# The Brandeis Annotation Tool

**Marc Verhagen**

Computer Science Department
Brandeis University
Waltham, USA
marc@cs.brandeis.edu

**Abstract**

The Brandeis Annotation Tool is a web-based text annotation tool that is centered around the notions of layered annotation and task decomposition. It allows annotations to refer to other annotations and to take a complicated task and split it into easier subtasks. The web-interface connects annotators to a central repository for all data and simplifies many of the housekeeping tasks while keeping requirements at a minimum (that is, users only need an internet connection and a well-behaved browser). BAT has been used mainly for temporal annotation, but can be considered a more general tool for several kinds of textual annotation.

## 1. Introduction

The Brandeis Annotation Tool (BAT) is a web-based text annotation tool that is centered around the notions of layered annotation and task decomposition. Layered annotation allows annotations to refer to other annotations, task decomposition allows taking a complicated task and splitting it into many easier subtasks. Web-based annotation provides a central repository for all data and simplifies many of the housekeeping tasks. This combination of features distinguishes BAT from existing tools like Callisto and GATE.[1]

BAT's first incarnation, in 2007, was originally motivated out of desperation in the context of the Tempeval evaluation task (VGS$^+$07), when quick task-specific annotation was needed in order to release the data sets in time. BAT has so far primarily been used for temporal annotation, but lately the focus has shifted towards more general annotation and over the last year BAT has been re-tooled to closely follow the spirit of the Linguistic Annotation Format (LAF). The next section outlines the basic principles of the Linguistic Annotation Format. Section 3 describes the main functionalities of BAT and gives an idea of where BAT differs from LAF. The conclusion gives some pointers on further expansion of functionality.

## 2. The Linguistic Annotation Format

The Linguistic Annotation Format (LAF) is a standard for linguistic annotation developed by the International Standard Organization (IR06; IS07). It is intended to provide guidance on the basic principles for representing linguistic annotation schemes that form one of the primary bases for language resource management. Some of the main principles of LAF are: (i) annotations are separated from the data they annotate (that is, LAF requires stand-off annotation), (ii) annotation structure and content are separated, and (iii) mappings between annotation occur via a pivot format.

Annotations can consist of an arbitrary number of sub-annotations called layers. Annotation layers can refer to any set of nodes created in already existing annotation layers. In this framework, arbitrarily complex annotations can be built using several layers, including layers that define alternative base segmentations or morpho-syntactic analyses (see Figure 1).

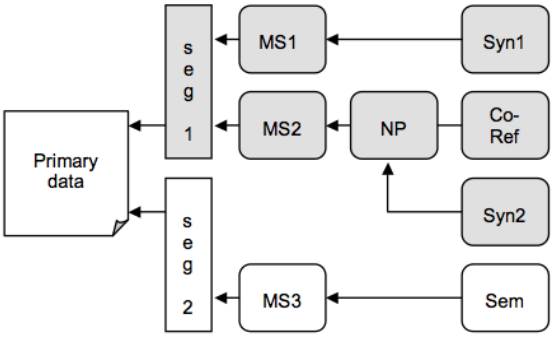


Figure 1: LAF Annotation Layers

The LAF data model for annotations comprises a directed graph referencing regions of primary data as well as other annotations, in which nodes are labeled with feature structures providing the annotation content. The graph is initiated by creating virtual nodes between all characters in the primary data. Then leaf nodes can be created by referring to spans defined by virtual nodes. For example, for the string *"The clock struck ten."*, we can create leaf nodes as follows:[2]

```
<edge id="e1" from="0" to="3"/>
<edge id="e2" from="4" to="9"/>
<edge id="e3" from="10" to="16"/>
<edge id="e4" from="17" to="20"/>
<edge id="e5" from="20" to="21"/>
```

Typically, this first layer of annotation is referred to as the base segmentation. Other annotations can be added on top of this annotation. Below is a node from a layer defining lemmatization and parts of speech:

[1] Available at `http://callisto.mitre.org/` and `http://gate.ac.uk/`.

