Generating Non-Intentional Semantically-Correct English Sentences in an Object-Oriented Environment

Khosrow Kaikhat and Craig Gandy
Department of Computer Science, Southwest Texas State University, San Marcos, TX 78666, USA
E-mail: kki02@swt.edu
Generating Non-Intentional Semantically-Correct English Sentences in an Object-Oriented Environment

Khoshoo Kaklakhi and Craig Gandy
Department of Computer Science, Southeast Texas State University, San Marcos, TX 78666, USA
email: R02@cc.utm.edu

Key words: Natural language processing (NLP), natural language comprehension, natural language generation, object-oriented design, COMMON LISP object system, object fiction, structural knowledge, contextual knowledge, benchmark evaluation tool

Abstract

When a human being is constructing a sentence, he or she chooses words that not only create syntactical integrity but also satisfy semantic and in most cases contextual constraints. The system developed here mimics the way human beings construct sentences. In our approach a sentence is generated as a chain of words satisfying a number of semantic, syntactic and contextual constraints concurrently. This chaining process ensures the compatibility of different components and results in a cohesive and unambiguous sentence. The static lexical and structural objects which contain both semantic and syntactic knowledge interact with dynamically created objects which contain contextual knowledge in order to construct the word and phrase objects which ultimately construct the sentence.

1. Introduction

Computers are a major part of today's society. They are used everywhere ranging from home use to sophisticated research and development laboratories. One of the major barriers to wider acceptance and usage of computers can be attributed to the restrictive methods of interaction between human and computer. It is expected that in the near future most computer software systems will contain an NLP component as part of their means of communication. The ultimate goal of NLP research has been to enable computers to comprehend and converse in a natural language. NLP research can be classified into two overlapping areas: NL comprehension and NL generation. NLP comprehension is concerned with the human to computer dialog, whereas NL generation is concerned with computer to human discourse.

Our research has been concentrated on NL generation, more specifically, the non-intentional generation of English sentences. The goal of non-intentional generation is to construct a semantically meaningful sentence, rather than to express a desired meaning. The sentence generator described in this article originated as a major component of a fully automated natural language evaluator based on the BenchMark Evaluation Tool (Real et al., 1992). However, its scope can be extended to generate sentences from an abstract representation of the intended meaning. The extension enables the sentence generation system to be applied to other NL generation applications, including machine translation, language learning assistance, question answering, and scene description (Mycocke, 1991).

We adopted an Object-Oriented Design (OOD) strategy (Booch, 1991). In our approach all concepts, including semantic and syntactic units, are defined as objects. Each sentence is generated as a chain...
of words satisfying a number of semantic, syntactic, and contextual constraints. The constraints imposed on the generation process influence dynamically while the sentence is being generated. This strategy guarantees semantic coherence while maintaining syntactic integrity by preventing generation of semantically incorrect sentences. Semantically incorrect sentences are those where a sentence is constructed from the available vocabulary that can satisfy the suggested syntactic structure, but is not semantically meaningful. For instance, the sentence: does the Atlanta flight have a stop in Boston? may be generated. Instead, the sentence: does the Atlanta to Boston flight have a stop in Boston? may be generated.

In our approach, the knowledge in logical form and the logical interface are implemented as separate modules. Each module represents an independent knowledge source with local knowledge that can decide whether it can be a part of the sentence being generated, called upon by the sentence generator to join the chain. This approach exposes semantic constraints and eliminates generation of grammatically correct but semantically incorrect sentences.

In the following sections, we will describe the object-oriented approach, design, and implementation of the system. We will describe the objects that were used in generating sentences and the way they interact to constitute the generation process. We will highlight the major design and implementation issues and demonstrate the internal behavior of the system with a detailed example.

2. Design and Methodology

Our objective has been to design and develop a system that would generate sentences that are grammatically correct and semantically correct. The sentences were to be built according to a grammatical structure or template, and would be non-intensional. In other words, there was no desired interaction specified for the sentence. Instead, Object-Oriented Design (OOD) (Booch, 1991) techniques were used to develop a system where objects that represent knowledge interact with dynamic elements of context in order to recursively build a sentence piece by piece. In this interactive mode, objects that have already been built are used to adjust the syntactic and semantic requirements of future sentence objects. We chose an object-oriented approach for the following two reasons:

Knowledge can be divided into two general categories: static or functional. The object-oriented framework provides a convenient way to combine the two on the same data structure: an object.

