GrAF: A Graph-based Format for Linguistic Annotations

Nancy Ide
Department of Computer Science
Vassar College
Poughkeepsie, New York USA
ide@cs.vassar.edu

Keith Suderman
Department of Computer Science
Vassar College
Poughkeepsie, New York USA
suderman@cs.vassar.edu

Abstract

In this paper we describe the Graph Annotation Format (GrAF) and show how it is used represent not only independent linguistic annotations, but also sets of merged annotations as a single graph. To demonstrate this, we have automatically transduced several different annotations of the *Wall Street Journal* corpus into GrAF and show how the annotations can then be merged, analyzed, and visualized using standard graph algorithms and tools. We also discuss how, as a standard graph representation, it allows for the application of well-established graph traversal and analysis algorithms to produce information about interactions and commonalities among merged annotations. GrAF is an extension of the Linguistic Annotation Framework (LAF) (Ide and Romary, 2004, 2006) developed within ISO TC37 SC4 and as such, implements state-of-the-art best practice guidelines for representing linguistic annotations.

1 Introduction

Although linguistic annotation of corpora has a long history, over the past several years the need for corpora annotated for a wide variety of phenomena has come to be recognized as critical for the future development of language processing applications. Considerable attention has been devoted to the development of means to represent annotations so that phenomena at different levels can be merged and/or analyzed in combination. A particular focus has been on the development of standards and best practices for representing annotations that can facilitate "annotation interoperability", that is, the use and re-use of annotations produced in different formats and by different groups and to enable easy adaptation to the input requirements of existing annotation tools.

In this paper we describe the Graph Annotation Format (GrAF) and show how it is used represent not only independent linguistic annotations, but also sets of merged annotations as a single graph. We also discuss how, as a standard graph representation, it allows for the application of well-established graph traversal and analysis algorithms to produce information about interactions and commonalities among merged annotations. GrAF is is an extension of the Linguistic Annotation Framework (LAF) (Ide and Romary, 2004, 2006) developed within ISO TC37 SC4[1] and as such, implements state-of-the-art best practice guidelines for representing linguistic annotations.

This paper has several aims: (1) to show the generality of the graph model for representing linguistic annotations; (2) to demonstrate how the graph-based model enables merging and analysis of multi-layered annotations; and (3) to propose as the underlying model for linguistic annotations, due to its generality and the ease with which it is mapped to other formats. To accomplish this, we have automatically transduced several different annotations of the *Wall Street Journal* corpus into GrAF and show how the annotations can then be merged, analyzed, and visualized using standard graph algorithms and tools. Discussion of the

[1] International Standards Organization Technical Committee 37 Sub-Committee 4 for Language Resource Management.

Proceedings of the Linguistic Annotation Workshop, pages 1–8,
Prague, June 2007.

transduction process brings to light several problems and concerns with current annotation formats and leads to some recommendations for the design of annotation schemes.

2 Overview

Graph theory provides a well-understood model for representing objects that can be viewed as a connected set of more elementary sub-objects, together with a wealth of graph-analytic algorithms for information extraction and analysis. As a result, graphs and graph-analytic algorithms are playing an increasingly important role in language data analysis, including finding related web pages (Kleinberg, 1999; Dean and Henzinger, 1999; Brin, 1998; Grangier and Bengio, 2005), patterns of web access (McEneaney, 2001; Zaki, 2002), and the extraction of semantic information from text (Widdows and Dorow, 2002; Krizhanovsky, 2005; Nastase and Szpakowicz, 2006). Recently, there has been work that treats linguistic annotations as graphs (Cui *et al.*, 2005; Bunescu and Mooney, 2006; Nguyen *et al.*, 2007; Gabrilovich and Markovitch, 2007) in order to identify, for example, measures of semantic similarity based on common subgraphs.