[2] The example is taken from (IR06).

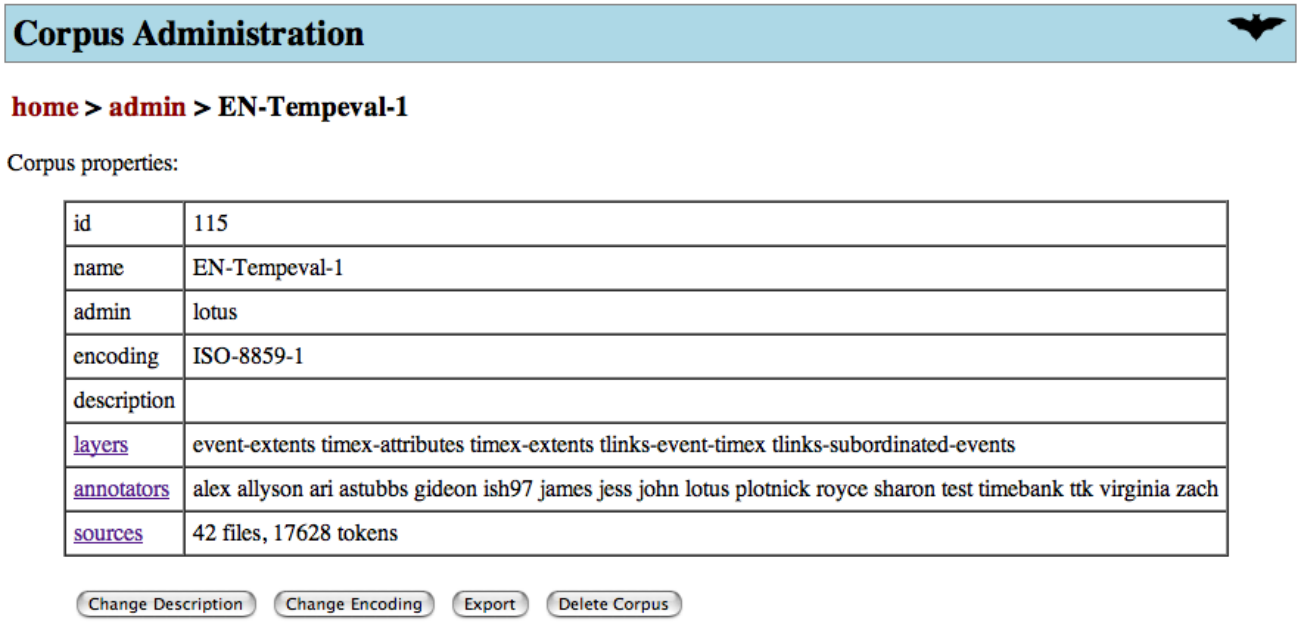


Figure 2: Administrator main page


Figure 3: Layer administration in BAT

```
<edge id="t2" ref="e2">
  <fs type="token">
    <f name="lemma" sVal="clock"/>
    <f name="pos" sVal="NN"/>
  </fs>
</edge>
```

GrAF is an XML serialization that instantiates the abstract graph of LAF. It functions as the pivot format to and from which annotations compliant with LAF can be mapped to. It is simple to combine several annotations into one GrAF, as long as stand-off annotation was used for all annotations.

## 3. BAT Functionality

Like any self respecting annotation tool, BAT tries to make annotation as intuitive as possible while providing a rich enough set of features. Unlike many tools available, BAT attempts to do some of the heavy lifting that generally falls into the hands of the corpus administrator.

### 3.1. Administration

The BAT Administrator pages provide functionality for creating corpora, importing source data, creating annotator logins, assigning tasks to annotators, defining layers, uploading layer data, viewing inter-annotator agreement statistics, and viewing general progress of annotators. The sources link in Figure 2 allows the administrator to upload the base segmentation as a tab-separated file with filename, token offset and token string columns. One current difference between LAF and BAT is that BAT only allows one base segmentation per corpus.

