ISO LAF/GrAF and the Open American National Corpus Model for Representing and Distributing Multi-layer Annotated Corpora

Plus an Exercise in Designing Annotation Schemes

Nancy Ide
Department of Computer Science
Vassar College

EBRALC Workshop • Sao Carlos, Brazil • September, 2012
Interoperability for NLP

• Much agreement that interoperability is a necessary next step for NLP
• Why?
  – Need an interoperable infrastructure for data, annotations, and tools
    • Enormous waste of effort adapting resources to different tools over the past 20+ years
  – Need to combine/merge/use together resources in different formats, different media/modalities, different languages to foster NLP development
Examples of problems

• US project funded by NSA involving multiple annotation projects annotating the same data (the Language Understanding (LU) corpus or “Boyan 10K”)
  – Different phenomena
  – Different formats

• Goal was to study inter-level interactions among linguistic phenomena

• Found that it was impossible to combine
  – Mostly due to alternative tokenizations
  – But also some information loss of implicit information in annotations when tried to put in common format
Examples

• ULA Project
  – Data annotated by multiple projects for different phenomena, each using their own scheme
  • TimeML, Penn Treebank, Penn Discourse Treebank, PropBank, NomBank, Opinion
  • Format (syntactic) incompatibility
    – Difficult to merge the annotations or otherwise use them together
  • Semantic incompatibility
    – Different categories, in general not mappable
Examples

• Wildly different representation formats
  – PTB in-line bracketed
  – PropBank …
  – FrameNet “stand-off” XML
  – Opinion Bank modified GATE output
  – …and many more

• Different tokenizations
Examples

• Difficult/impossible to link resources of different types
  – E.g. lexicon and corpus

• Conversion of one resource to another for use in different tool sometimes impossible because critical information embedded in in-house tool

• Etc.
Interoperability defined

- Broadly: a measure of the degree to which diverse systems, organizations, and/or individuals are able to work together to achieve a common goal
- For computer systems: defined in terms of syntactic interoperability and semantic interoperability
Syntactic interoperability

• Relies on specified data formats, communication protocols, and the like to ensure communication and data exchange
• Systems involved can process the exchanged information, but no guarantee that the interpretation is the same
Semantic interoperability

• Two systems have the ability to automatically interpret exchanged information meaningfully and accurately in order to produce useful results via deference to a common information exchange reference model.

• The content of the information exchange requests are unambiguously defined: what is sent is the same as what is understood.
Interoperability for language resources

• Trend in the field is to specify
  – An abstract data model for structuring linguistic data to which syntactic realizations can be mapped (syntactic interoperability)
  – A mapping to a set of linguistic data categories that communicate the information (linguistic) content (semantic interoperability)

• Efforts include
  – ISOCat
  – OLiA
Syntactic Interoperability

• Requirements
  – A **Standard Representation Format** that provides for syntactic and semantic interoperability
    • Several *de facto* best practices:
      – Independence of source (stand-off annotations)
      – Primary language data should be regarded as read-only, annotations in stand-off form
  – Conformance to a **Common Representation Model**
    • Directed graph has emerged as the standard abstract model for linguistic annotations
    • Underlies Annotation Graphs, XML and RDF models, UIMA CAS, and ISO Linguistic Annotation Framework
  – **Standard Encoding**
    • UNICODE (e.g., UTF-8 for primary data in western languages)
Syntactic interoperability

- Use of a pivot format addresses syntactic interoperability
  - Enables merging
  - Enables easy transduction to other formats

Original format → pivot → Merged

Conversion tools:
- In-line XML
- CONLL IOB
- NLTK
- GATE
- UIMA
- …
Linguistic Annotation Framework

- “LAF”
- ISO 24612
- Developed in ISO TC37 SC4 WG1
- A general framework for representing all types of linguistic data and annotations
- Based on 20+ years of efforts in the community to find a common representation mechanism
Linguistic Annotation

• Data is annotated for more than one type of linguistic phenomenon
  – Morpho-syntax, syntax, word senses, semantic roles, discourse structure, entities, events…
  ➔ This leads to the need to enable merging annotation information of different types

• Many different annotation schemes are used for the same phenomenon
  ➔ This leads to the need to handle/merge annotations of the same type
  – The problems of combining information from FrameNet, PropBank, VerbNet, etc. are a small example
Historically, there have been many standardization efforts, none fully successful

- TEI, Corpus Encoding Standard (CES, XCES) (representation)
- EAGLES/ISLE (content)
- De facto standards such as WordNet for sense tags
Biggest Problem

• Very often no separation between
  – Annotation content categories
    • Semantic interoperability
    • E.g., sense tags, syntactic categories, POS categories, etc.
  – Annotation structure
    • The syntax for conveying the information and specifying relations among categories, features, etc.