As the need to merge and study linguistic annotations for multiple phenomena becomes increasingly important for language analysis, it is essential to identify a general model that can capture the relevant information and enable efficient and effective analysis. Graphs have long been used to describe linguistic annotations, most familiarly in the form of trees (a graph in which each node has a single parent) for syntactic annotation. Annotation Graphs (Bird and Liberman, 2001) have been widely used to represent layers of annotation, each associated with primary data, although the concept was not extended to allow for annotations linked to other annotations and thus to consider multiple annotations as a single graph. More recently, the Penn Discourse TreeBank released its annotations of the Penn TreeBank as a graph, accompanied by an API that provides a set of standard graph-handling functions for query and access[2]. The graph model therefore seems to be gaining ground as a natural and flexible model for linguistic annotations which, as we demonstrate below, can represent all annotation varieties, even those that were not originally designed with the graph model as a basis.

[2] http://www.seas.upenn.edu/~nikhild/PDTBAPI/

2.1 LAF

LAF provides a general framework for representing annotations that has been described elsewhere in detail (Ide and Romary, 2004, 2006). Its development has built on common practice and convergence of approach in linguistic annotation over the past 15-20 years. The core of the framework is specification of an abstract model for annotations instantiated by a *pivot format*, into and out of which annotations are mapped for the purposes of exchange.

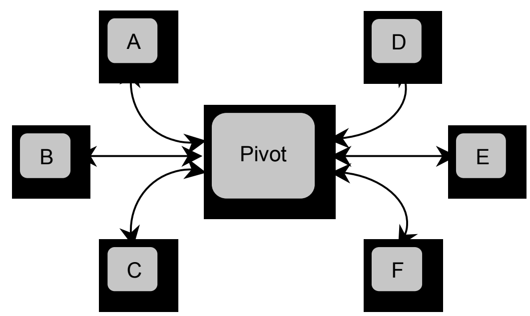


Figure 1: Use of the LAF pivot format

Figure 1 shows the overall idea for six different user annotation formats (labeled A – F), which requires two mappings for each scheme—one into and one out of the pivot format, provided by the scheme designer. The maximum number of mappings among schemes is therefore $2n$, vs. n^2-n mutual mappings without the pivot.

To map to the pivot, an annotation scheme must be (or be rendered via the mapping) isomorphic to the abstract model, which consists of (1) a *referential structure* for associating stand-off annotations with primary data, instantiated as a directed graph; and (2) a *feature structure representation* for annotation content. An annotation thus forms a directed graph referencing *n*-dimensional regions of primary data as well as other annotations, in which nodes are labeled with feature structures providing the annotation content. Formally, LAF consists of:

- A data model for annotations based on directed graphs defined as follows: A graph of annotations G is a set of vertices $V(G)$ and a set of edges $E(G)$. Vertices and edges may be labeled with one or more features. A feature consists of a quadruple (*G', VE, K, V*) where, *G'* is a graph, *VE* is a vertex or edge in *G'*, *K* is the name of the feature and *V* is the feature value.

- A *base segmentation* of primary data that defines edges between virtual nodes located between each "character" in the primary data.[3] The resulting graph *G* is treated as an *edge graph G'* whose nodes are the edges of *G*, and which serve as the leaf ("sink") nodes. These nodes provide the base for an annotation or several layers of annotation. Multiple segmentations can be defined over the primary data, and multiple annotations may refer to the same segmentation.
- Serializations of the data model, one of which is designated as the pivot.
- Methods for manipulating the data model.

Note that LAF does not provide specifications for annotation *content categories* (i.e., the labels describing the associated linguistic phenomena), for which standardization is a much trickier matter. The LAF architecture includes a *Data Category Registry* (DCR) containing pre-defined data elements and schemas that may be used directly in annotations, together with means to specify new categories and modify existing ones (see Ide and Romary, 2004).

2.2 GrAF

GrAF is an XML serialization of the generic graph structure of linguistic annotations described by LAF. A GrAF document represents the referential structure of an annotation with two XML elements: `<node>` and `<edge>`. Both `<node>` and `<edge>` elements may be labeled with associated annotation information. Typically, annotations describing a given object are associated with `<node>` elements. Although some annotations, such as dependency analyses, are traditionally depicted with labeled edges, GrAF converts these to nodes in order to analyze both the annotated objects and the relations of a graph uniformly. Associating annotations with nodes also simplifies the association of an annotation (node) with multiple objects.