The most interesting part may be the possibility to define and populate layers. Layers are created by uploading a definition file that contains some basic facts about the layer

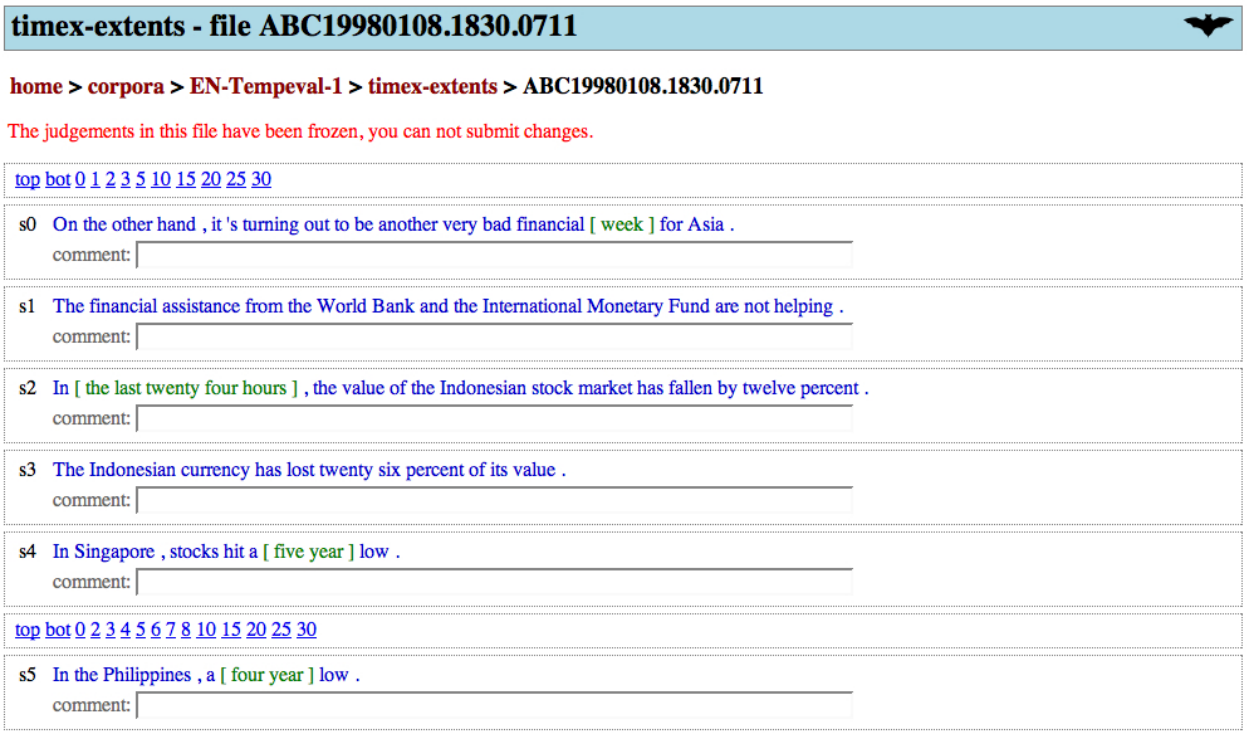


Figure 4: Extent annotation in BAT

like layer name, layer type and layer domain. The domain of a layer is the set of layers that it annotates over (that is, the arrows in Figure 1). In the case shown in Figure 3, the domain is all nodes labeled *timex3* in the timex-extents layer. The default domain is the base segmentation.

BAT includes the layer types *extent* and *attribute*, reflecting the LAF difference between nodes and associated features structures. For attribute layers, like the one shown in Figure 3, attribute names and their possible values will need to be defined as well. BAT uses a third layer type named *relation* even though under LAF there is no theoretical difference between annotating extents and relations, they are both concerned with adding nodes. BAT departs from LAF conventions here so it can align more with annotation practice, where there is a clear difference between marking extents, even if on top of other extents, and marking relations between extents.

