Representing Syntactic-Semantic Knowledge from English Texts

Richard Guidry Jr. and Jianhua Chen
Division of Computer Science and Engineering
School of EECS, Louisiana State University
Baton Rouge, LA 70803, USA

Abstract - We present a unified representational framework \textit{LSeN} (Language to Semantic Network) for representing syntactic-semantic knowledge extracted from natural language (e.g. English) texts. Our representation is a variant of the standard semantic networks with some important distinctive features that facilitate bridging the interface between syntactic and semantic knowledge using automatic computational tools. We show the basic constructs in our representation and briefly describe the developed tool for automatically building the knowledge structure in \textit{LSeN} from parsed English sentences. Our representational framework and the associated tool establish a good platform for further studies in automatic knowledge extraction from texts.

Keywords: Semantic Network, Syntactic and Semantic Knowledge, Natural Language Processing

1 Introduction

Natural Language Processing (NLP) has achieved great progress in the last two decades. With the rapid development of research in this area, nowadays we have various computational tools for extracting knowledge from natural language texts represented in one form or the other. Developing a suitable representation formalism for syntactic/semantic knowledge from texts and subsequently designing an efficient computational tool for actually extracting knowledge into the desired form (structure) are important research problems tackled by numerous researchers in the NLP field.

In natural language processing tasks (such as language understanding, query-answering and text summarization), we need first perform syntactical processing of the natural language texts through parts-of-speech tagging and syntactical parsing. Then semantic processing is performed on the syntax parse trees to obtain semantic knowledge. Clearly there is a close connection between the syntactic knowledge and the subsequent semantic knowledge, as both could be represented as graphs/trees (parse trees) of some sort, with nodes representing words and edges labelled by dependencies/relationships between nodes, as remarked by some researcher in the “research gate” social media website [12].

We believe that a representational framework facilitating the incorporation of both syntactical and semantic knowledge is beneficial to the natural language understanding task. Syntactic knowledge often provides very good clues to the semantic aspects of sentences. Many shallow semantic processing techniques such as semantic role-labeling, conceptual dependency analysis, etc., make extensive use of the close connection between syntactic and semantic knowledge. The availability of various syntactic parsers such as the Link grammar parser [13], the Stanford Parser [14], and the Malt Parser [15] made it much easier to exploit the results of syntactic parsing in building shallow semantic models.

In this paper we present a framework for representing syntactic-semantic knowledge from English text with the objective to make it easier to utilize the close connection between syntactic and semantic knowledge in NLP tasks. Based on this model, we have constructed a computational tool for automatically building a semantic-network-like structure from English sentences. This forms a good basis for performing query-answering with the constructed semantic structures.

What we present is a generic language knowledge interface which is domain independent. In the arena of computer systems handling natural human language, we often meet challenges that require extensive domain-specific knowledge in order to achieve high level of performance. The current systems for NLP seem to fall into two categories. The first category is domain specific. These cater specifically to a small domain and work well in that domain only. The second category is systems that require priming with hand fashioned databases of knowledge. These systems often require such an amount of perquisite data that it is difficult to produce a sufficient base from which to get the desired results. We propose a system which does not deal in the restrictiveness of these categories. The system could operate without hand fed data, and operate in any domain. The proposed system representation is outlined (Section 3) after the discussions in Section 2 on relevant works and existing technologies.

2 Background and Relevant Works

Natural Language Processing (NLP) is the field which lies at the junction of Linguistics and Computer Science, also commonly known as computational linguistics. The field had its beginnings in the 1930’s with the creation of mathematical and logical formalisms, which were the predecessors to
current linguistics theories, and also modern computer languages. NLP is concerned with the interactions between computers and human (natural) languages. As such, NLP is related to the area of human–computer interaction. Many challenges in NLP involve natural language understanding, that is, enabling computers to derive meaning from human or natural language input, and others involve natural language generation. Other major tasks in NLP include automatic text summarization, discourse analysis, machine translation, question-answering, etc.

2.1 Stages of Language Processing

The processing of natural language can be divided into stages as seen here [5].

![Fig. 1: Stages in NLP](image)

Lexical analysis and tokenization encapsulate linguistic elements into tokens, usually words. These tokens are provided in a stream to the next stage. For audio input, a speech recognizer would provide the token stream. The syntactic analyzer or parser takes the token stream, and derives a hierarchical syntax tree. Parsers may be based on a grammar specification, which generates a parse table; or they may be based on statistical methods as the Maximum Entropy Markov Model (MEMM) parser [11].

