Team UDEL KBGen 2013 Challenge

Keith Butler, Priscilla Moraes, Ian Tabolt, Kathleen F. McCoy
Computer and Information Science Department
University of Delaware
Newark, DE 19716
keithb@udel.edu, pmoraes@udel.edu, itabolt@udel.edu, mccoy@cis.udel.edu

Abstract

This document describes the University of Delaware's entry into KBGen 2013 Challenge which provided teams with input data representation from the AURA knowledge base (KB), developed in the context of the HALO Project at SRI International, along with a lexicon mapping for concepts present on those input files. Training sentences were also provided. The task was to accurately generate an English sentence depicting the information from a set of triples from the knowledge base.

1 Approach

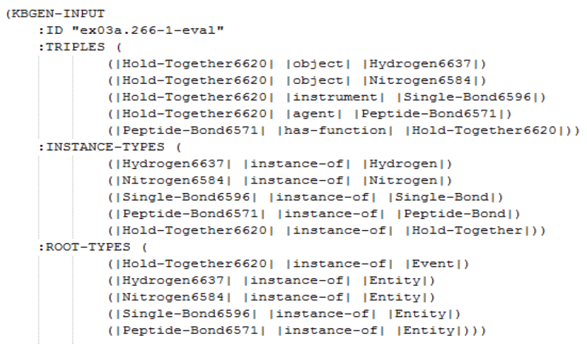
Our approach to the problem was to develop a set of rules for translating KB structures into English structures and to use an existing generator, such as SimpleNLG (Gatt & Reiter, 2009) or FUF-SURGE (Elhadad, 1993) to generate the sentences.

Our analysis of pre-release data provided by the KBGen organization (triple-files, training sentences, tree graphs, lexicon) was facilitated by writing a mashup program (KBGenMashup) that enabled viewing/searching the data. The program initially loads all the training sentences into a clickable list box. When a sentence is clicked, all data relating to that sentence is displayed: corresponding triples, tree-graph, and Stanford parse of the sentence. The displayed triples are given "hot spots" so clicking on them will present a list of other sentences containing (or NOT containing) that same relation or instance type. Finally, KBGenMashup enables a search for other sentences that contain a given word or phrase. Using this tool allowed us to discover common realization patterns for certain KB triples.

2 Major sentence types

Our initial generator implementation utilized SimpleNLG in a java wrapper. Our tack was to focus on the realization of major sentence types, generally identified by the presence of a particular function in the KB triples, e.g. has-function, subevent, plays. These functions provided the main verb and sentence structures, and other KB relations were fit into this structure (in subject/object position or as adjuncts) in a rule-based way.

For instance, Figure 1 shows a triples file from the testing data that was identified under the cluster *has-function*, along with the sentence generated by our system and the rule used to realize the cluster for this relation type.



```
(KBGEN-INPUT
    :ID "ex03a.266-1-eval"
    :TRIPLES (
            (|Hold-Together6620| |object| |Hydrogen6637|)
            (|Hold-Together6620| |object| |Nitrogen6584|)
            (|Hold-Together6620| |instrument| |Single-Bond6596|)
            (|Hold-Together6620| |agent| |Peptide-Bond6571|)
            (|Peptide-Bond6571| |has-function| |Hold-Together6620|))
    :INSTANCE-TYPES (
            (|Hydrogen6637| |instance-of| |Hydrogen|)
            (|Nitrogen6584| |instance-of| |Nitrogen|)
            (|Single-Bond6596| |instance-of| |Single-Bond|)
            (|Peptide-Bond6571| |instance-of| |Peptide-Bond|)
            (|Hold-Together6620| |instance-of| |Hold-Together|))
    :ROOT-TYPES (
            (|Hold-Together6620| |instance-of| |Event|)
            (|Hydrogen6637| |instance-of| |Entity|)
            (|Nitrogen6584| |instance-of| |Entity|)
            (|Single-Bond6596| |instance-of| |Entity|)
            (|Peptide-Bond6571| |instance-of| |Entity|)))
```

Figure 1: A triples file from the testing data.