Note that the strict LAF/BAT distinction between nodes (extents or relations) and associated feature structures (attributes) may annoyingly split what one considers to be one task into two task, for example when the task involves selecting named entities with their types. Therefore, BAT will in the future introduce mixed tasks where both extents and attributes are annotated in one fell swoop.

There are advantages however to separating extent and attribute annotation. One is that in some cases the extents may be created by external tools. For example, for Tempeval2, one of the tasks is to annotate the temporal relation between events in subordination relations, as in *"The spokesperson said his country was provoked by the enemy."*. In this case, the syntactic relation between the events *said* and *provoked* could be created by a parser or taken from another annotation like the Penn Treebank. Node pairs like these can be uploaded as an extent layer and the annotator can then concentrate on defining the attributes to the relation (that is, whether one event is before, during or after the other).[3]

### 3.2. Annotation

When annotators log in, they see all the layers in which files have been assigned to them as annotation tasks. Clicking a layer name takes the annotator to a list of files in the layer that were assigned to her. The list also indicates the status of all these files. Files are either assigned, submitted or frozen. In the second case, an assigned file has been previously annotated and submitted by the annotator; in the third case, a submitted file was frozen and can therefore not be edited anymore (although the annotator can go in and see the annotation).

Clicking on a file name in the list of assigned files will lead the annotator to a screen as in Figure 4. Extents are defined by clicking the first and last tokens of the extents.[4] Annotators can add and delete extents and change the boundaries of extents.

Figure 5 shows a screenshot of event attribute annotation with an example taken from current annotation of events performed at Brandeis University. For any kind of attribute annotation, the extents are highlighted in the sentence and the attributes defined in the layer definition are displayed below the sentence. The tool uses the layer definition to

[3]This, by the way, is what motivated the very first version of BAT since it made it feasible to quickly annotate large numbers of temporal links for Tempeval.

[4]In cases when extents are always deemed to be one token, something that can be specified in the layer definition, this is optimized to needing just one click.