Semantic analysis is a broad step, where information is extracted from the syntax tree. The extracted information may then be analyzed pragmatically for application of context. Between semantic and pragmatic analyses, tasks need to be performed such as resolving references, disambiguating words, and resolving the scope of modifiers. Through interaction with the knowledge system, the result of semantic and pragmatic analyses is encoded into some body of information that can be consulted at later times. If an inquiry is made, it will be necessary to move to the language generation stage.

2.2 Graph/Semantic network-based Approach for Knowledge Representation in NLP

The focus of this paper is on knowledge representation for syntactic and semantic knowledge which are essential for natural language understanding and question-answering. The approach taken in the representation framework is graph-based, but it has close connections with the symbolic/logic-based approach to natural language knowledge representation. We can see three distinct approaches to NLP tasks: The Symbolic/logic-based approach, The Graph/Semantic network-based approach, and The Empirical approach. Symbolic NLP deals with the more traditional logic-based methodologies, where semantic knowledge is represented by a set of logic formulas, and algorithms operate on the symbols to perform inference and answer queries. Empirical NLP uses corpora of language data which are processed by a system to extract syntactic/semantic knowledge which would then be used for various NLP tasks such as Parts of Speech Tagging and semantic similarity computation. The graph/semantic network-based approach represents the syntactic/semantic knowledge in natural languages by various graphs or semantic networks. Algorithms for graph processing (traversal, path-finding, homomorphism, etc.) are utilized to perform inference and query-answering. Our proposed representational framework is a graph-based approach and so we discuss below in some detail the relevant existing works in graph/semantic network-based knowledge representation for NLP.

John Sowa [19] in 1976 proposed the Conceptual Graph (CG) as a common knowledge representational framework. Conceptual graphs are closely related to Pierce’s Existential Graphs (EGs) [25] which were developed by Pierce in 1896 as a simple, readable, expressive graphic notation for First Order Logic (FOL). Conceptual Graphs combine a logical foundation based on EGs with the structural features of Semantic Networks [21] in providing an expressive and formally grounded formalism for representing knowledge [24]. In Conceptual graphs, concepts are represented by rectangles and relations are represented by ellipses (ovals). Links (directed or undirected) indicate the connection between concept nodes and relation nodes. The Conceptual Graph Interchange Format (CGIF) is a linearization of CGs specified by the ISO/IEC standard 24707 for Common Logic (CL). CGIF essentially specifies the equivalent common logic expression corresponding to a conceptual graph.

As an example, the sentence “If a farmer owns a donkey, then he beats it” is represented in CG as in the following Figure 2. In Figure 2, the concepts “Farmer” and “Donkey” are drawn as rectangles. The “Owns” relationship between “Farmer” and “Donkey” is shown by an ellipse with links, indicating that “Farmer” is the subject of the “Owns” relationship and “Donkey” is the object in that relationship. Similarly for the “Beats” relationship.
Conceptual graphs are closely related to semantic networks [21] that have been widely used in Artificial Intelligence and NLP research. According to Sowa [20], “A semantic network or net is a graph structure for representing knowledge in patterns of interconnected nodes and arcs. Computer implementations of semantic networks were first developed for artificial intelligence and machine translation, but earlier versions have long been used in philosophy, psychology, and linguistics.” Semantic networks are intended for a graphical and declarative representation of knowledge, and should support automated reasoning systems to perform inference on that knowledge. Some semantic networks are defined quite informally for the sake of human cognitive purposes whereas some do have formally specified semantics - and intended for computer automatic inference. For example, the system KL-One (Knowledge Language One) [27, 28] specifies a subset of the classical First Order Logic (FOL), in which nodes represents concepts (types or instances of types) and labeled arcs represents roles (attributes) and relations. Generalization (“is-a”) hierarchy can also be represented in KL-One.

The representational system we propose in this paper is also a semantic network. Our proposal is different from Sowa’s conceptual graphs in several ways. Our system is designed with the intension to work with NLP tasks and not necessarily as a generic knowledge representation tool, a category CG would be classified into. CG has a formal semantic grounding in FOL (with the flexibility to be extended to handle natural languages and meta-languages) whereas we are yet to work out the semantic aspects of our system (which we believe should have a logical underpinning). We have introduced various types of nodes (Class Restriction and Derived Class, etc.) in our semantic network which seem to be quite similar to some of the graph operations (e.g., “restrict”/”un-restrict”) in CG.

