GRAPH-BASED REPRESENTATION AND TECHNIQUES FOR NLU APPLICATION DEVELOPMENT

Juan M. Huerta and David Lubensky

IBM T. J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY, 10598
{huerta,davidlu}@us.ibm.com

ABSTRACT

We describe a method to represent, manipulate, structure and aid in the design and development of NLU oriented parsers. Our method is based on the representation of the semantic parser domain into a single directed graph showing the parser's labels and their immediate inter-relationships as they exist in the annotated development corpora. Furthermore, we present methods developed around this representation illustrating how, a developer can visualize, manipulate, design and construct new applications simply by acting on the domain graph and sub-graphs. We also describe how the graph representation method can be utilized in the reduction of the complexity of the parser by identification and removal of nodes, edges and structures of the domain graph whose impact on attribute accuracy is small. We present the following examples of applications of our technique: extension of an existing air travel information domain to include car rental reservation by manipulating the corresponding graphs, structuring such graphs' vertices into 3-tiers, an example of a method for complex domain decomposition into simpler sub-graphs, and experiments on the reduction of a parser's complexity. Our technique can serve as a foundation of GUI toolkits for NLU development built around these concepts.

1. INTRODUCTION

Natural Language Understanding (NLU) technology is a fundamental component of dialog-based automatic speech understanding systems. Such systems are typically implemented on telephony platforms and are used to automate the communication process between humans and machines through natural speech. An important component of such systems is the semantic parser whose purpose is to recognize structures in the sentence, with the goal to facilitate meaning extraction.

Many of the currently used semantic parsers employ statistical methods to perform their task and thus, need to be trained. The training of a parser normally requires the collection and annotation, or labeling, of large pools of training data. Sets of annotated or bracketed data are usually referred to as treebanks, as the parsing annotation of each individual sentence is represented by a tree.

The set of nodes used in an NLU-system's training treebank typically includes semantic and sometimes syntactic labels of the domain. The set of leaves appearing in the treebank constitute the collection of words.

Currently, there exist methods and tools to manipulate and annotate these treebanks while designing and constructing the parser for an NLU application. These tools present to the developer each sentence and its corresponding parse tree individually, and the annotator sequentially navigates through the treebank one sentence at a time and modifies those trees or constructs new ones (*e.g.*, [2]).

In this work we present a method in which the complete parser domain contained in the treebank can be parsimoniously described in a directed acyclic graph. This representation can be utilized then to manipulate domains, and it allows the developer to be presented with modules that can be processed simply by manipulating the domain graph. Additionally, we will demonstrate that this representation can be utilized to achieve parser simplification. We also propose the use of structured graphs to design new NLU applications: the first tier of nodes representing tasks, the second tier representing topics, and the third tier representing named-entities. Such modular graphs can easily be manipulated to construct novel domains and combine existing ones.

2. GRAPH REPRESENTATION OF NLU DOMAINS

2.1. Directed acyclic graph representation

Let *T* denote an annotated treebank which is the set of annotated sentences $T = \{S_1, S_2, .., S_m\}$ and let $L = \{L_1, .., L_n\}$ be the set of labels (or tags) occurring in *T*.

A directed graph (digraph) *G(V,E)* describes or spans *T* if,

- There is a one to one mapping between the elements of the vertex set *V* and the elements of *L*.

- An edge e_{ij} in E will connect vertices v_i and v_j if there exists at least one adjoined co-occurrence *of* v_i and v_j in T.
- A unique terminal edge *END* is connected to all the edges whose label realizations in T encompass no other labels.

We can further constrain the domain to avoid loops in the graph thus making the domain representable by a Directed Acyclic Graph [6]. This implies that no node can have itself as an ancestor.

2.2. Associating weights to edges

If *G(V,E)* describes or spans T , an edge e_{ij} in E can be associated to a weight $w_{i\,j}$ using,

- **Counts**: The number of times c_{ij} in T of adjoint co-occurrences of v_i and v_j.
- **Relative co-occurrences**: The above counts normalized by the total number instances of children leaving v_i in T.