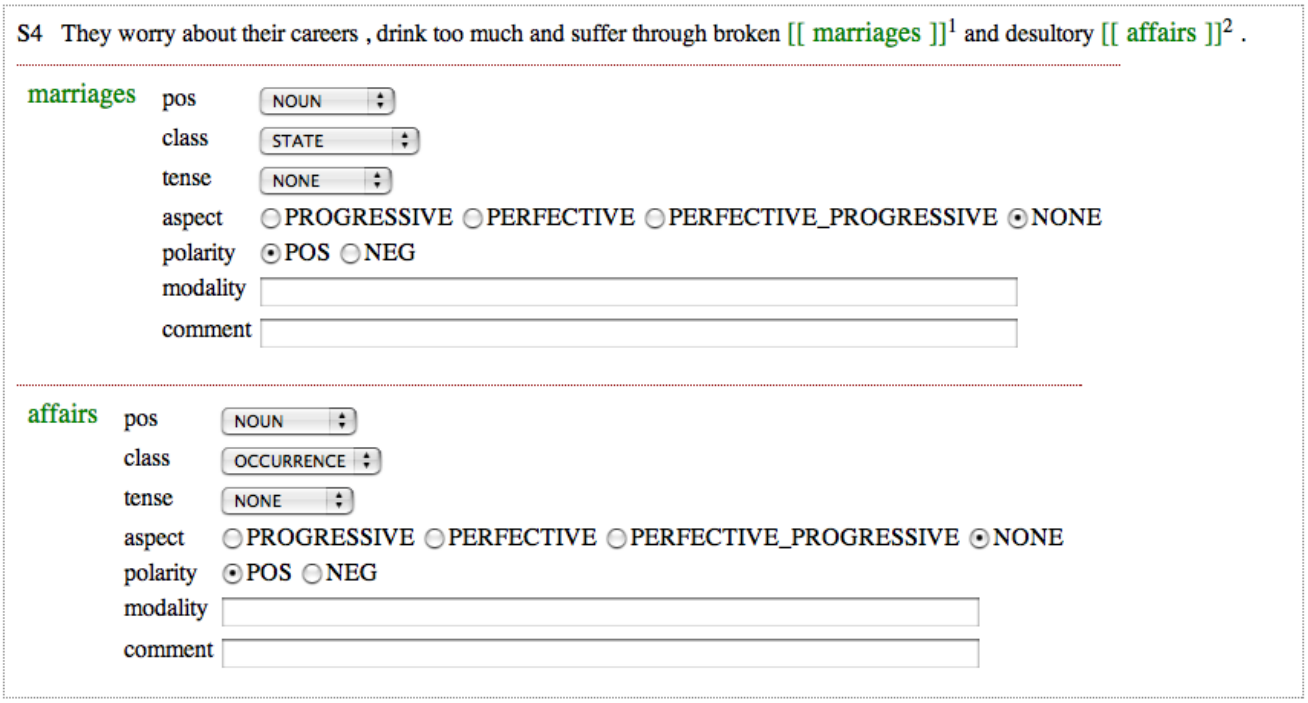


Figure 5: Attribute annotation in BAT

create the interface. In particular, the list of values listed for each attribute (see Figure 3) determines whether the user is presented with a radio button, pulldown list or a general text input field. Currently, radio buttons are used if there are four or less possible values and pulldown lists if there are more. Text inputs are used if the value of the attribute is not constrained. Finally, defaults spelled out in the layer definition will be preselected.

An example of relation annotation is shown in Figure 6. Relation layers are defined on top of one or two extent layers. In the current example, the relation layer *tlinks-subordinated-events* is defined on top of the event extents layer. The layer definition for *tlinks-subordinated-events* is printed below:

```
name = tlinks-subordinated-events
type = relation
domain = event-extents:event
label = tlink
attr.relationType = string []
```

In addition, the layer definition specifies the label of relation and what attributes are defined for this label. In the case above, the label is *tlink* which has one attribute named *relType* whose possible value is not constrained.
As with attribute layers, the interface will take the layer definition and act accordingly. Initially, all it will show is the document with all events highlighted. The annotator can click an event, after which it will turn red, then click another event. After the second click a little widget will materialize towards the right of the event that occurs first in the text. Now the annotator can fill in a value that describes the relation between the two events. Relations can be removed by clicking the little red button on the left of the relation.

Relation functionality is not complete yet since currently only one attribute can be defined for each label. For example, for the *tlinks-subordinated-events* layer the only attribute defined is the relation type of the temporal link (the name of the attribute is not shown because confusion is not possible). The more general approach though would be to allow a set of attributes and thereby stay in lock step with attribute annotation and the LAF philosophy.

### 3.3. Adjudication phase

The administrator can assign as many annotators to each file as she desires, typically using one to three annotators. For each file in an annotation layer, the administrator can also assign one judge. Judges will see an interface very similar to the annotators, but with some major differences:

- the judge will see all annotations of the annotators but cannot change them
- attributes submitted by the annotators will be presented in a more compact form
- cases where annotators disagree are highlighted in red so the judge can easily focus on them
- a judge cannot submit results until all annotators have frozen their judgements

When the annotators agree, their judgments will be copied as defaults to the judge's extents or attributes. Ideally, if the annotators agree on everything, all the judge needs to do is to glance over the results and hit the submit button.