2.3 Approaches Addressing the Connection between Syntactic-Semantic Knowledge

Keep in mind that the key motivation of the current work is to develop a representation formalism that would facilitate the use of connections between syntactic and semantic knowledge in NLP. Similar efforts can be seen from the works in [16, 18]. In [16], the long-term project is the use of Transparent Intentional Logic (TIL) as a semantic representation of knowledge and subsequently as a transfer language in automatic machine translation. In [18], researchers focused on the mapping between syntactic and semantic knowledge, applied hyper-graph homomorphism for language comprehension and constraint satisfaction for language generation. One can also see the various works in computational linguistics and NLP using, for example, dependency grammars and conceptual dependency graphs (Schank [26]). Clearly the conceptual graphs approach could be seen as addressing this issue.

3 The Proposed Representation: the LSeN Interface

LSeN stands for Language to Semantic Network. It is our automatic conversion tool that takes an English sentence as the input and encodes it into a semantic network. It currently works with generic sentences in the present tense. The figures shown in this section are automatically generated by our system.

3.1 LSeN Encoding Structure

The LSeN encoding structure strives to represent general English language with as basic a tool set as possible. This is done in the form of a semantic network with node types and connections as follows. Classes represent a set or predicates. This includes adjectives, verbs, and nouns that are types. An adjective as a class represents all things to which that adjective applies. For verbs, any subject to which that verb applies generically is in the class. For nouns that are types, its class just represents anything that is considered that type. These classes may be related by subset links, denoted as is-a. Classes that we have words for are named as such.

![Figure 3. Basic Class Relationships](image-url)
However, classes may be composed of other classes. A class which is the intersection of two other classes is said to be a restricted class, and is formed with a class restrictor (CR) node. CR nodes derive their meaning from their two out-links, labeled is-a.

![Fig. 4. Class Restriction (CR)](image)

The above Figure 4 is obtained after processing sentences “Mammals are furry animals.” as well as a few others (such as “dogs are mammals”). CR nodes are static; they are created with their two links that do not change throughout its lifetime. This does not enable us to say something new about a class represented by a CR (we call it an unqualified class because of this), so another node type is used. The derived class (DC) node is pinned to a CR, linked with an equal sign. The DC is qualified and may be assigned a superclass (an is-a link) just as named classes, which is what allows us to express new information about a CR represented class. Instances are determined objects that are members of some class. Each instance has one out-link labeled inst-of, which is linked to the most specific class node that this object is known to be in.

![Fig. 5. Derived Classes (DC) – “big dogs are mean”](image)

The DC Verbs are represented as classes; however, a class node that represents an intransitive verb may have instances linked to it while a class node that represents a transitive verb may not. Transitive verb classes must first have a direct object argument bound via a BIND node. BIND nodes represent unqualified classes, similar to CR nodes.

![Fig. 6. “Roy is a dog” – Fragment “Roy is a ”](image)

The information structure heretofore discussed is intended to facilitate being built from a syntactical parse. In that process, we obtain intermediate stages of representation (fragments) which start close to the parse. The fragments are then integrated together, up to a top fragment which can be merged into the network.

![Fig. 7. “Roy likes Jim” – Fragment “likes Jim”](image)

After a sentence has been processed into a subject and predicate fragment, we represent the notion that these concepts have been unified with a unified concept (UC) node, bearing a link to each concept; specifically the uc-a link is to the subject, and the uc-b link is to the predicate. The order is not reversible. The unified concept represents three different notions depending on the nature of subject and predicate links, and it does so in the same manner as to be. In the case of unifying two instances, it represents equality; for two classes, a subset relationship; and for instance and a class it represents membership.