Essential to separate these two
Content Categories

• Most people think only of this when considering annotation standards
• Very difficult to standardize
• Some *de facto* standards
  – Penn Treebank POS categories for English
  – Penn Treebank bracketing for representation *(ugh!)*
  – WordNet synsets for sense tagging
  – …
Annotation Structure

• Standardization of annotation representation has focused on formats (e.g. XML) rather than representing the relevant relations among content categories

• Rather than a prescribed format (e.g. XML) we need to focus on means to
  – Represent any annotation type
  – Enable trivial mapping of one format to another
  – Enable merging of different annotation types
Why Is This Difficult?

• Because annotation content is often not distinguished from annotation structure, *annotation content is often conveyed implicitly by the structure*
  – This can make it difficult (or impossible) to extract the content

• Example:
  – Use same structural mechanism to convey different relations
    • E.g. nesting in LISP-like format:
      – A set of alternatives?
      – An ordered list?
    • Only way to know is to understand the content categories!
      \[(X \ ((Y) \ (Z)))\]
Other problems

• Inconsistency
  – Some content represented explicitly, some represented via structure
Example: Syntactic Annotation

Constituency analysis

Penn Treebank

```
((S  (NP-SBJ-1 Paul)
    (VP intends)
    (S (NP-SBJ *-1)
      (VP to
        (VP leave
          (NP IBM))))))
```

Dependency analysis

Carroll, Minnen, and Briscoe

```
subj(intend,Paul,_) 
xcomp(intend,leave,to) 
subj(leave,Paul)  
dobj(leave,IBM,_) 
```
What’s Going On?

((S (NP-SBJ-1 Paul)
 (VP intends)
 (S (NP-SBJ *-1)
 (VP to
   (VP leave)
     (NP IBM )))
).

Category information: ‘NP’
Relational information: ‘SBJ’ (implicit head)
Node identification: 1

Category information: ‘NP’
Relational information: ‘SBJ’ (implicit head)
Node reference *-1

Category information: ‘NP’
Relational information: implicit ‘OBJ’
subj(intend, Paul, _)
xcomp(intend, leave, to)
subj(leave, Paul, _)
dobj(leave, IBM, _)

Explicit relation

Head (implicit in position)

Dependent (implicit in position)

Category-specific information (may be absent)
What are the Underlying Principles?

- These annotation schemes include
  - Relational information
    - Relations implicit in the hierarchy of bracketed components
    - Explicitly specified relations
  - Category information
    - Syntactic category (noun phrase, prepositional phrase…)
    - Role (subject, object…)
    - Thematic role (agent, patient…)

- Inconsistent!
  - This makes it difficult to extract the information or map to another scheme via automatic means
Another Problem

• Often, annotations are in the same document/file as the primary data

**Penn Treebank**

```
<MUC-7>
<TEXT>
<p>
<s>
   <lex pos=DT>The</lex>
   <lex pos=NNP>Federal</lex>
   <lex pos=NNP>Aviation</lex>
   <lex pos=NNP>Administration</lex>
   <lex pos=VBD>underestimated</lex>
   <lex pos=DT>the</lex>
   <lex pos=NN>number</lex>
   <lex pos=IN>of</lex> ...

((S (NP-SBJ-1 Jones)
  (VP followed)
  (NP him)
  (PP-DIR into
   (NP the front room)))
 ,
  (S-ADV (NP-SBJ *-1)
   (VP closing
    (NP the door)
    (PP behind
     (NP him)))))))))
```
Why is this a bad idea?

- Hard to have several different types of annotation for the same data
- Hard/impossible to have alternative annotations of same type
- Hard to maintain
- Disrupts text, may lose information (e.g., punctuation, spacing)
Solutions

• A number of ideas and practices are converging

• ISO TC37 SC4 groups attempting to reflect this in a number of standards for representing LRs

• WG1: General framework for representing LRs
  – Linguistic Annotation Framework (LAF)
LAF Approach

• Develop a common, abstract model that can capture all types of annotation information, regardless of the physical encoding
• Develop a generic, XML instantiation of the model, to and from which specific formats can be mapped
• Define a common set of data categories, for reference and use by annotators
Simple Principles

- Separation of data and annotations
- Separation of user annotation formats and the exchange (“pivot”) format
- Separation of annotation structure and content in the pivot format
  - Separation of syntactic and semantic properties
Collaboration through standards

• **Principles**
  – Don’t impose a single syntactic format
  – Promote best practices through tools
  – Make best practices known
    • Use an obvious de facto standard *OR*
    • Design a task to determine which emerging standard is most interoperable *OR*
    • Create well-formed bridges rather than require a specific programming language, format, etc.
Collaboration through standards

For both standards bodies and individual developers:

