prep(based, in)			
	prep_in(based, LA)	prep_in(based, LA)	prep_in(based, LA)
pobj(in, LA)			
cc(makes, and)			
	conj_and(makes, distributes)	conj_and(makes, distributes)	conj_and(makes, distributes)
conj(makes, distributes)			
nn(products, computer)	nn(products, computer)	nn(products, computer)	nn(products, computer)
dobj(makes, products)	dobj(makes, products)	dobj(makes, products)	dobj(makes, products)
		dobj(distributes, products)	

5 In practice

In practice, two classes can be used to get the typed dependencies of a sentence using the code in the Stanford parser (downloadable at http://nlp.stanford.edu/software/lex-parser.shtml).

⋆ edu.stanford.nlp.parser.lexparser.LexicalizedParser

If you need to parse texts and want to get different formating options for the parse tree, you should use this class. To get the dependencies, add `typedDependencies` in the `-outputFormat` option. By default, this will give you collapsed dependencies with propagation of conjunct dependencies. If you want another representation, specify it in the `-outputFormatOptions` using the following commands according to the type of dependency representation you want:

`basicDependencies`	basic dependencies
`collapsedDependencies`	collapsed dependencies (not necessarily a tree structure)
`CCPropagatedDependencies`	collapsed dependencies with propagation of conjunct dependencies (not necessarily a tree structure) [this representation is the default, if no option is specified]
`treeDependencies`	collapsed dependencies that preserve a tree structure

You also need to use the `-retainTmpSubcategories` option to get best performance in recognizing temporal dependencies. In the following command, `file.txt` should contain one sentence per line. The `penn` option will also give you the context-free phrase structure grammar representation of the sentences.

Command line example:

```
java -mx100m edu.stanford.nlp.parser.lexparser.LexicalizedParser
```

```
-retainTmpSubcategories -outputFormat "penn,typedDependencies"
-outputFormatOptions "collapsedDependencies" englishPCFG.ser.gz file.txt
```

⋆ **edu.stanford.nlp.trees.EnglishGrammaticalStructure**

If you already have Penn treebank-style trees (whether hand-annotated or as output from another parser), you can use this class to get the Stanford dependencies. Use the `-treeFile` option as shown in the command-line example below. The options to get the different types of representation are as follows:

`-basic`	basic dependencies
`-collapsed`	collapsed dependencies (not necessarily a tree structure)
`-CCprocessed`	collapsed dependencies with propagation of conjunct dependencies (not necessarily a tree structure)
`-collapsedTree`	collapsed dependencies that preserve a tree structure

Command line example:

```
java -mx100m edu.stanford.nlp.trees.EnglishGrammaticalStructure
-treeFile file.tree -collapsedTree -CCprocessed
```

You can also use this class to parse texts, but you will not be able to specify options for the parse tree output on the command line. You will only be able to specify the type of the dependencies. Use the option `-sentFile` instead of `-treeFile` where the file contains one sentence per line. You will need to specify the parser file using the `-parserFile` option. You can get the parse tree by using the `-parseTree` option.
Command line example:

```
java -mx100m edu.stanford.nlp.trees.EnglishGrammaticalStructure
-sentFile file.txt -collapsedTree -CCprocessed -parseTree -parserFile
englishPCFG.ser.gz
```

⋆ **grammarbrowser**

Bernard Bou has written grammarbrowser, a GUI interface to the Stanford Dependencies representation, which allows not only viewing dependencies, but altering their definitions. This is a separate download. It is available at: http://grammarbrowser.sourceforge.net/ .

6 Further references for Stanford Dependencies

The Stanford Dependencies representation was first made available in the 2005 version of the Stanford Parser. Subsequent releases have provided some refinements to and corrections of the relations defined in the original release. The initial written presentation of the Stanford typed dependencies representation was (de Marneffe et al. 2006). A more thorough discussion of the motivations behind the design of the representation appears in (de Marneffe and Manning 2008).

The SD representation has seen considerable use within the biomedical text mining community. It has been used to give a task relevant evaluation scheme for parsers (Clegg and Shepherd 2007, Pyysalo et al. 2007) and as a representation for relation extraction (Erkan et al. 2007, Greenwood and Stevenson 2007, Urbain et al. 2007, Fundel et al. 2007, Clegg 2008, Airola et al.

2008, Giles and Wren 2008, Özgür et al. 2008, Ramakrishnan et al. 2008, Björne et al. 2008). Pyysalo et al. (2007) develops a version of the BioInfer corpus annotated with (a slight variant of) the SD scheme. It is available for download at http://mars.cs.utu.fi/BioInfer/. A small amount of SD gold standard annotated data was separately prepared for the Parser Evaluation Shared Task of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation (see de Marneffe and Manning 2008) and is available at http://www-tsujii.is.s.u-tokyo.ac.jp/pe08-st/, but the BioInfer corpus is the main source of gold-standard SD data which is currently available.

The SD representation has also been used in other domains, such as for extracting opinions, sentiment, and relations (Zhuang et al. 2006, Meena and Prabhakar 2007, Banko et al. 2007, Zouaq et al. 2006; 2007, Chaumartin 2007, Kessler 2008). Several groups used it as a representation in recent PASCAL Recognizing Textual Entailment (RTE) challenges (Adams et al. 2007, Blake 2007, Chambers et al. 2007, Harmeling 2007, Wang and Neumann 2007).

References

Rod Adams, Gabriel Nicolae, Cristina Nicolae, and Sanda Harabagiu. Textual entailment through extended lexical overlap and lexico-semantic matching. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 119–124, Prague, June 2007. URL http://www.aclweb.org/anthology/W/W07/W07-1420.