In a natural language, sentences are controlled by different forces at different levels. For instance, at the top level, a complete sentence is controlled by its main verb. Each phrase is also dominated by different forces. For instance, in a common phrase, the noun determines the type and number of words, and in a prepositional phrase, the preposition is the driving force. We realized that natural language is controlled by the generation of words within each phrase, and pressure within each sentence. This approach proved to be extremely efficient due to its robustness. By choosing the most dominant and least limiting component of each sentence at each level, we have reduced the need for exhaustive backtracking and have increased the expressive power of the system. This approach differs from the sequential approach employed in PARSER (Ballin and Thomson, 1988) and ALICE (Green, Evans, and Gates, 1991). In addition, the use of object-oriented methodology allowed us to implement our generation algorithm by facilitating the dynamic generation of constraints.

This section describes in detail the design and implementation of our system.

2.1 System Overview

Randomly generating a sentence from a template with a fixed vocabulary can yield many invalid and unclear combinations of words. Obviously, not all words can be combined to create a meaningful sentence. The central idea of our system is that the decisions, which are made to form part of the sentence must limit the available choices for other parts of the sentence.

We view the process of generating a sentence as a series of phrase generation sub-goals. Furthermore, each phrase generation is divided into word generation sub-goals.

The generation of the phrase comprising a sentence is performed on a hierarchical basis, based on their semantic role. The hierarchical order is as follows: action, agent, patient, recipient, attribute, manner, auxiliary, literal (Sowa, 1988). This ordering ensures that the objects that have the greatest influence on the
Contextual Knowledge Guide: The three areas (sentence, phrase, and word) that comprise the contextual knowledge guide are built from objects that have already been created. These dynamically created objects, along with the contextual knowledge rules, regulate the generation process of subsequent parts of the sentence.

2.1.2. Syntax objects

The system is implemented in an object-oriented environment. The objects utilized in the system are described below:

Template Objects: The syntactic knowledge obtained from the Benchmark Evaluation Tool is stored in template objects (Neel et al., 1992). This knowledge is organized in a tree hierarchy with branch nodes and leaf nodes. The leaf nodes contain the template name, an internal parsed representation of the template, selection flags, and sentence level constraints.

Expansion Objects: A template is a collection of grammatical patterns, each pattern (such as [NP]) may have multiple ways that it can be satisfied. The expansion of the patterns found in the templates are stored in expansion objects (Thomas and Martinet, 1985). These objects are also organized in a tree hierarchy. The leaf nodes for these objects are specific to the object, the root node from the bottom up have generically general patterns.

Lexical Objects: These objects are the words that make up the system's vocabulary. They store semantic, syntactic, and morphological information. The syntactic elements include parts of speech, and any syntactic variation associated with a word. The morphological information is used to form the word (what it deviates from the normal rules of English orthography). The semantic information is used to ensure semantic agreement; it contains conceptually entities that are part of the sentence (Gazdar and Milne, 1989, Thomas and Martinet, 1985).

Constraint Objects: Constraint objects are built for every phrase and for every word. They contain the information to be used to limit the generation. These objects have slots for the type, number, person, tense, form, case, gender, concept, and quantifier (Thomas and Martinet, 1985).
Sentence Objects: The sentence objects contain the instructions for building the sentence and the final version of the sentence. Sentence object sub-classes include declarative, imperative, interrogative, and clauses.

Phrase Objects: The phrase objects are the most complex objects in the system. Most phrase objects either from semantic, syntactic, and phrase-type super-classes. For instance, the phrase object, agent-verb, is a combination of agent (semantic role), subject (syntactic role), and noun phrase (phrase type). The slots found in any one phrase object will vary, however, any information that could be needed later in the generation process is stored in a phrase object (first, 1987; Kees, 1989).

Word Objects: The word objects store the expansion patterns for the words, the lexical object selected, and the final version of the word. Each part of speech recognized in the system has a word object sub-class.

2.2. Temporal and Expansion

The temporal and expansion objects store the structure upon which a sentence is constructed. They serve as a blueprint and a scaffold for the sentence. The template objects contain the instructions for building a sentence from phrases, while the expansion objects contain the instructions for building a phrase from words. Both types of objects are organized along the same basis. Their class structure (kind-of) involves a super-class with branch node and leaf node sub-classes, while their object structure (part-of) is a tree hierarchy with more specific knowledge contained at the leafs.
Fig. 2. Partial template hierarchy tree.

Fig. 3. Partial expansion hierarchy tree.

(Broad, 1994). This organization allows for a variety of general concepts, while still maintaining the ability for specific reasons.

The template superclass object consists of two slots: a label for naming the object, and a selection flag, a bit vector which encodes its category. The template branch node objects contain an additional slot for children of that node, while the template leaf node objects (Figure 3) contain additional slots for a textual template, the sentence type (imperative, declarative, interrogative), the template parse, and any sentence level constraints.