• Standards must generalize over multilingual and multimodal data
• Standards should be tested on multilingual and multimodal data
Collaboration through standards

- Inviolate source
  - “Do no harm” to any layer you are annotating
    - If you modify, add as standoff annotation
  - Primary data inviolate
    - Allows multiple annotations of the same type over same data, as well as multiple annotations of different types
LAF Abstract Model

• Annotations represented as a graph of feature structures
  – Nodes are labeled with feature structures containing the annotation content
  – Edges link nodes to locations in primary data or other annotations

• Any format instantiating the model can be trivially mapped to another format

• Map to a “pivot format” (XML) that instantiates the model
  – GrAF, a generic graph representation
Main Idea

Pivot Format

Format A
Format B
Format C
Format D
Format E
Format F

Combined
Stand-off Annotation

• Language data is regarded as “read-only” and contains no annotations
• All annotations are contained in stand-off documents linked to the primary data or other annotation documents
• Use of stand-off annotations is becoming the standard practice in the field
Segmentation

- Minimal unit of granularity
- In text, points to virtual nodes between characters in primary data
  - Image: Cartesian coordinates
  - Video: Frame, coordinates
  - Audio: time stamps
- No associated annotation content (at this level)
- Set of linearly ordered regions
Segmentation

• May have multiple segmentations over the same data
  – E.g., multiple tokenizations that refer to the requisite “quarks” of which the different tokens are composed

\[
\text{(can’t)}
\]

\[
\text{can} \quad \text{R1} \quad \text{R2} \quad \text{‘t} \quad \text{R3}
\]

\[
\text{ca} \quad \text{R1} \quad \text{n’t} \quad \text{R2} \quad \text{R3}
\]
User formats

• Users may use any format for annotations
• Requirement: user’s annotation format is automatically mappable to the abstract data model (ADM) instantiated by the pivot format
  — Annotation information in the original format must be made explicit in the pivot format representation
• The various ISO formats MAF, SynAF, ISO-TimeML, ISO Space, etc. are user formats directly mappable to the LAF ADM
• In principle, users will never deal directly with, or even see, the pivot format
Mapping to the Pivot Format

Penn Treebank
((S
  (NP-SBJ-1 Paul)
  (VP intends)
(S
  (NP-SBJ */-1)
  (VP to
  (VP leave)
  (NP IBM )))
).