According to the LAF specification, an annotation is itself a graph representing a feature structure. In GrAF, feature structures are encoded in XML according to the specifications of ISO TC37 SC4 document 188[4]. The feature structure graph associated with a given node is the corresponding `<node>` element's content. Note that the ISO specifications implement the full power of feature structures and define inheritance, unification, and subsumption mechanisms over the structures, thus enabling the representation of linguistic information at any level of complexity. The specifications also provide a concise format for representing simple feature-value pairs that suffices to represent many annotations, and which, because it is sufficient to represent the vast majority of annotation information, we use in our examples.

`<edge>` elements may also be labeled (i.e., associated with a feature structure), but this information is typically not an annotation *per se*, but rather information concerning the meaning, or role, of the link itself. For example, in PropBank, when there is more than one target of an annotation (i.e., a node containing an annotation has two or more outgoing edges), the targets may be either co-referents or a "split argument" whose constituents are not contiguous, in which case the edges collect an ordered list of constituents. In other case, the outgoing edges may point to a set of alternatives. To differentiate the role of edges in such cases, the edge may be annotated. Unlabeled edges default to pointing to an unordered list of constituents.

A base segmentation contains only `<sink>` elements (i.e., nodes with no outgoing edges), which are a sub-class of `<node>` elements. As noted above, the segmentation is an edge graph created from edges (spans) defined over primary data. The *from* and *to* attributes on `<sink>` elements in the base segmentation identify the start and end points of these edges in the primary data.

Each annotation document declares and associates the elements in its content with a unique namespace. Figure 2 shows several XML fragments in GrAF format.

[3] A character is defined to be a contiguous byte sequence of a specified length .For text, the default is UTF-16.

[4] See ISO TC37 SC4 document N188, Feature Structures-Part 1: Feature Structure Representation (2005-10-01), available at http://www.tc37sc4.org/

Base segmentation:

```
<seg:sink seg:id="42" seg:start="24"
      seg:end="35"/>
```

Annotation over the base segmentation:

```
<msd:node msd:id="16">
   <msd:f name="cat" value="NN"/>
</msd:node>

<msd:edge from="msd:16" to="seg:42"/>
```

Annotation over another annotation:

```
<ptb:node ptb:id="23">
   <ptb:f name="type" value="NP"/>
   <ptb:f name="role" value="-SBJ"/>
</ptb:node>
```

Figure 2: GrAF annotations in XML

3 Transduction

To test the utility of GrAF for representing annotations of different types produced by different groups, we transduced the Penn TreeBank (PTB), PropBank (PB), NomBank (NB), Penn Discourse TreeBank (PDTB), and TimeBank (TB) annotations of the *Wall Street Journal (WSJ)* corpus to conform to the specifications of LAF and GrAF. These annotations are represented in several different formats, including both stand-off and embedded formats. The details of the transduction process, although relatively mundane, show that the process is not always trivial. Furthermore, they reveal several seemingly harmless practices that can cause difficulties for transduction to any other format and, therefore, use by others. Consideration of these details is therefore informative for the development of best practice annotation guidelines.

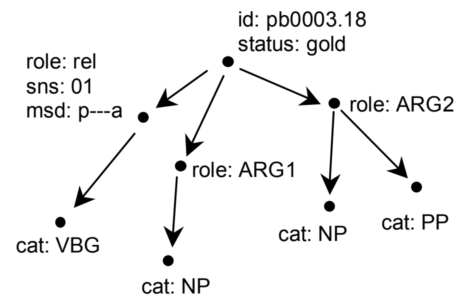
The Penn TreeBank annotations of the *WSJ* are embedded in the data itself, by bracketing components of syntactic trees. Leaf nodes of the tree are comprised of POS-word pairs; thus, the PTB includes annotations for both morpho-syntax and syntax. To coerce the annotations into LAF/GrAF, it was necessary to

- extract the text in order to create a primary data document;
- provide a primary segmentation reflecting the tokenization implicit in the PTB;
- separate the morpho-syntactic annotation from the syntactic annotation and render each as a stand-off document in GrAF format, with links to the primary segmentation.