![Fig. 8: Network1 obtained after the sentences.](image)

We further illustrate the various constructs in the proposed semantic network by some examples. Consider the sentences “Roy is a dog. Jim is a cat. Roy likes Jim. Jim ate a mouse.” Figures 6, 7 and 8 show the fragments of the semantic network constructed in processing the above sentences.
Consider the sentence “Roy saw the mouse Jim ate” which was presented to the LSeN system after the network has been constructed. We see the following fragment (Fig. 9):

![Fig. 9. Fragment “hypothetical-mouse_whom_Jim_ate”](image)

Note that here hypothetical is used to match the right mouse instance that is seen by Roy that Jim ate. Hypothetical (inst2_mouse...) is filled in, and can be tested to see if inst_2 is that mouse. Network2 is the result semantic network after processing the last sentence “Roy saw the mouse Jim ate”.

![Fig. 10. Network2 obtained after “Roy saw the mouse Jim ate”](image)

The last type of node, \( \eta \), represents a hypothetical, which can be matched against the network and evaluated for truth. These can be used to match definite noun phrases to an object already in the tree if there is one. Note that the previously discussed UC node could be represented as an \( \eta \), though the nodes under the \( \eta \) would be different in the each of the three cases discussed.

### 3.2 Semantic Analysis in LSeN

LSeN’s chief function is to perform semantic analysis on user input in order to generate a semantic network. Before semantic analysis can begin, however, it is required that the input be processed into a syntax tree. An MEMM parser from OpenNLP [17] is used for this. Once LSeN has the syntax tree, it determines additions to the semantic network. Handling functions are recursively called on the syntax tree. Handling functions produce a result on each node in the syntax tree, and that result is a reference to a node in the semantic network. Each type of node in the syntax tree has a separate handing function.

For example, let’s take the noun phrase “the dog”. It is composed of two child nodes: the determiner “the”, and the noun “dog”. The noun phrase handler processes its children, and incorporates them together. It requests the result of handling dog. The handler for nouns processes “dog”, and produces a result which is the class node for dogs. The noun phrase handler now holds a reference to the class of dogs. The determiner tells the noun phrase handler to find the unique thing that is an instance of the class denoted by the rest of the noun phrase. This has the effect of finding the unique instance of a dog in the semantic network.

The previous example demonstrates the workings of noun phrases in the system. Verb phrases are handled similarly. Transitive verbs take the result of processing the noun phrase direct object, and bind it as an argument to the verb to yield the result. Intransitive verbs are simply returned as a class. Once at the top of the sentence, the resultant semantic network nodes of handling the subject and predicate are connected. This usually involves creating a subclass relationship, except in the case of equating two instance objects.

### 4 Conclusions and Future Work

#### 4.1 Contributions

In this paper we present a generic framework for representing syntactic and semantic knowledge for NLP, along with a computational system for automatically constructing the knowledge structure (semantic network) from syntactic parse trees. The proposed framework is generic (domain-independent), and facilitates the mapping between syntactic and semantic knowledge and the tool provides a good starting point for further experimental studies in various tasks of NLP such as co-reference resolution and question-answering. Although the system is in its primitive stage, building such a system is an important step towards further development.
4.2 Issues

The system is currently in its very early stages, and does have problems as it is. Currently, it would be impossible to handle relative clauses with a direct object binding, but with an unbound subject, as there is no way to represent this in the semantic network. Rather than adding another structure, we hope to modify the BIND node concept to be more flexible, or perhaps even replace the BIND nodes. It is a goal of the system to maintain minimum complexity in the semantic network, which is why we would prefer not to add multiple types of binders. The system can also answer textually posed questions regarding the semantic net, though it has not been tested or updated in the last wave of expansion. With this being the target user interface, rather than visually inspecting the graph, it is imperative to update it to be current with the rest of the system. The MEMM parser is sometimes a bit off in generating the syntax tree. Means need to be developed to handle this gracefully. Ambiguities and failed determiner references also need graceful handling.

4.3 Future Work

Considerable future work can be done, but here are some specific and realizable tasks. It needs to be resolved how to incorporate knowledge represented by DC nodes into determiner resolution and similar tasks. For instance if Rex is a mean dog, and Mean dogs are big, then does the big dog reference Rex? Prepositional phrases need to be added. It is not a problem to represent them in the semantic network, as they would work like transitive verbs; however, they are often ambiguous in scope. This leads to a whole new problem of scope resolution, which is not completely handled in the MEMM parser. Adverbs could also be added, but with all the problems of prepositional phrases in addition to needing a semantic network representation. A whole system of handling spatial-temporal information, verb tense, modals, and, consequently state vs. individual-level predicates can be added, but not before the current system is refined.

5 Reference