Result

```
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6
  1                   2
|Paul| |intends| |to| |leave| |IBM|
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6
```

```
NP  VP  VP  NP
   ↑   ↑   ↑
SUBJ SUBJ S   S
```

```
Multiple Representations of same Information

My dogs have fleas.

Category : NNP
Lemma : dog
Orthography : dogs

<w cat="NNP" lemma="dog">dogs</w>

<seg target="xptr(substring(/p/s[1]/text(),1,5))" lex="MyCats#DX51"/>

<w><orth>dogs</orth>
  <cat>NNP</cat>
  <lemma>dog</lemma>
</w>
GrAF

– GrAF is the XML serialization of LAF abstract model
  • Graph is the abstract data model
  • Provides syntactic interoperability
– A (true) graph of nodes and edges, decorated with feature structures providing the annotation content
– Map user formats to this representation
  • Intended as a “pivot” for exchange/merging
Annotations

• **Label**
  – Arbitrary (in terms of GrAF specification) : desired label in XML element, GATE, etc.
  – LAF : Map to standard category definition

• **Feature Structure**
  – Feature structure content providing annotation information
  – Attribute-value pairs
  – Recursive
  – Can specify alternatives etc.
  – Has formal properties of feature structures, can apply standard FS operations (subsumption, unification, etc.)
Feature Structures

• Annotation content associated with nodes in the graph represented as feature structures
• Encoded using a variant of the ISO/TEI feature structure XML representation
  – A simplified version used for most cases
• Note: Feature structures are also graphs
  – Sub-graphs attached to nodes
Other parts of GrAF

- **Annotation spaces**
  - Like an XML namespace, provides context and enables disambiguation of names
    - Retains logical grouping, useful when merging annotations
- **Groups**
  - Groups annotations according to some logical principle
    - E.g., layers: morpho-syntax, syntax, etc.
Example

Segmentation

Morpho-syntactic layer

Syntactic layer
The Graph

- Nodes associated with annotations (label and feature structure)
- Edges connect nodes to
  - Other nodes in the annotation
    • This is where GrAF differs from some earlier representation formalisms, esp. Annotation Graphs
- Edges may also be associated with annotations or other information (e.g. type of relation)
- Links connect nodes to
  - Regions of primary data (when direct reference)
Example

Base segmentation:

<region xml:id="seg-r223" anchors="632 633"/>

Annotation over the base segmentation:

<node xml:id="ptb-n00256">
  <link targets="seg-r223"/>
</node>

<a label="tok" ref="ptb-n00256" as="PTB">
  <fs>
    <f name="msd" value="PRP"/>
  </fs>
</a>

Annotation over another annotation:

<node xml:id="ptb-n00255"/>

<a label="NP" ref="ptb-n00255" as="PTB">
  <fs>
    <f name="cat" value="NP"/>
    <f name="roles" value="SBJ"/>
  </fs>
</a>

<edge xml:id="ptb-e00252" from="ptb-n00255" to="ptb-n00256"/>
GrAF is a Pivot Format

- Intended to be used as the intermediary between specific annotation schemes
- Not intended to be the format in which user annotates data or uses the annotations
- Map user scheme to GrAF, map GrAF to other scheme
Merging Annotations

• Involves simply combining the graphs for each annotation

• Graph algorithms can be applied to collapse identically-labeled nodes with edges to common subgraphs
GrAF as a pivot

Different formats

Transduce to GrAF

Transduce merged graph

Transduce to other formats

Different formats

XML
CONLL
RDF
NLTK
XML
CONLL
RDF
NLTK
Advantages of Graph Model

• Isomorphic to formats used by emerging annotation frameworks and tools
  – E.g., UIMA’s Common Analysis System

• Underlies Web formats such as RDF and OWL
  – Annotation graph is trivially transducable to their serializations (including XML and several others)

• Provides a well-understood model and basis for devising linguistic annotation schemes
Using the Graph

• GrAF format isomorphic to input to many graph-analytic tools
• Graph-traversal and graph-coloring algorithms can identify commonly annotated components
  – E.g., identify higher-level nodes in PTB that cover the same spans as TB annotations
  – E.g., generate a list of all nodes annotated as ARG0 by both PB and NB
    • reveals clusters of verb/nominalization pairs
    • could be used to augment semantic lexicons
Other Possibilities

- Graph of annotations for each sentence interpreted as a finite state machine
  - Exploit wealth of analytic tools (e.g., reachable states)
  - Employ temporal logic
Graph-analytic Algorithms

• Common sub-graph analysis, shortest paths, minimum spanning trees, connectedness, identification of articulation vertices, topological sort, graph partitioning, frequent subgraphs, etc.
  – Useful for mining information from a graph of annotations at multiple linguistic levels
  – May reveal relationships and interactions previously difficult to observe
GraphViz Output
Other Advantages

• Graph format makes it easy to
  – add information
  – modify graph to reflect additional analysis, correct errors, etc.
    • E.g., delete or move constituents such as punctuation and parenthetical phrases, conjoin sub-graphs joined by “and”, correct PP attachments based on information in the tree, etc.
  – align in-line annotations of the same data
    • E.g., TimeBank’s version of the WSJ and the PTB’s version
GrAF: Formal specifications

• Schema (XML, Relax NG, RNC, DTD, plus HTML)
• Formal metadata specifications
• Physical and logical organization of the resource
  – File organization
  – Naming conventions
  – Annotation spaces
  – Annotation groups
  – Media definitions and associated anchor types

➢ Enables automatic validation and selection of sub-components
Overall GrAF resource architecture

- One or more **primary data documents**, in any medium
- One or more **base segmentation documents** defining a set of **regions** over a primary data document
- Any number of **annotation documents** containing feature structures associated with nodes and/or edges in a directed graph
- **Header documents** associated with each primary data document and annotation document, and a resource header that provides information about the resource as whole
GrAF Data Model
GrAF Headers

• **Resource header**
  - Contains all the formal specifications for whole resource
    • Creator, project information, etc.; domain/genre category definitions, media definitions, annotation set definitions, layers/tiers, file structure definition, annotation types, pointer to annotation scheme documentation…

• **Primary data document header**
  - Contains information about the primary data
    • Provenance, medium (point to resource header), language, writing system, genre/domain information (point to resource header), associated annotations…

• **Annotation header**
  - Information about a particular annotation
    • Format, creator, location of original, dependencies on other annotations, medium, anchor types (references to text or other annotations), annotation set…
GrAF Resource Header

• Specifications formal enough for machine processing
  – Validation, selection of sub-parts of the corpus
• E.g. define domain / genre categories

```xml
<classDecl>
<!-- Category codes are referenced in the header of each primary data document -->
<taxonomy id="MASC">
  <category id="WR">
    <catDesc>Written</catDesc>
  </category>
  <category id="JO">
    <catDesc>journal</catDesc>
  </category>
</taxonomy>