NB, PB, and PDTB do not annotate primary data, but rather annotate the PTB syntax trees by providing stand-off documents with references to PTB Tree nodes. The format of the NB and PB stand-off annotations is nearly identical; consider for example the following PB annotation:

```
wsj/00/wsj_0003.mrg 18 18 gold include.01
p---a 14:1,16:1-ARG2 18:0-rel 19:1-ARG1
```

In GrAF, this becomes



Each line in the PB and NB stand-off files provides a single annotation and therefore interpreted as an annotation node with a unique *id*. Each annotation is associated with a node with an edge to the annotated entity. The PB/NB comma notation (e.g., `14:1,16:1`) denotes reference to more than one node in the PTB tree; in GrAF, a dummy node is created to group them so that if, for example, a NB annotation refers to the same node set, in a merged representation a graph minimization algorithm can collapse the dummy nodes while retaining the annotations from each of PB and NB as separate nodes.

Some interpretation was required for the transduction, for example, we assume that the sense number and morpho-syntactic descriptor are associated with the element annotated as “rel” (vs. the “gold” status that is associated with the entire proposition), an association that is automatically discernible from the structure. Also, because the POS/word pairs in the PTB leaf nodes have been split into separate nodes, we assume the PB/NB annotations should refer to the POS annotation rather than the string in the primary data, but either option is possible.

Given the similarities of the underlying data models for the PDTB and LAF, creating GrAF-compliant structures from the PDTB data is relatively trivial. This task is simplified even further because the PDTB API allows PDTB files to be loaded in a few simple steps, and allows the programmer to set and query features of the node as well as iterate over the children of the node. So, given a node *P* that represents the root node of a PDTB tree, an equivalent graph *G* in GrAF format can be created by traversing the PDTB tree and creating matching nodes and edges in the graph *G*.

Like the PTB, TimeBank annotation is embedded in the primary data by surrounding annotated elements with XML tags. TB also includes sets of "link" tags at the end of each document, specifying relations among annotated elements. The same steps for rendering the PTB into GrAF could be followed for TB; however, this would result in a separate (and possibly different) primary data document. Therefore, it is necessary to first align the text extracted from TB with the primary data derived from PTB, after which the TB XML annotations are rendered in GrAF format and associated with the corresponding nodes in the base segmentation.

Note that in the current GrAF representation, TB's *tlink*, *slink*, and *alink* annotations are applied to edges, since they designate relations among nodes. However, further consideration of the nature and use of the information associated with these links may dictate that associating it with a node is more appropriate and/or useful.

Variations in tokenization exist among the different annotations, most commonly for splitting contractions or compounds ("cannot" split into "can" and "not", "New York-based" split into "New York", "-", and "based", etc.). This can be handled by adding edges to the base segmentation (not necessarily in the same segmentation document) that cover the relevant sub-spans, and pointing to the new edge nodes as necessary. Annotations may now reference the original span, the entire annotation, or any sub-part of the annotation, by pointing to the appropriate node. Alternative segmentations of the same span can be joined by a "dummy" parent node so that when different annotations of the same data are later merged, nodes labeling a sub-graph covering the same span can be combined. For example, in Figure 3, if the PTB segmentation (in gray) is the base segmentation, an alternative segmentation of the same span (in black) is created and associated to the PTB segmentation via a dummy node. When annotations using each of the different segmentations are merged into a single graph, features associated with any node covering the same sub-tree (in bold) are applied to the dummy node (as a result of graph minimization), thus preserving the commonality in the merged graph.