### 3.4. Technical details and current status

BAT is implemented in PHP and Javascript, the data are stored in a MySQL database. The code relies on PHP4

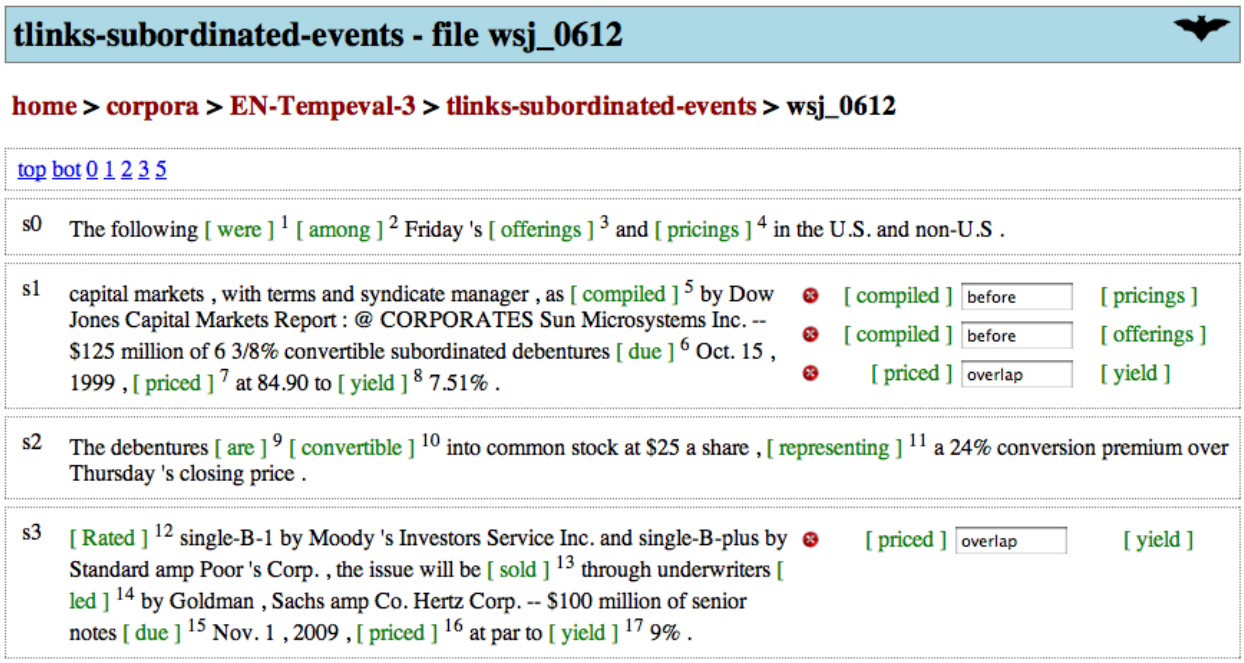


Figure 6: Relation annotation in BAT

or higher and MySQL 4 or higher. The tool has been installed on various Linux servers and on Mac OSX 10.4. User requirements are limited, but an internet connection is required. Most browsers are known to work without problems, with the notable exception of many version of Internet Explorer.[5]

The tool has been used for temporal annotation in five languages for the Tempeval2 task scheduled for Semeval-2010 (VGS+09). For temporal annotation, BAT has now fully supplanted the old Alembic and Tango tools that were used while creating Timebank (DFG+03; VKMP06; DAH+97). The current version of BAT is available at `http://timeml.org/site/bat/`. Anyone can get an administrator account simply by following the instructions in the manual.

## 4. Conclusion and Future Work

BAT is a generic annotation tool that can be used with little overhead for the annotator and that gives ample control to the corpus administrator. Annotation tasks can be defined to fit the needs of the annotation and collecting results from annotators is a simple matter of clicking a button in the administrator interface. Likewise, progress can be easily monitored and inter-annotator agreement statistics are readily available.

It should be noted that there are things that are hard, if not impossible, to annotate with BAT. For example, BAT is not set up to be a tool for creating syntactic structure. It is possible, but in a very laborious and roundabout way. This is due to the interface, which focusses on the text, extents selected in the text, and attributes associated with the extents. It does not do well when extents overlap or are embedded in each other, as with syntactic structure.