Would like to have a URI here
```
Overview of Resource Header

resourceHeader

encodingDesc

extent
publicationStmt
projectDesc
samplingDecl
editorialDecl
classDecl

resourceDesc

fileStruct
annotationSpaces
annotationDecls
media
anchorTypes
GrAF Resource Header

<fileType
  xml:id="f.entities"
  suffix="ne"
  a.ids="ne"
  medium="xml"
  requires="f.ptbtk"/>

Suffix in filenames
Id for reference
Id of medium type
Filetypes required to process this filetype
GrAF Resource Header

Annotation spaces

<annotationSpaces>
  <annotationSpace xml:id="ptb" pid="http://www.cis.upenn.edu/~treebank/">
    <annotationSpace xml:id="fn" pid="http://framenet.icsi.berkeley.edu/">
      Reference to persistent identifier for the annotation type
    </annotationSpace>
  </annotationSpace>
</annotationSpaces>
Annotation Declaration

<annotationDecl xml:id="a.ne" as="xces">  
  <a.desc>named entities</a.desc>  
  <a.resp lnk:href="http://www.anc.org">ANC project</a.resp>  
  <a.method type="automatic-validated"/>  
</annotationDecl>
GrAF Resource Header

<medium xml:id = "text"
    type = "text/plain"
    encoding = "utf-8"
    extension = "txt"/>

<medium xml:id = "video"
    type = "video"
    encoding = "Cinepak"
    extension = "mov"/>

<medium xml:id = "image"
    type = "image"
    encoding = "jpeg"
    extension = "jpg"/>

<anchorType xml:id="text-anchor" medium = "text" default = "true"
    lnk:href = "http://www.xces.org/ns/GrAF/1.0/#character-anchor"/>

<anchorType
    xml:id="text-anchor"
    medium = "text"
    default = "true"
    lnk:href = "http://www.xces.org/ns/GrAF/1.0/#char-anchor"/>
GrAF Resource Header

Associating files, annotations, media, anchors, etc.

```
<fileType xml:id = "f.entities" suffix = "ne" a.ids = "a.ne"
   medium = "xml" requires = "f.ptbtok"/>
...
<annotationSpace xml:id = "xce" pid = "http://www.xces.org/schema/2003"/>
...
<annotationDecl xml:id="a.ne" as="xce">
   <a.desc>named entities</a.desc>
   <a.resp lnk:href="http://www.anc.org">ANC project</a.resp>
   <a.method type="automatic-validated"/>
</annotationDecl>
...
<medium xml:id = "text" type = "text/plain"
   encoding = "utf-8" extension = "txt"/>
<medium xml:id = "xml" type = "text/xml"
   encoding = "utf-8" extension = "xml"/>
...
<anchorType medium = "text" default = "true"
   lnk:href = "http://www.xces.org/ns/GrAF/1.0/#character anchor"/>
```
Groups (layers, tiers, etc.)

<groups>
  <group xml:id = "g.token">
    <!-- all annotations in any annotation space with label "tok" -->
    <g.member value = "*:tok" type = "annotation"/>
  </group>
  <group xml:id = "g.example">
    <!-- all annotations of type logical -->
    <g.member value = "a logical" type = "type"/>
    <!-- all files containing entity annotations -->
    <g.member value = "f.entities" type = "file"/>
    <!-- all annotations with a feature "speaker" with value "Alice" -->
    <g.member value = "@speaker = 'alice" type = "expression"/>
    <!-- annotations with ids "id_1" to "id_n" in file "myfile.xml" -->
    <g.member xml:base = "myfile.xml" value = "id1 id2 ... idN"
      type = "enumeration"/>
    <!-- the annotations included in group g.token, as defined earlier -->
    <g.member value = "g.token" type = "group"/>
  </group>
</groups>
GrAF Primary Document Header

<primaryData loc="Day3PMSession.txt" medium="text"/>
<annotations>
  <annotation ann.loc="Day3PMSession‐logical.xml" type="logical">Logical structure</annotation>
  <annotation ann.loc="Day3PMSession‐s.xml" type="s">Sentence boundaries</annotation>
  <annotation ann.loc="Day3PMSession‐nc.xml" type="nc">Noun chunks</annotation>
  <annotation ann.loc="Day3PMSession‐penn.xml" type="penn">Penn part of speech tags</annotation>
  <annotation ann.loc="Day3PMSession‐ptb.xml" type="ptb">Penn Tree Bank</annotation>
  <annotation ann.loc="Day3PMSession‐ptbtok.xml" type="ptbtok">Penn Tree Bank tokens and part of speech tags</annotation>
  <annotation ann.loc="Day3PMSession‐seg.xml" type="seg">Base segmentation (quarks)</annotation>
  <annotation ann.loc="Day3PMSession‐ne.xml" type="ne">Named Entities</annotation>
  <annotation ann.loc="Day3PMSession.txt" type="content">Document content</annotation>
</annotations>
GrAF Annotation Documents

• Annotation documents contain both a header and the graph of feature structures comprising the annotation
• Header contains:
  – a list of the annotation labels used in the document and their frequencies;
  – a list of documents required to process the annotations, which will include a segmentation document and/or any annotation documents directly referenced in the document;
  – a list of annotation Spaces referenced in the document, one of which may be designated as a default for annotations in the document;
  – optional) The root node(s) in the graph, when the graph contains one or more graphs that comprise a well-formed tree.
Dependencies among headers

- DIRECTORY
  - FILETYPE
    - medium
    - suffix
    - annotationType
    - status
  - MEDIUM
    - type
    - encoding
    - extension
  - ANCHORTYPE
    - medium
  - ANNOTATIONDECL
    - annotationSpace
  - ANNOTATIONSPACE
  - GROUP
  - MYFILE-FSUFFIX.M-SUFFIX
Anchors and Regions

<!-- Regions in the segmentation document -->
<region xml:id="r1" anchor_type="time-slot" anchors="980 983"/>
<region xml:id="r2" anchor_type="image-point"
    anchors="10,59 10,173 149,173 149,59"/>
<region xml:id="r3" anchor_type="video-anchor"
    anchors="frame1(10,59) frame2(59,85) frame3(85,102)"/>
<region xml:id="r4" anchor_type="text-anchor"
    anchors="34 42"/>
Nodes and Regions

<region xml:id="seg-r770" anchors="2211 2216"/>
<region xml:id="seg-r771" anchors="2216 2217"/>
<region xml:id="seg-r772" anchors="2217 2221"/>

<node xml:id="n1019">
  <link targets="seg-r770 seg-r771 seg-r772"/>
</node>
<a label="tok" ref="n1019" as="xces">
  <fs>
    <f name="msd" value="JJ"/>
  </fs>
</a>

Three-fold
Nodes and Edges