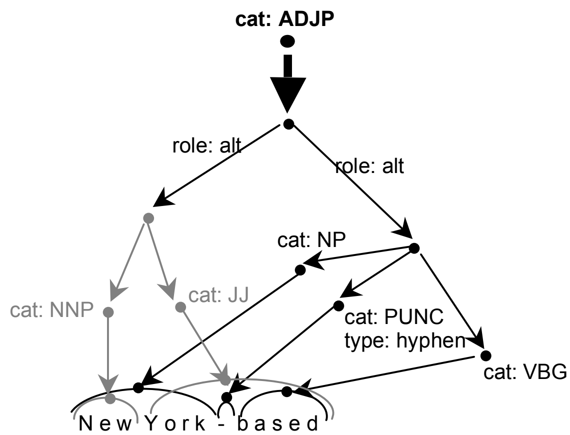


Figure 3: Alternative segmentations

4 Merging Annotations

Once they are in in GrAF format, merging annotations of the same primary data, or annotations referencing annotations of the same primary data, involves simply combining the graphs for each annotation, starting with graph *G* describing the base segmentation and using the algorithm in Figure 4. Once merged, graph minimization, for which efficient algorithms exist (see, e.g., Cardon and Crochemore, 1982; Habib *et al.*, 1999), can be applied to collapse identically-labeled nodes with edges to common subgraphs and eliminate dummy nodes such as the one in Figure 3.

```
Given a graph G :

for each graph of annotations Gp do
  for each vertex vp in Gp do
    if vp is not a leaf in Gp then
      add vp to G
  for each edge (vi, vj) in Gp do
    if vj is a leaf in Gp then
      find corresponding vertex vg ∈ G
         add a new edge (vi, vg) to G
      else
       add edge (vi, vj) to G
```

Figure 4: Graph-merging algorithm

5 Using the Graphs

Because the GrAF format is isomorphic to input to many graph-analytic tools, existing software can be exploited; for example, we have generated graph diagrams directly from a merged graph including PTB, NB, and PB annotations using GraphViz[5], which takes as its input a simple text file representation of a graph. Generating the input files to GraphViz involves simply iterating over the nodes and edges in the graph and printing out a suitable string representation. Figure 5 shows a segment of the GraphViz output generated from the PTB/NB/PB merged annotations (modified slightly for readability).

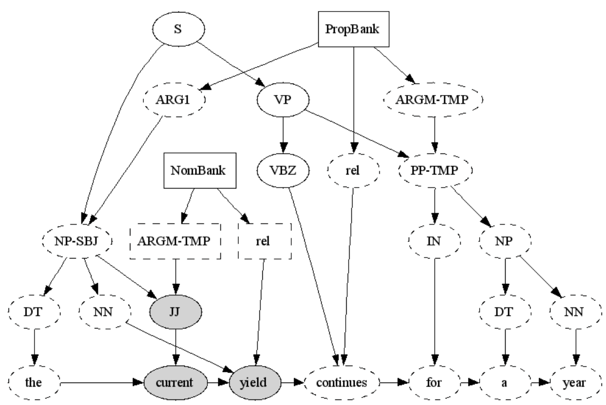


Figure 5: Fragment of GraphViz output

Graph-traversal and graph-coloring algorithms can be used to identify and generate statistics concerning commonly annotated components in the merged graph. For example, we modified the merging algorithm to "color" the annotated nodes as the graphs are constructed to reflect the source of the annotation (e.g., PTB, NB, PB, etc.) and the annotation content itself. Colors are propagated via outgoing edges down to the base segmentation, so that each node in the graph can be identified by the source and type of annotation applied. The colored graph can then be used to identify common subgraphs. So, for example, a graph traversal can identify higher-level nodes in PTB that cover the same spans as TB annotations, which in the merged graph are connected to sink nodes (tokens) only, thus effectively "collapsing" the two annotations.

