

Toward the Interoperability of Language Resources

July 13-15, in conjunction with the 2007 LSA Summer Institute



Interoperability of Syntactic and Semantic Annotation Schemes

Marc Verhagen and James Pustejovsky
Brandeis University

We propose the following requirements for the interoperability of syntactic and semantic annotations:

1. Each annotation scheme has its own philosophy and is independent from the other annotations.
2. Simple tree-based and one-directional merging of annotations is useful for visualization of overlap between schemes (see section 1).
3. Simple, generally applicable methods should be used to define relations between schemes.

1. Simple Tree-Based Merging

The Penn Treebank annotation can be selected as a starting point. Data from other annotations can be attached at tree nodes. If a tag from another annotation does not correspond to a TreeBank node, then simple heuristics should be used to find a node whose extent is closest to the one requested. Alternatively, one could select a head. The attached node however will be marked if its original extent does not line up with the extent at the tree node. This merging is one-directional since no attempt is made to change the shape of the tree.

2. Interoperability of Annotations

Simple and extendable interoperability can be put in place using three components.

1. All annotations need to be embedded in a unified environment in which each annotation has its own namespace in a merged annotation. This unified environment would take care of some basic functionality. For example, given a tag from one annotation scheme, there should be a method that returns tags from other annotation schemes that have the same text extent, or that have an overlap in text extent.
2. Each annotation defines an interface. For example, an annotation scheme that encodes predicate-argument structure, like PropBank and NomBank, should define methods like `giveArgumentsOfRelation(pred)` and `giveRelationOfArgument(arg)`.
3. Specific interactions can be built using the unified environment and the specified interfaces of each annotation scheme. For example, an application where an named



entity is followed through time depends on three kinds of information: identification of named entities, predicate-argument structure, and temporal relations. Each of these derive from a separate annotation scheme. A use case can be built using the interfaces for each annotation: the named entity annotation would return the text extents of the named entity, the predicate-argument annotation would return the predicates that go with an named-entity argument, and the time annotation would return the temporal relations between all those predicates.

This kind of interoperability does not require an elaborate representational structure allowing the interaction. Rather, it relies on independent annotation schemes with interfaces to the outside world that interact given a specific use case. The more annotations there are, the more interactions can be defined.