```
<node xml:id = "fn-n3"/>
<a label = "FE" ref = "fn-n3" as = "FrameNet">
  <fs>
    <f name = "name" value = "Supplier"/>
    <f name = "GF" value = "Ext"/>
    <f name = "PT" value = "NP"/>
  </fs>
</a>
<edge xml:id = "e46" from = "fn-as1" to = "fn-n3"/>
<edge xml:id = "e92" from = "fn-n3" to = "fntok:fn-t3"/>
```
Real interoperability

• GrAF has been used to represent annotations contributed to MASC and the Open ANC in a wide variety of formats
  – Penn Treebank, FrameNet, PropBank, TimeBank, and many more
    • Proof of concept

• The tools are there that show how GrAF can be used to achieve interoperability
  – Change between environments to do different tasks as suited: GATE for manual annotation, use of plugins; UIMA for other tasks; NLTK for statistical analysis…
MASC/OANC Example

<table>
<thead>
<tr>
<th>FrameNet</th>
<th>AGENT [FN-n1 FN-n2 FN-n3 FN-n4 FN-n5 FN-n6]</th>
</tr>
</thead>
<tbody>
<tr>
<td>FN-n1 NNP [r0]</td>
<td>FN-n2 POS [r1]</td>
</tr>
<tr>
<td>FN-n3 NNP [r2 r3]</td>
<td>FN-n4 - [r4]</td>
</tr>
<tr>
<td>FN-n5 JJ [r5]</td>
<td>FN-n6 NN [r6]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FN msd</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Noun chunk</td>
<td>NC-n1 [r1 r2 r3 r4 r5 r6]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PTB syntax</th>
<th>NP-n1 [PTB-n1 PTB-n2 PTB-n3 PTB-n4 PTB-n5 PTB-n6]</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTB-n1 NNP [r0]</td>
<td>PTB-n2 POS [r1]</td>
</tr>
<tr>
<td>PTB-n3 NNP [r2 r3]</td>
<td>PTB-n4 - [r4]</td>
</tr>
<tr>
<td>PTB-n5 JJ [r5]</td>
<td>PTB-n6 NN [r6]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PTB msd</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>segmentation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>r0 0-3</td>
</tr>
<tr>
<td></td>
<td>r1 3-5</td>
</tr>
<tr>
<td></td>
<td>r2 6-9</td>
</tr>
<tr>
<td></td>
<td>r3 10-14</td>
</tr>
<tr>
<td></td>
<td>r4 14-15</td>
</tr>
<tr>
<td></td>
<td>r5 15-20</td>
</tr>
<tr>
<td></td>
<td>r6 21-27</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>text</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>01B2M34sNEwYoKr-bas18ed02ffic26e27</td>
</tr>
</tbody>
</table>
ANC Tool

• ANC Tool
  – Converts stand-off annotations in LAF/GrAF to other formats
    • Currently, in-line XML, MonoConc, basic NLTK, CONLL, RDF
  – Merges annotations of the user’s choosing
  – Freely available on ANC website
    http://www.anc.org
ANC2Go

• Custom corpus creation
• Web application
  – Online interface to the ANC Tool
• User selects
  – Part/all of corpus
  – Annotations to be included
  – Output format
• Output downloaded from web
GrAF as a stand-alone format

- GrAF API to access graph
  - Well-known data model and algorithms
  - Well-known access and manipulation operations
- Also
  - Modules for GATE (General Architecture for Text Engineering - U Sheffield) to read in and write out annotations in GrAF format
  - Module for UIMA (Unstructured Information Management System - IBM / Oasis) to read in and write out annotations in GrAF format
  - Corpus reader for NLTK to read in annotations in GrAF format
  - GraphViz renderer (in GrAF API) to enable visualizing graphs
Summary

- LAF/GrAF is a state-of-the-art means to represent linguistic data and annotations
- Implemented in a real corpus, tools to manage
- Shows the power of the well-known graph formalism to represent as well as search, access, and manipulate annotations
- **LAF/GrAF is now an ISO standard (ISO 24612)**
  - Contact me for a copy
Beyond GrAF

- GrAF says nothing about a number of important things:
  - What should be the object of an annotation and what should be features
  - What the role of labels on an edge should be
    - Labels as in a dependency parse?
    - Relations only? (coreference, timelink, etc.)
    - Processing specification? (multiple edges are ordered constituents, alternatives, etc.)
  - How to formally define relations among annotation objects
- The next step is to address these issues
  - New work item within ISO TC37 SC4 WG1 (convener: Nancy Ide)
Exercise

Designing Annotation Schemes

EBRALC
Sao Carlos, Brazil
September 11-15, 2012
Why annotate?

• Language annotation is a critical link in developing intelligent human language technologies
• Large bodies of data enable linguistic modeling problems to be viewed as machine learning tasks