Traversal of the colored graph can also be used to generate statistics reflecting the interactions among annotations. As a simple example, we generated a list of all nodes annotated as ARG0 by both PB and NB[6], the "related" element (a verb for PB, a nominalization for NB), the PTB annotation, and the set of sink nodes covered by the node, which reveals clusters of verb/nominalization pairs and can be used, for example, to augment semantic lexicons. Similar information generated via graph traversal can obviously provide a wealth of statistics that can in turn be used to study interactions among linguistic phenomena. Other graph-analytic algorithms—including common sub-graph analysis, shortest paths, minimum spanning trees, connectedness, identification of articulation vertices, topological sort, graph partitioning, etc.—may prove to be useful for mining information from a graph of annotations at multiple linguistic levels, possibly revealing relationships and interactions that were previously difficult to observe. We have, for example, generated frequent subgraphs of the PB and NB annotations using the IBM Frequent Subgraph Miner[7] (Inokuchi *et al.*, 2005). We are currently exploring several additional applications of graph algorithms to annotation analysis.

The graph format also enables manipulations that may be desirable in order to add information, modify the graph to reflect additional analysis, correct errors, etc. For example, it may be desirable to delete or move constituents such as punctuation and parenthetical phrases under certain circumstances, conjoin sub-graphs whose sink nodes are joined by a conjunction such as "and", or correct PP attachments based on information in the tree.

6 Discussion

GrAF provides a serialization of annotations that follows the specifications of LAF and is therefore a candidate to serve as the LAF pivot format. The advantages of a pivot format, and, in general, the use of the graph model for linguistic annotations, are numerous. First, transduction of the various formats into GrAF, as described in section 4, demanded substantial programming effort; similar effort would be required to transduce to any other

[5] www.graphviz.org

[6] The gray nodes in Figure 5 are those that have been "colored" by both PB and NB.

[7] http://www.alphaworks.ibm.com/tech/fsm

format, graph-based or not. The role of the LAF pivot format is to reduce this effort across the community by an order of magnitude, as shown in Figure 1. Whether or not GrAF is the pivot, the adoption of the graph *model,* at least for the purposes of exchange, would result in a similar reduction of effort, since graph representations are in general trivially mappable.

In addition to enabling the generation of input to a wide range of graph-handling software, the graph model for annotations is isomorphic to representation formats used by emerging annotation frameworks, in particular, UIMA's Common Analysis System[8]. It is also compatible with tools such as the PDTBAPI, which is easily generalized to handle graphs as well as trees. In addition, the graph model underlies Semantic Web formats such as RDF and OWL, so that any annotation graph is trivially transducable to their serializations (which include not only XML but several others as well), and which, as noted above, has spawned a flurry of research using graph algorithms to extract and analyze semantic information from the web.

A final advantage of the graph model is that it provides a sound basis for devising linguistic annotation schemes. For example, the PB and NB format, although ultimately mappable to a graph representation, was not developed with the graph model as a basis. The format is ambiguous as to the relations among the parts of the annotation, in particular, the relation between the information at the beginning of the line providing the status ("gold"), sense number, and morpho-syntactic description, and the rest of the annotation. Human interpretation can determine that the status (probably) applies to the whole annotation, and the sense number and msd apply to the PTB lexical item being annotated, as reflected in the graph-based representation given in section 3. This somewhat innocuous example demonstrates an all-too-pervasive feature of many annotation schemes: reliance on human interpretation to determine structural relations that are implicit in the *content* of the annotation. Blind automatic transduction of the format to any other format is therefore impossible, and the interpretation, although more or less clear in this example, is prone to human error. If the designers of the PB/NB format had begun with a graph-based model—i.e., had been forced to "draw the circles and lines"—this ambiguity would likely have been avoided.

[8] http://www.alphaworks.ibm.com/tech/uima

7 Conclusion

We have argued that a graph model for linguistic annotations provides the generality and flexibility required for representing linguistic annotations of different types, and provides powerful and well-established means to analyze these annotations in ways that have been previously unexploited. We introduce GrAF, an XML serialization of the graph model, and demonstrate how it can be used to represent annotations originally made available in widely varying formats. GrAF is designed to be used in conjunction with the Linguistic Annotation Framework, which defines an overall architecture for representing layers of linguistic annotation. We show how LAF stand-off annotations in GrAF format can be easily merged and analyzed, and discuss the application of graph-analytic algorithms and tools.