2.3. Example: Air travel information domain

As an example we refer to figure 1.a, depicting a graph of a simple air travel domain in which a user can utter queries related to flights based on their dates, times and cities of departure and arrival. Figure 1.c, shows the small 5-sentence annotated sample corpus T that generates graph 1.a. An edge exists between the node *arr* (meaning arrival) and *date* but not between *air-inf* and *date* because in the corpus, the label *date* occurs immediately below *arr* at least once, while *date* doesn't occur immediately under *air-info*, and so on. In this way, we can represent in a single graph the labels occurring in the corpus depicted in 1.c and their immediate interrelationships. The root label in T (represented in this case by *!S!*) is the starting vertex of the graph. The terminal labels (those whose children include no other labels, only words) will correspond to vertices in the graph having a single outgoing edge that will go to the ending node (represented by *END*). In figure 1.b. the edges in the graph are labeled with relative label co-occurrence information. For example, the edge connecting the nodes *arr* and *time* is labeled with the weight 0.28, which corresponds to the ratio of the counts of the co-occurrences of those labels (2 counts) over the total number of children of node *arr* (7 counts), and so forth.

3. GRAPH-ELEMENT MUTUAL INFORMATION CONDITIONED TO GRAPH STRUCTURES

3.2 Graph-element mutual information

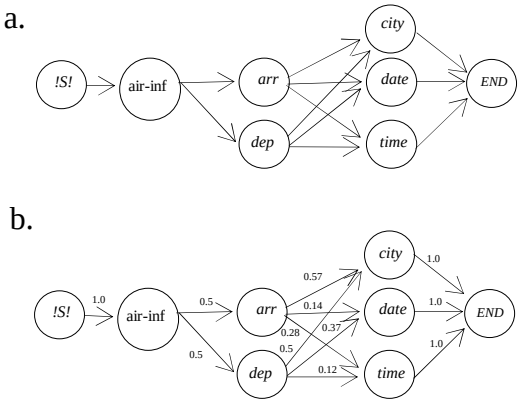
Consider now the graph *G(V,E)* representing an NLU domain and spanning the treebank T . An annotated utterance S_i and its associated parse tree are said to be a *realization* of G since the tree (minus the leaves) and the graph corresponding to S_i (or any other subset of T) are subgraphs of G.

Let us define a binary valued Random Variable X which takes the value 1 if a realization (*i.e.*, a single sentence, for this specific case) of G contains a vertex v_i ; and 0 if it doesn't. Similarly, we can define another Random Variable Y which will take values 0 or 1 depending on the existence of another vertex v_j. Based on G we can compute the Mutual Information between X and Y, or $MI(v_i,v_j)$ as follows:

$$MI(X,Y) = H(X) - H(X \mid Y) = \sum_{v_i,v_j} p(v_i,v_j)\log\frac{p(v_i,v_j)}{p(v_i)\,p(v_j)}$$

The mutual information between these two vertices will be a measure of their independence [3,4].

The random variables X and Y cannot only be vertices (as we have described), but they also can be edges, or, in general, any subtree or structure g_k occurring in G. We can, for example, measure the mutual information between an edge e_{ij} and a vertex v_k and $MI(e_{ij},v_k)$ reflecting the level of probabilistic dependence between both structures. If we annotate each sentence in T with their corresponding semantic attributes, we can measure the mutual information between any graph structure g_i and the presence of a given attribute a_j, *i.e.*, $MI(g_i,a_j)$.



c.

[!S! I want to know of *[air-inf* any flights *[dep* departing *[city* New York *city] [date* today *date] dep]]air-inf] !S!]*

[!S! Information on *[air-inf* flights *[dep* from *[city* London *city] dep] [arr* to *[city Paris city] arr] air-inf]]!S!]*

[!S! Which *[air-inf* flights *[arr* get to *[city* Seattle *city] [time* before noon *time] [date* this Saturday *date] arr] air-inf]]!S!]*

[!S! Are there *[air-inf* any flights *[arr* to *[city* Pittsburgh *city] arr] [dep* from *[city* Boston *city]* departing *[date* today *date][time* after ten *time] dep] air-inf]]!S!]*

[!S! [air-inf flights *[dep* from *[city* Chicago *city] dep] [arr* to *[city* Houston *city] arr] [dep* leaving *[date* this Sunday *date][time* morning *time] dep] [arr* arriving *[time* before noon *time] arr] air-inf]]!S!]*

Figure 1. (a) Graph representation of the air information domain corpus (b) weighted graph (c) corresponding generating corpus.

3.3. Graph-structure conditional mutual information

Similarly, we can also compute the mutual information between graph-structures *X* and *Y*, conditioned on the value of *Z*, where Z is also any graph-associated structure (*i.e.*, edge, vertex, sub-graph, or attribute). The conditional mutual information between g_i and g_j conditioned on g_k is defined as:

$$MI(X,Y \mid Z) = \sum_{g_i,g_j,g_k} p(g_i,g_j,g_k)\log\frac{p(g_i,g_j \mid g_k)}{p(g_i \mid g_k)p(g_j \mid g_k)}$$

Similarly to the case of the vertex based Mutual Information described above, we consider graph-associated structures to have binary values: with value 1 if the realization T_j contains g_k, and 0 if it doesn't. The use of Mutual Information to discover non-random associations in data patterns has been applied in other domains (*e.g.*, [3,5]). In the next section we describe the application of Mutual Information to the simplification of domain graphs.