  – Not limited to relatively small amounts of data that humans can process
• Data must be prepared so computer can find patterns and make inferences
  – Add annotations over the data identifying linguistic phenomena
    • Must be
      – Accurate
      – Relevant to the task
Designing an Annotation Scheme

• Two phases
  – Specification of a model (abstract syntax)
  – Specification of (possibly multiple) representation structures (concrete syntax)

• Possibilities
  – GrAF
  – XML Corpus Encoding Standard
  – Bracketed constituency
  – Etc.
Developing the Annotation Scheme

• Several steps in the process
  – Model – Annotate – Train – Test – Revise (MATTER)

• Exercise will focus on Modeling
  – Steps involved vary depending on what you will use the annotations for
  – What is a model?
    • Characterization of a phenomenon in terms that are more abstract than the elements in the domain being modeled
      ➢ Abstract model
Abstract Model

• Consists of
  – C: Conceptual inventory (terms)
  – R: Annotation construction rules that specify structural relations between terms
  – I: Interpretation (semantics)

Can represent using nodes for terms and edges for relations – a graph!
Simple Example

- **Named entities**
  - C = \{Organization, Person, Location, Date\}
  - R = \{Named_entity = Organization | Person | Location| Date\}

- I = \{ Organization = "an organization listed in a database", Person = "a person from a list of people in a database", Location= "a geographic location", Date= "one of all possible dates on the calendar" \}

**Named entity**: an object in the world that has a name that uniquely identifies it

E.g., Barack Obama (person), IBM (organization), Brazil (Location), September 11, 2012 (Date)
More complex example

- **Temporal annotation**
  - $C = \{ \text{Time\_Expression} = \{\text{time, date, duration}\}, \text{Event}, \text{Temporal\_relation} = \{\text{before | after | during | overlap}\}\} \}$
  - $R = \{ \text{Temporal\_anchoring} = (\text{Event, Time\_expression, Temporal\_relation}) \}$
  - $I = \{ \text{Time: 10:15 am, 3 o'clock, etc.; Date: Monday, April, 2012, etc.; Duration: 30 minutes, two years, four days, etc.; Event: meeting, vacation, promotion, maternity leave, etc.} \}$
Concrete syntax

• Realization in physical format
  – Example: Named entities
    • `<named_entity type="Person">Henry Jones</named_entity>`
    • `<Person name="Henry Jones" text_span="52 62"/>`
    • `(Named_entity (Person Henry Jones))`
  – Example: Temporal annotation
    
    I fasted for two days.
    `<EVENT xml:id="e1" type ="fast" ref="token2"/>`
    `<TIME_expression xml:id="t1" type="duration" ref="token4 token5"/>`
    `<TIME-ANCHORING anchoredEvent="e1" anchorTime="t1" relType="during" ref="token3"/>`

    (TIME-ANCHORING (EVENT fasted) (TIME_EXPRESSION two days) for)
Creating an annotation scheme

• What are the categories?
  – This depends on several factors:
    • Theoretical framework
      – Different linguistic theories have different views
        » Syntax: phrase structure vs. dependency
        » Semantic roles: different labels for same phenomenon (patient vs. object, ARG0 vs. ??)
    • Goal of the annotation
      – Different tasks will require specific entities, events, attributes, or facts
        » E.g. classifying emails as spam or not-spam probably won’t need named entities
        » Machine learning may need certain features (e.g., POS tags, orthographic characteristics, etc.)
        » Domain-specific analysis may require special annotations
          • E.g., biomedical domain would need annotation for genes, proteins, etc.
Example: Film genre classification

• Classifying documents into categories
  – Use film reviews or summaries to determine the genre of the film being described
  – For machine learning:
    • Model: label summary with all genres the movie applies to
    • Then feed those labels into a classifier, train to identify relevant parts of the document
  – Create the spec:
    • Create a tag that captures the information needed for the goal and model
Example: Film genre classification

• One possibility:
  – An annotation called genre with attribute label
  – Values for label:
    • Adventure, Animation, Biography, Comedy, Crime, Documentary, Drama, Family, Fantasy, Film-Noir, Game-Show, History, Horror, Music, Musical, Mystery, News, Reality-TV, Romance, Sci-Fi, Sport, Talk-Show, Thriller, War, Western