Another limitation is that any annotation starts with a base-segmentation and that this segmentation determines what can be selected in the tool. It is not possible to simply swipe a couple of characters and than create a label. This is problematic for morphology rich languages, where the base-segmentation now has to stipulate that some morphemes are actually tokens.

There is a large list of requested features and wanted improvements, as well as an even larger list of minor bugs and annoyances. Here are some of the most useful and interesting ones:

- introduce a distinction between morphemes and tokens, allowing annotators to select segments of words
- introduce a new layer type that combines extent annotation and attribute annotation
- allow unlimited attributes for relations
- supply convenience scripts to deal with corpus import and export
- create a task repository with some standard tasks
- optimize code that calculates statistics (this is now one of the major bottlenecks that make it hard to have corpora lager than 50,000 tokens)
- add timestamps to judgements so we have a temporal trail of annotations
- allow users more control over the interface
- provide example corpora that people can play with without registering

[5] Although this wasn't checked thoroughly for the latest Internet Explorer versions.

Finally, one potential problem to many administrators could be that all the data and annotations live on a server maintained by strangers. There is really no reason why BAT should not be installed elsewhere and the source code will be made publicly available under a reasonable license (Creative Commons or GPL) once its level of maturity has been pushed up a bit.

## 5. Acknowledgments

The first version of BAT was developed under the ARDA AQUAINT grant NBCHC040027, "The TARSQI Toolkit: Temporal Awareness and Event-based Chronicles", and the NSF-CRI grant 0551615, "Towards a Comprehensive Linguistic Annotation of Language". BAT functionality is now further developed in the context of the NSF-INT-0753069 project "Sustainable Interoperability for Language Technology (SILT)", funded by the National Science Foundation.

Many people were involved in creating BAT. Thanks to Alex Plotnick for much of the Javascript coding and to Royce Stubbs for implementing the inter-annotator agreement statistics. And many thanks to all the people who provided input in the design phase and who actually used BAT in its early stages and who kept harassing me to improve the sub-optimal features known as bugs. Listed in alphabetical order: Valentina Bartalesi, Tommaso Caselli, Lotus Goldberg, Allyson Ettinger, Nancy Ide, Seohyun Im, Jessica Moszkowicz, Alex Plotnick, James Pustejovsky, Roser Saurí, Rachele Sprugnoli, Nianwen Xue, and Yuping Zhou. Of course, all bone-headed design decisions remain firmly my responsibility.

## 6. References

David Day, John Aberdeen, Lynette Hirschman, Robyn Kozierok, Patricia Robinson, and Marc Vilain. Mixed-Initiative Development of Language Processing Systems. In *Fifth Conference on Applied Natural Language Processing Systems*, pages 88–95, Washington D.C., U.S.A., 1997.

David Day, Lisa Ferro, Robert Gaizauskas, Patrick Hanks, Marcia Lazo, James Pustejovsky, Roser Saurí, Andrew See, Andrea Setzer, and Beth Sundheim. The TimeBank Corpus. *Corpus Linguistics*, March 2003.

Nancy Ide and Laurent Romary. Representing Linguistic Corpora and Their Annotations. In *Proceedings of the Fifth Language Resources and Evaluation Conference (LREC)*, Genoa, Italy, 2006.

Nancy Ide and Keith Suderman. GrAF: A graph-based format for linguistic annotations. In *Proceedings of the Linguistic Annotation Workshop*, pages 1–8, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Graham Katz, and James Pustejovsky. SemEval-2007 Task 15: TempEval Temporal Relation Identification. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 75–80, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Jessica Moszkowicz, and James Pustejovsky. The TempEval challenge: identifying temporal relations in text. *Language Resources and Evaluation*, 43(2), 2009.

Marc Verhagen, Robert Knippen, Inderjeet Mani, and James Pustejovsky. Annotation of Temporal Relations with Tango. In *Proceedings of LREC 2006*, Genoa, Italy, 2006.