4. PRACTICAL APPLICATIONS

4.1. Synthesis of novel domains

Let $G_1(V_1,E_1)$ represent a domain T_1, and $G_2(V_2,E_2)$ represent a different domain T_2; we describe the following operations and relationships between T_1 and T_2:

- **Domain synthesis**: A new domain G_3 can be defined as the union of G_1 and G_2, or $G_3 = G_1 \cup G_2$; where the edge and vertex sets of the new domain are also the union of the simple domain's corresponding conforming sets. An algebra of graphs can be similarly defined for the intersection and the complement of domain graphs. The union operation allows the developer to define new domains using simple arithmetic expressions.
- **Isomorphic domains**: G_1 and G_2 are said to be isomorphic if there exists a one-to-one correspondence between their vertex sets as well as between their edge sets [6]. Isomorphisms across NLU domains can be exploited for domain or parser bootstrapping, at the parser level or at the treebank level by means of a simple remapping of labels and/or words.
- **Canonical forms**: Rules and guidelines can be established for the design of graphs in a way that allows modularity and reusability. Domain graphs' vertices can be then structured in well defined layers (*e.g.*, a 3-tier topology: tasks layer, topics layer, and named entities layer; described in the next paragraph) that will facilitate the existence of isomorphic graphs.

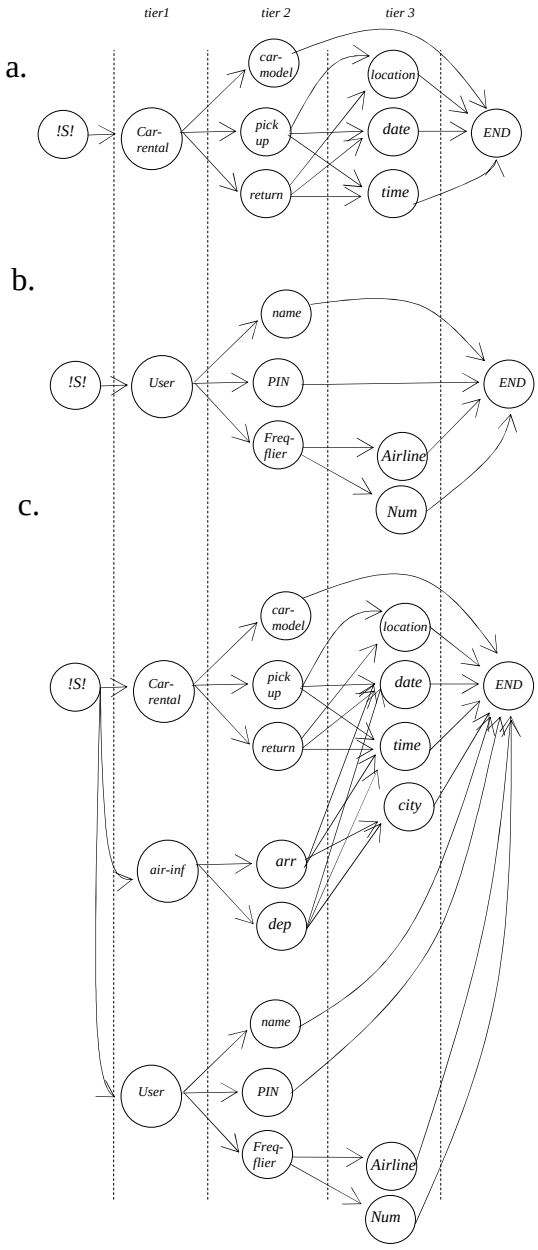


Figure 2. (a) Graph of a car rental domain, represented in 3 tiers. (b) Graph of a customer service support domain (c) Graph of the union of the air-travel information, car rental and customer service support domains. The final graph maintains the 3-tier topology.