      – List from www.imdb.com/genre
      – Other genre lists exist (e.g., www2.netflix.com/AllGenresList)
        ➢ Choose the one that best matches your task, or create your own list
Revising the spec

- As you annotate and train, you may find places where your model/spec needs revision to get the results that you want
  - Normal, even for tasks that seem as straightforward as putting genre labels on movie summaries
  - Computer algorithms don’t think and interpret the way people do
Example 1

- Romance and comedy are separate genres; romantic comedy would have to have two labels
  - If in a significant portion of reviews those two tags appear together, an algorithm may learn to always associate the two even when the summary is really a romantic drama or musical comedy
  - Might find it necessary to create a rom-com label to prevent classifier from creating false associations
Example 2

• Many historical action movies take place over very different periods in history
  – Machine learning algorithm may have trouble finding enough common ground between a summary of Batman, Braveheart, and Pearl Harbor to create an accurate association with the “history” genre
  – Might have to add different levels of historical genres that reflect different periods in history in order to train a classifier accurately
Revising the spec:
Adding named entities

• Could change model to better fit the task by adding annotations that more closely reflect relevant information
  – E.g., identify named entities in movie summaries that give insight into genre
    • film titles, directors, writers, actors, characters…
Generality vs. specificity

• Major concern when designing a scheme
• Again, depends on goal

• Example
  – Entity classifications such as **Organization**, **Person**, **Place**, and **Date** too general for film genre annotation
    • Director, writer, actor, and character all fall under label **Person**
    • Film title doesn’t fit under any of them
Generality vs. specificity

• Using these labels would lead to unhelpful annotations

• Labels would be so generic as to be useless for the task
  – Labeling Director, writer, actor, and character as Person won’t help distinguish one genre from another
  – Difficult to expect annotators to annotate only the entities that are relevant to the film
    • E.g., mention of another reviewer should not be labeled as a Person
Generality vs. specificity

• Alternatively, make tags in the spec more specific
  – E.g., actor_star, actor_minor_character, character_main, character_minor, writer_film, writer_book, writer_book_and_film…
  – Is this useful?
The hierarchy

Too general?

Probably about the right level

Too specific?
More revisions?

• Closer look at the film genres shows not all genres in the list describe the same aspects of the film
  – Labels like action, adventure, crime, romance refer to events in the film
  – Labels historical, sci-fi, fantasy refer to the setting
  – Animation, talk-show, reality-tv describe production circumstances

• Could break categories (production, setting, etc.) and assign at least one label from each category
Your Turn

• Your task is to design a rudimentary annotation scheme for one of the following phenomena
  – Entities
    • Add to the usual Person, Location, Organization, Date by either specifying more precise set of sub-categories for these types, or extending the set of types
    • May focus on a specific genre if desired
  – Spatial information
    • Spatial indicators (words that convey spatial information)
    • Entities that can be involved in spatial relations
    • Spatial relations among entities
Strategy

• Choose a text from the MASC data
  – Suggestions:
    • Travel guides
    • Chapter 10 (9/11 report)
  – But use any text you want

• Use the text to suggest the possible elements of your annotation scheme

• Annotate a different text (in same genre if appropriate) with your scheme in GATE
General approach

• Work in teams of 3-4
• Start with a general specification!!!
  – High level entities etc.
  – Skip over the more difficult phenomena at first
  – Try some annotation using this scheme
  – If there is time you can refine
• Results
  – Compare schemes for same phenomenon designed by different groups (should be interesting)
  – See how well the scheme works on a new text