Antti Airola, Sampo Pyysalo, Jari Björne, Tapio Pahikkala, Filip Ginter, and Tapio Salakoski. A graph kernel for protein-protein interaction extraction. In *Proceedings of BioNLP 2008: Current Trends in Biomedical Natural Language Processing (ACL08)*, 2008.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. Open information extraction from the web. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, 2007.

Jari Björne, Sampo Pyysalo, Filip Ginter, and Tapio Salakoski. How complex are complex protein-protein interactions? In *3rd International Symposium on Semantic Mining in Biomedecine*, 2008.

Catherine Blake. The role of sentence structure in recognizing textual entailment. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 101–106, Prague, June 2007. URL http://www.aclweb.org/anthology/W/W07/W07-1417.

Nathanael Chambers, Daniel Cer, Trond Grenager, David Hall, Chloe Kiddon, Bill MacCartney, Marie-Catherine de Marneffe, Daniel Ramage, Eric Yeh, and Christopher D. Manning. Learning alignments and leveraging natural logic. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 165–170, Prague, June 2007. URL http://www.aclweb.org/anthology/W/W07/W07-1427.

François-Régis Chaumartin. UPAR7: A knowledge-based system for headline sentiment tagging. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007)*, pages 422–425, 2007.

Andrew B. Clegg. *Computational-Linguistic Approaches to Biological Text Mining*. PhD thesis, School of Crystallography, Birkbeck, University of London, 2008.

Andrew B. Clegg and Adrian J. Shepherd. Benchmarking natural-language parsers for biological applications using dependency graphs. *BMC Bioinformatics*, 8:24, 2007.

Marie-Catherine de Marneffe and Christopher D. Manning. The Stanford typed dependencies representation. In *COLING Workshop on Cross-framework and Cross-domain Parser Evaluation*, 2008. URL http://nlp.stanford.edu/pubs/dependencies-coling08.pdf.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. Generating typed dependency parses from phrase structure parses. In *5th International Conference on Language Resources and Evaluation (LREC 2006)*, 2006. URL http://nlp.stanford.edu/pubs/LREC06_dependencies.pdf.

Gunes Erkan, Arzucan Ozgur, and Dragomir R. Radev. Semi-supervised classification for extracting protein interaction sentences using dependency parsing. In *Proceedings of the 2007*

Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), 2007.

Katrin Fundel, Robert Küffner, and Ralf Zimmer. RelEx – relation extraction using dependency parse trees. *Bioinformatics*, 23, 2007.

Cory B. Giles and Jonathan D. Wren. Large-scale directional relationship extraction and resolution. *BMC Bioinformatics*, 9(Suppl 9):S11, 2008. doi: 10.1186/1471-2105-9-S9-S11. URL http://www.biomedcentral.com/1471-2105/9/S9/S11.

Mark A. Greenwood and Mark Stevenson. A semi-supervised approach to learning relevant protein-protein interaction articles. In *Proceedings of the Second BioCreAtIvE Challenge Workshop, Madrid, Spain*, 2007.

Stefan Harmeling. An extensible probabilistic transformation-based approach to the third recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 137–142, Prague, June 2007. URL http://www.aclweb.org/anthology/W/W07/W07-1423.

Jason S. Kessler. Polling the blogosphere: a rule-based approach to belief classification. In *International Conference on Weblogs and Social Media*, 2008. URL http://www.cs.indiana.edu/ jaskessl/icwsm.pdf.

Arun Meena and T. V. Prabhakar. Sentence level sentiment analysis in the presence of conjuncts using linguistic analysis. In *Advances in Information Retrieval*, volume 4425 of *Lecture Notes in Computer Science*. Springer, 2007.

Arzucan Özgür, Thuy Vu, Günes Erkan, and Dragomir R. Radev. Identifying gene-disease associations using centrality on a literature mined gene-interaction network. *Bioinformatics*, 24(13):i277–i285, 2008. doi: 10.1093/bioinformatics/btn182. URL http://bioinformatics.oxfordjournals.org/cgi/reprint/24/13/i277.

Sampo Pyysalo, Filip Ginter, Katri Haverinen, Juho Heimonen, Tapio Salakoski, and Veronika Laippala. On the unification of syntactic annotations under the Stanford dependency scheme: A case study on BioInfer and GENIA. In *Proceedings of BioNLP 2007: Biological, translational, and clinical language processing (ACL07)*, 2007.

Cartic Ramakrishnan, Pablo N. Mendes, Shaojun Wang, and Amit P. Sheth. Unsupervised discovery of compound entities for relationship extraction. In *16th International Conference on Knowledge Engineering: Practice and Patterns (EKAW 2008)*, pages 146–155, 2008.

Jay Urbain, Nazli Goharian, and Ophir Frieder. IIT TREC 2007 genomics track: Using concept-based semantics in context for genomics literature passage retrieval. In *The Sixteenth Text REtrieval Conference (TREC 2007) Proceedings*, 2007.

Rui Wang and Günter Neumann. Recognizing textual entailment using sentence similarity based on dependency tree skeletons. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 36–41, Prague, June 2007. URL http://www.aclweb.org/anthology/W/W07/W07-1406.

Li Zhuang, Feng Jing, Xiao yan Zhu, and Lei Zhang. Movie review mining and summarization. In *Proc. ACM Conference on Information and Knowledge Management (CIKM)*, pages 43–50, 2006.

Amal Zouaq, Roger Nkambou, and Claude Frasson. The knowledge puzzle: An integrated approach of intelligent tutoring systems and knowledge management. In *Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2006)*, pages 575–582, 2006.

Amal Zouaq, Roger Nkambou, and Claude Frasson. Building domain ontologies from text for educational purposes. In *Proceedings of the Second European Conference on Technology Enhanced Learning: Creating new learning experiences on a global scale*, 2007.