The textual template represents the syntactical grammar of the template. Sentence level constraints
are a partial realizations of the sentence formation. The template parse is a generalization representation of the phrases within a template. In this representation, a sentence is viewed as a collection of phrases. Each phrase has semantic knowledge as well as the expansion objects which are used in generation process. The semantic knowledge describes the role each phrase plays within the sentence and is used to avoid structural ambiguity and to prioritize the phrase generation within a sentence. The possible semantic roles are agent (subject), action (verb), patient (direct-object), recipient (indirect-object), attribute (adjective), manner
(Expand NP-SIMP)

(Indefinite-Article Count-Noun)
(Definite-Article Count-Noun)
(Definite-Article Mass-Noun)
(Indefinite-Article Plural-Noun)
(Name)
(Mass-Noun)

Fig. 4. Scope expansion.

Root: "employee"
Type: (Count Person)
 Syntax: (INDEFINITE)
Concepts: (ANTIMATE CONCRETE MOBILE
 HUMAN (LABOR)
Qualities: (AGS INTELLIGENCE SIZE SEX
 WORK-HIST INTR WHERE)

Fig. 5. Simple transitive object: noun "employee".

Root: "give"
Type: DTRANS
 Morphology: ((prod "give") (prep-past "giving")
 Aspect: (HUMAN (INDIVIDUAL PERSONAL)
 Patient: (PRODUCT INPERSONAL)
Recipient: (INDIVIDUAL PERSONAL)
Quantifiers: (WHEN PRIO)

Fig. 6. Simple transitive object: verb "give"
3. System Flow

This section describes the behavior of the controller in generating a sentence.
3.1 Sentence level

After a template has been selected, the appropriate sentence type subclass is instantiated using the template parser, sentence structure constraints, and the template parse from the template object. This sentence object is passed to the function generate-sentence which returns the object with the sentence string filled in. The text template and the sentence string are displayed as the system's output.

The generate sentence function transforms the template parse into a list of phrase objects. These phrase objects are then sorted into a list of goals. The order of the phrases is based on their semantic roles and is as follows: agent, patient, activity, preposition, recipient, attribute, manner, adverbial, head. Each sub-goal phrase is formed by the function generate phrase. After all the phrases have been generated, the function sentence function builds the sentence string. The completed sentence object is then returned.

3.2 Phrase level

The build phrase function takes the phrase knowledge in the parsed template and builds the appropriate phrase object. This knowledge includes the semantic role and an expansion object. For example, a proper semantic object, it also includes the object being modified and the type of modifying phrase. During the construction of this phrase object, the expansion object is transformed into a list of possible expansions.

The generate phrase function selects one of possible expansions randomly and transforms it into a list of word objects. These word objects are ultimately sorted into a list of goals, based on the type of phrase. Each word in each goal is formed through the function generate word. After all the words have been generated, the form phrase function builds the phrase string. The completed phrase object is then returned.

3.3 Word level

The function build words creates a list of word objects by copying each expansion half code object in the
phrase's expansion list to a newly created word object.

The generate word function receives the current phrase object, the current word object, a constraint object, and the phrase local knowledge pad as input. Before any code is executed in the primary method the multiple before methods are executed to build the complete constraint object. Included in this process is the selection of those lexical entries which match the part of speech for the target word. Next, all the constraints are applied in order to select only those words that can pass the filter conditions from the selected lexical list. One of the candidates is then chosen at random; the winning entry is formed and the modified word object is returned.

4. Analysis

The sentences are generated to satisfy the syntactic, semantic, and contextual constraint. The syntactic constraints are stored as objects in the contextual knowledge base as well as in the object lexicon. The contextual constraints are also generated dynamically and are stored as objects on the contextual knowledge pad during the generation of each sentence, phrase, and word (Figure 11). These constraints objects are only kept temporary during each sentence, phrase, or word generation. The contextual constraints objects placed on the sentence level pad remain throughout the generation of the entire sentence. The constraints placed on the phrase level and word level pad remain during the generation of a phrase or a word. The contextual objects ensure the cohesiveness and compatibility of the grammatical phrases as well as the sentence. For instance, if the main verb of the sentence being generated is the past tense of the verb to go which semantically expresses a movement, its subject must be mobile and its receiver must express a location or place. Therefore a semantically correct sentence such as John went into the old store may be generated, and semantically incorrect sentences such as John went into the table or the table went into the jungle will not be generated.

With a relatively small number of words, approximately 100, we were able to demonstrate the power of our system. Expanding the object lexicon, by adding new words, will dramatically enhance the flexibility and strengthen the functionality of the system. Due to our highly modular design which relies on an object-oriented framework, adding additional functionalities to the existing system is extremely simple. The system produces a sentence. The generation time for each sentence is generally in the range of 0.5 to 1.5 seconds.

A number of non-sentential sentences generated by the system and included in Table 1 above.

Figure 12 contains the key for the symbols found in the following example. Each phrase is represented by an oval. Before a phrase is completed, it will contain the phrase type. After the phrase is completed, the oval is filled and the completed phrase is placed directly below it. Likewise, words are represented by rounded rectangles. Before the word is formed, it contains the subphrase to be used in its formation. After the formation, the rectangle is filled and the word is placed directly below it. Finally, a directed line between objects denotes a controlling relationship, while an undirected line denotes a sequential relationship.