Sentence generated: *The function of the peptide bond is to hold together hydrogen and nitrogen using a single bond.*

The identified rule for this input is the *has-function* rule. The main entity is the entity that is related to the event of the instance by the *has-function* relation type. The rule states that the subject of the sentence is the "*the function of [main entity]*". For this template the verb *to be* is identified as the main verb and the object of the sentence is a verb phrase (VP) composed of the events present in the triples file in the infinitive form, and the existing secondary entities. Each secondary entity is related to an event by a semantic relation type. The nature of this relation defines which role the secondary entity plays in the sentence (e.g. the "object" relation, when present, usually links the event to the head(s) of the noun phrase (NP) within the VP). Although the majority of the input files have secondary entities that are related to the main event by the object relation, some other cases do not present them. The head of the noun phrases can be represented, in those cases, by secondary relations connected to the main event by one of *agent, base, result, raw-material*, relation types. Heuristics are applied in order to define the head of the noun phrase since the relation that will define which entity is the head of the NP is based on the combination of the existing relations. The relations in the set of triples that are not already realized as one of the previous roles in the sentence are then realized recursively for each event, complementing the VP. Those relations are represented by prepositional phrases (PP) and the preposition chosen for each PP represents the semantic role of the relation type (e.g. *instrument* relations often use the prepositions **with** or **using**, while *donor* and *origin* relations often use **from**).

3 No-Events and other sentence types

The simple strategy described above worked well for some sentence types, but others required more sophisticated triple traversal. In particular, realizing triple sets not containing an event was problematic. With time running short, we implemented a second realizer to handle these types. It used its own heuristics, plus stored the sets of triples in a database that allowed for flexible traversing. Consider its heuristics to handle no-event triple sets (events generally provide the verb and sentence structure). No-event sentences would use a form of "be" as the main verb, but we still needed to identify the sentence's main subject. To do this, the software looks for the Entity that is on the left side of the most triples. Why? There is more information about this Entity than about any other. Consider ex29b.4 (Figure 2). The tree graph shows that "Restriction-Site" is on the left side of five triples. It should be the subject of the sentence, which could be realized as "A restriction site is a short DNA sequence which consists of 2 deoxyribose and a deoxyribonucleoside monophosphate." Note the order of the Entities in the sentence. The subject is mentioned first, then its adjective ("short"), then class ("DNA sequence"), then remaining entities. In realizing the remaining Entities, a common routine is used to check for cardinality and perform any rewording as appropriate.

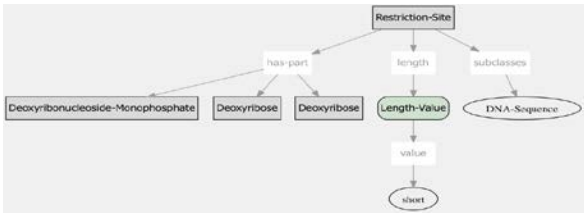


Figure 2: ex29b.4

In many cases, there was a tie among the times Entities were on the left. In one type of "tie" (ex05a2.265, Figure 3), there is a cycle in the graph (see "Fibronectin, "Carbohydrate-Side-Chain", "Surface".) In these cases, the heuristic chooses the "middle" Entity in the cycle (Carbohydrate in this case) as the subject. Then in choosing mention-order, the software (usually) starts the sentence by putting the adjective before the subject (i.e. "branched" & "carbohydrate side chain"), then visits each Entity around the cycle, then traverses up to the "Top" Entity. This sentence is realized as "There are branched carbohydrate side chains at the surface of the fibronectin of an animal plasma membrane."

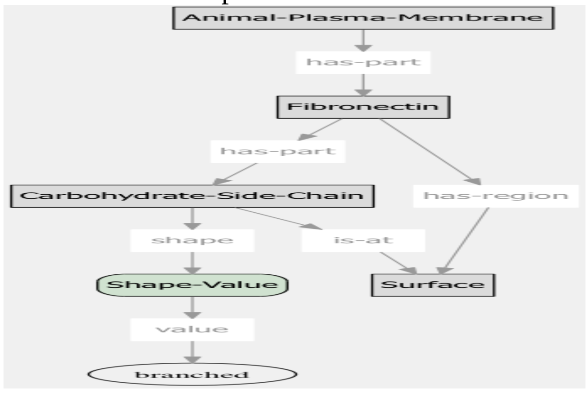


Figure 3: ex05a2.265

4 Conclusions

We have described a template-based generation entry based on two different paradigms. In one, sentences are formed on the basis of a major relation that generally selects the main verb and fits the realization of the other pieces according to the structures specific for that sentence type. The second piece that we needed is based on flexibly traversing the knowledge base and realizing based on patterns found in the triples.

References

Albert Gatt and Ehud Reiter. 2009. *SimpleNLG: A realization engine for practical applications*, Proceedings of the 12th European Workshop on Natural Language Generation, pages 90–93, Athens, Greece, 30 – 31 March 2009.

Michael Elhadad. 1993. FUF: the Universal Unifier User Manual Version 5.2.