Linguistic annotation has a long history, and over the past 15-20 years we have seen increasing attention to the need for standardization as well as continuing development and convergence of best practices to enable annotation interoperability. Dramatic changes in technology, an in particular the development of the World Wide Web, have impacted both the ways in which we represent linguistic annotations and the urgency of the need to develop sophisticated language processing applications that rely on them. LAF and GrAF are not based on brand new ideas, but rather reflect and make explicit what appears to be evolving as common best practice methodology.

References

A. Cardon and Maxime Crochemore, 1982. Partitioning a graph in O(|A| log2 |V|).*Theoretical Computer Science,* 19(1):85–98.

Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda, 2005. A General Framework for Mining Frequent Subgraphs from Labeled Graphs. *Fundamenta Informaticae,* 66:1-2, 53-82.

Andrew A. Krizhanovsky, 2005. Synonym search in Wikipedia: Synarcher. http://www.citebase.org/abstract?id=oai:arXiv.org:cs/0606097

Dat P.T Nguyen, Yutaka Matsuo, and Mitsuru Ishizuka, 2007. Exploiting Syntactic and Semantic Information for Relation Extraction from Wikipedia. IJCAI *Workshop on Text-Mining & Link-Analysis (TextLink 2007)*.

Dominic Widdows and Beate Dorow, 2002. A graph model for unsupervised lexical acquisition. *Proceedings of the 19th International Conference on Computational Linguistics,* 1093-1099.

Evgeniy Gabrilovich and Shaul Markovitch, 2007. Computing Semantic Relatedness Using Wikipedia-based Explicit Semantic Analysis. *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, Hyderabad, India.

Hang Cui, Renxu Sun, Keya Li, Min-Yen Kan and Tat-Seng Chua, 2005. Question answering passage retrieval using dependency relations. *SIGIR '05: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 400-407.

Jeffrey Dean, Monika R. Henzinger, 1999. Finding related pages in the World Wide Web. *Computer Networks,* 31(11-16):1467–1479.

John E. McEneaney, 2001. Graphic and numerical methods to assess navigation in hypertext. *International Journal of Human-Computer Studies*, 55, 761-786.

Jon M. Kleinberg, 1999. Authoritative sources in a hyper-linked environment. *Journal of the ACM* 46(5):604-632.

Michel Habib, Christophe Paul, Laurent Viennot, 1999. Partition refinement techniques: An interesting algorithmic tool kit. International Journal of Foundations of Computer Science, 10(2):147–170.

Mohammed J. Zaki, 2002. Efficiently mining trees in a forest. *Proceedings of SIGKDD'02*.

Nancy Ide and Laurent Romary, 2004. A Registry of Standard Data Categories for Linguistic Annotation. *Proceedings of the Fourth Language Resources and Evaluation Conference* (LREC), Lisbon, 135-39.

Nancy Ide and Laurent Romary, 2004. International Standard for a Linguistic Annotation Framework. *Journal of Natural Language Engineering,* 10:3-4, 211-225.

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

Razvan C. Bunescu and Raymond J. Mooney, 2007. Extracting relations from text: From word sequences to dependency paths. In Anne Kao and Steve Poteet (eds.), *Text Mining and Natural Language Processing*, Springer, 29-44.

Sergey Brin, 1998. Extracting patterns and relations from the world wide web. *Proceedings of the 1998 International Workshop on the Web and Databases*, 172-183.

Sisay Fissaha Adafre and Maar ten de Rijke, 2005. Discovering missing links in Wikipedia. *Workshop on Link Discovery: Issues, Approaches and Applications*.

Stephen Bird and Mark Liberman, 2001. A formal framework for linguistic annotation. *Speech Communication,* 33:1-2, 23-60.

Vivi Nastase and Stan Szpakowicz, 2006. Matching syntactic-semantic graphs for semantic relation assignment. *Proceedings of TextGraphs: the Second Workshop on Graph Based Methods for Natural Language Processing*, 81-88.