A. Example

Let us now look at the generation of the following sentence: Who did the separation here? The template for this sentence is Who [DO-Verb] [NP] [VP] which is parsed by (Patent Auxiliary Agent Action).

**Generate Sentence:** Build phrase objects from the presence in the sentence object. For this sentence four objects are created: personal, noun, verb, and action-verb. Order phrases by priority based on semantic role. The ordered phrase list is: noun, verb, personal, action-verb. Generate each phrase.

Form the sentence by combining the phrases:

**Generate Act-Pred-VP:** Before: No action taken.
Primary: Select one of the possible expansions at random. For this template, the only choice is
the verb root.
Build word constraint list for each expansion element.
Order words by priority based on part of
speech. Generate each word.
After: No action taken.

Generate Verb-Root:
Before: Build word constraint list
Word List = Verbs

Type = Transitive
Tense = Root
Primary: Apply constraints to reduce list of verbs. Select a verb to be chosen.
Stats: root form and in word object.
Aux: Copy predicate, agent concepts, patient concepts and quantified list to act predicate object on phrase level slot.

Generate Agg-Sub-VP:
Before: Build phrase constraint
Copy agent-concepts from the act-"ed"-object.
Primary: Select one of the possible expansions at random. [article] [noun-count] is chosen.
Build word object(s) for each expansion element.
Order words by priority based on part of speech. Generate each word.
After: No action taken.

Generate Noun-Count:
Before: Build word concept object.
   Word List = Nouns
   Type = Count
   Concept-List = Individual Management Personnel
Primary: Apply constraints to reduce list of nouns.
   Select a noun supervisor is chosen.
   Form word and store in word object.
Chose the number and person at random. Singular, third is chosen.

Generate Art-Def:
Before: Build word concept object.
   Word List = Articles
   Type = Definite
Primary: Apply constraints to reduce list of articles.
   Only the word that meets the constraints.
   Store the in word object.
After: No action taken.

Generate Pat-DO-NP:
Before: Build phrase concept.
   Copy patient-concepts from the act-"ed"-object.
After: No action taken.

Generate Aux-VP:

Before: No action taken.
Primary: Select one of the possible expansions at random. For this template (DO-Verb), the only choice is the verb chosen. Build word object(s) for each expansion element.
Order validity, priority based on part of speech. Generate each word.

After: No action taken.

Generate DO-Verb:

Before: Build word constraint object:
WordList = Verbs
Root = do

Primary: Apply constraints to reduce list of verbs. Only the verb do meets the constraints. Form word and store in word object.
Choose tense at random. Past is chosen.
Use standard formation rules or lexical object morphology to build correct form of the word.
The string did is built.

After: No action taken.
Finally, the complete sentence Who did the supervisor hire? is generated.
Fig. 26. The phrase "who" is focused.

Fig. 27. The expansion of Aux-VP.

Fig. 28. The phrase "who" is focused.
5. Conclusions.

We designed and implemented a system in an object-oriented framework for generating non-functional syntactically-correct English sentences. In this system, syntactic and semantic knowledge is utilized concurrently in our design. Each lexical-object has local static knowledge which enables it to make decisions. Hence, each word is an independent knowledge source. In addition, contextual constraints are generated syntactically which will limit the available choices for the subsequent parts of the sentence. This approach, which mirrors the way human form sentences, worked well and showed much future promise. This is an effective strategy for preserving the cohesion and for maintaining the compatibility of parts of each sentence. The prototype’s viewpoint proved extremely beneficial in guiding rules, which the same object can be viewed in a variety of ways. This modular design allows for the addition of lexical entries and templates that can increase the expressive power of the system with little or no changes to the code.

A natural extension of this work is to modify the system to generate sentences which express a desired meaning. To achieve this, an intention-to-constraints mapping interface can be added to the current system (Figure 23).

This interface component would receive as input a collection of objects, each object expressing a segment of the desired intention, and collectively expressing the complete theme or intention. This information is used to select the appropriate sentence structure and to constrain the proper thematic constraints and would then serve as input to the current system to guide the generation process. This extension would require very little change in the current system. The mechanism for allowing sentence-level constraints already exists in the current system. The thematic constraints would simply need to be expressed as sentence-level constraints in order to be passed throughout the system. Some additional work would be needed to ensure that the new top-level constraints are not in conflict with the static constraints of the current system. The current system is implemented in CLIPS (COMMON LISP Object System), an object oriented extension of COMMON LISP (Keene, 1989). This work was partially funded by a grant (Kohlberg, 1993) from the Air Force Office of Scientific Research (AFOSR).

References