Figure 2 illustrates the above concepts; figure 2.a shows the graph of a car rental domain in a 3-tier topology. It is easy to observe its topological similitude with the air travel information domain, particularly on the nodes pertaining date, time and location. Figure 2.b shows a general customer care support domain. Figure 2.c shows the union of these domains with the air travels system shown in figure 1. One can observe how, by designing graphs in a well structured

way, the resulting domain preserves a reasonable topology. Thus, for this graph, the first tier of nodes after the root correspond to task related nodes (*i.e.*, car-rental, air-inf, customer care) the second tier correspond to semantic topics (*i.e.*, pickup, return, arrival, departure, car model) which usually are associated with the forms in a form based NLU system. The third tier shows general named entities which are usually task independent (city, time, date, rental location, airline name, number). Designing a domain using such canonical topologies (task-topic-entity) permits easy domain modularization and integration.

4.2. Analysis of existing domains

Mutual Information can be computed between structures in the set of realizations *T* of the graph and the semantic attributes associated with each tree realization (*i.e.*, an attribute a_i is associated with sentence S_j if it contains information pertinent to slot *i*). We then can rank the elements of the domain based on the Mutual Information they share with a given attribute, then build a corresponding subdomain graph by first selecting the subset of high ranking vertices and then selecting the edges in *G* that link them.

A more straightforward approach selects the subset of *T* whose sentences invoke the given attribute and then simply construct the subdomain's graph based on the labels and trees of such subset. In either case, the resulting subgraph represents a topic or subdomain of the overall domain, which is a modular representation of the subtopic.

4.3. Parser reduction techniques

We conducted domain reduction experiments based on the graph of the IBM 401k NLU system [1]. This application contains a parser employing 123 word tags and 177 labels. The graph of such domain has 569 arcs, and the average tree depth is 5 nodes deep. We pruned first the vertices below the threshold value, and then removed the edges that became disconnected from the root edge. Table 1 shows the effect of different thresholds on parser size and accuracy (total test set is approx 4100 sentences). We also reduced the number of tags (associated with words) in the domain by removing tags whose average MI with the set of attributes was small. The combined reductions substantially diminished the complexity of the parser while preserving the parser accuracy. Excessive tag and label removal, however, decreases the presence of features able to trigger attributes: the last column shows how the Attribute-Value accuracy suffers (in the training set) if the label and tag set of the parser, and thus the annotation style, is substantially simplified. The number of arcs is reduced to less than half of its original size.

5. CONCLUSIONS

We have introduced a method to represent an NLU domain in a graph. Such representation facilitates the design, composition and analysis of novel and existing NLU parsers and applications. The application of Mutual Information measurements to structures in the graph permitted further analysis and simplification of the graphs; we demonstrated this procedure by reducing the number of labels and arcs in a graph substantially without major impact in the parser's accuracy. We believe that such graph representations, combined with the appropriate GUI tools and set of established canonical parser forms (*e.g.*, 3-tier graphs) and guidelines for the design and structuring of such modular domain graphs would simplify substantially the design and development process of complex NLU applications.

6. REFERENCES

[1] Balchandran, R. *et al. "A Statistical Mixed-initiative Dialog System for 401k Management"*, Demonstration presentation in Human Language Technology Conference 2002, San Diego.

[2] Brew, C. *"An extensible visualization tool to aid Treebank exploration"* in H. Uszkoreit, editor, Linguistically Interpreted Corpora, Bergen. EACL Post-conference workshop.

[3] Butte, A. J. and Kohane, I.S. *"Mutual Information Relevance Networks: Functional Genomic Clustering using Pairwise Entropy Measurements"*, Pac. Symp. Biocomput, 2000.

[4] Cover, T. M. and Thomas, J. A. *Elements of information theory*, John Wiley and Sons Inc.

[5] Meng, H. *"Semiautomatic Acquisition of Semantic Structures for Understanding Domain-Specific Natural Language Queries"*, in IEEE Transactions on Knowledge and Data Engineering Vol. 14 No. 1 Jan-Feb 2002.

[6] Swamy, N.M.S. and Thulasiramant, K. *"Graphs, Networks and Algorithms"*, John Wiley and Sons Inc.

Table 1. Domain simplification experiments based on graph pruning and vertex-based Mutual Information.

	Number of tags	Number of labels	Number of arcs	Avg. tree depth	Parser acc. (sent. correct)	**Attribute-Value Accuracy (train set)**
Baseline	123	177	569	5.0	3836	**100% (ref)**
Prune @ 0.045	123	98	232	2.7	3828	**97%**
Prune @ 0.060	123	88	211	-	3797	**97%**
Prune @ 0.075	123	70	175	-	3319	**95%**
Prune @ 0.045 + M. I.	**72**	**98**	**232**	**2.7**	**3802**	**97%**