Natural Language Generation for a Smart Biology Textbook

Eva Banik[1], Eric Kow[1], Vinay Chaudhri[2], Nikhil Dinesh[2], and Umangi Oza[3]

[1]{ebanik,kowey}@comp-ling.co.uk, Computational Linguistics Ltd, London, UK
[2] {chaudhri,dinesh}@ai.sri.com, SRI International, Menlo Park, CA
[3]umangi.oza@evalueserve.com, Evaluserve, New Delhi, India

1 Application Context

In this demo paper we describe the natural language generation component of an electronic textbook application, called Inquire[1]. Inquire interacts with a knowledge base which encodes information from a biology textbook. The application includes a question-understanding module which allows students to ask questions about the contents of the book, and a question-answering module which retrieves the corresponding answer from the knowledge base. The task of the natural language generation module is to present specific parts of the answer in English. Our current generation pipeline handles inputs that describe the biological functions of entities, the steps of biological processes, and the spatial relations between parts of entities. Our ultimate goal is to generate paragraph-length texts from arbitrary paths in the knowledge base. We describe here the natural language generation pipeline and demonstrate the inputs and generated texts. In the demo presentation we will show the textbook application and the knowledge base authoring environment, and provide an opportunity to interact with the system.

2 The Knowledge Base

The knowledge base contains information from a college-level biology textbook[2], encoded by biologists as part of project HALO at SRI[3]. The core of the knowledge base is the CLIB ontology[4], which is extended with biology-specific information. The knowledge base encodes entity-to-event relations (similar to thematic roles in linguistics), event-to-event relations (discourse relations), various property values and relations between properties, spatial relations, cardinality constraints, and roles that participants play in events. The input to the generation pipeline is a set of triples extracted from the biology knowledge base. Currently our content selection includes either an event and the entities that participate in the event, or a set of entities and spatial relations between them.

3 Generation Grammar and Lexicon

Our generation grammar consists of a set of Tree Adjoining Grammar (TAG) elementary trees. Each tree is associated with either a single relation, or a set of relations in the knowledge base. As an example, Fig 1 illustrates the mapping between elementary trees and event participant relations in the KB for the above input. We currently associate up to three different elementary trees with each event and the connected set of participant relations: an active sentential tree, a passive sentential tree and a complex noun phrase.

The knowledge base provides concept-to-word

[1]The work described in this paper and presented in the demo is funded by Vulcan Inc.

[2] **Reece et al. 2010.** Campbell biology. Pearson Publishing.

[3] **Gunning Et al, 2010.** Project halo update progress toward digital aristotle. AI Magazine Fall:33-58. See also http://www.projecthalo.com/

[4]http://www.cs.utexas.edu/users/mfkb/RKF/clib.html

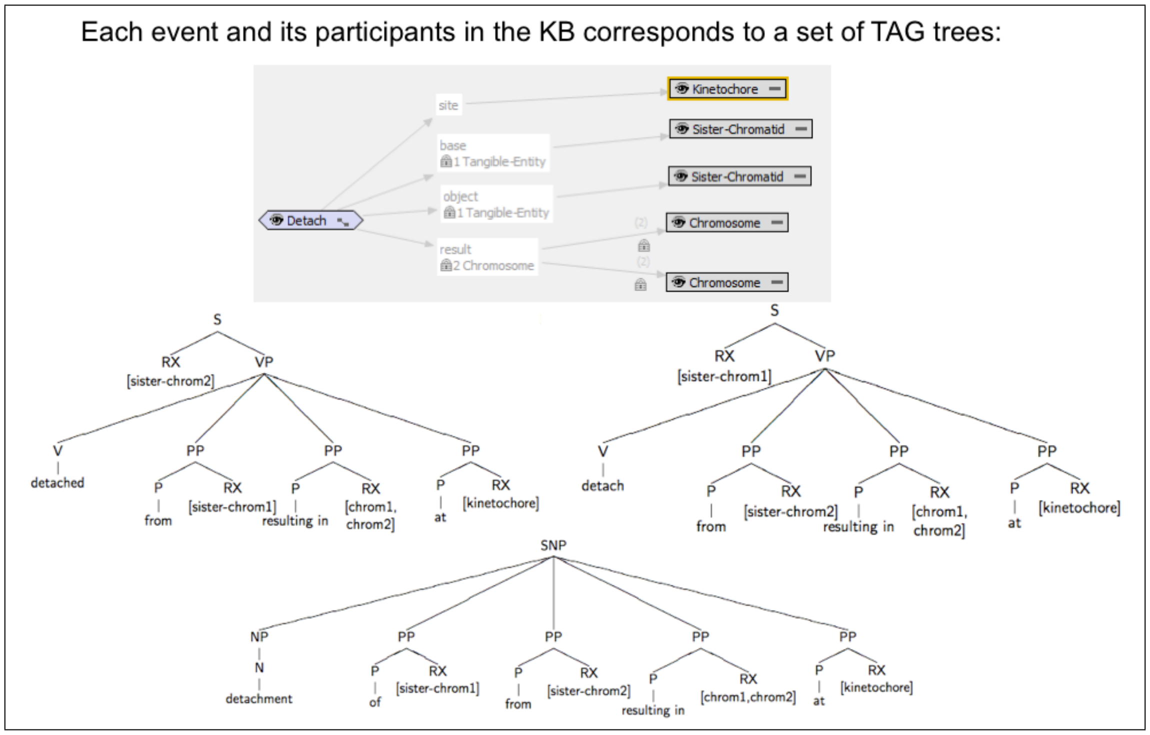

Figure 1: The grammar of the surface realizer

mappings (a list of synonyms) for every concept, and the words are used in the generation lexicon to anchor elementary TAG trees. Our generation grammar consists of a set of TAG tree templates, which are defined as combinations of tree fragments and are compiled using the XMG metgrammar toolkit[5].

These underspecified elementary trees are further specified in the generation lexicon, which maps concepts onto elementary tree templates, and associates a word (an anchor) with the tree, along with other idiosynchratic information (e.g., preposition choice). We create a generation lexicon dynamically at run-time, by mapping tree templates onto concepts based on the number and types of participants, and the lexical information associated with the event (e.g., the preposition requirements of the verb).

Concept names for entities are included in the elementary trees as features on the corresponding NP nodes. These features form part of the input to the referring expression generation module, which looks up the concept name in the concept-to-word mapping to obtain a list of possible noun phrases.

4 Realization

Our natural language generation pipeline is centered around the GenI surface realizer[6,7]. The set of triples yielded by content selection are first aggregated and converted to GenI's input format, a set of flat semantic literals. We then feed this input to GenI to produce an underspecified surface form in which referring expressions are still underspecified:

NP is detach from NP resulting in NP at NP

NP detach from NP resulting in NP at NP

Detachment of NP from NP resulting in NP at NP

A post-processing module carries out referring expression generation and morphological realization to produce the fully specified output.

[5]https://sourcesup.renater.fr/xmg/

[6] **Kow, Eric. 2007.** Surface realisation: ambiguity and determinism. Doctoral Dissertation, Universite de Henri Poincare - Nancy 1.

[7] **Banik, Eva 2010.** A minimalist approach to generating coherent texts. Phd thesis, Department of Computing, The Open University

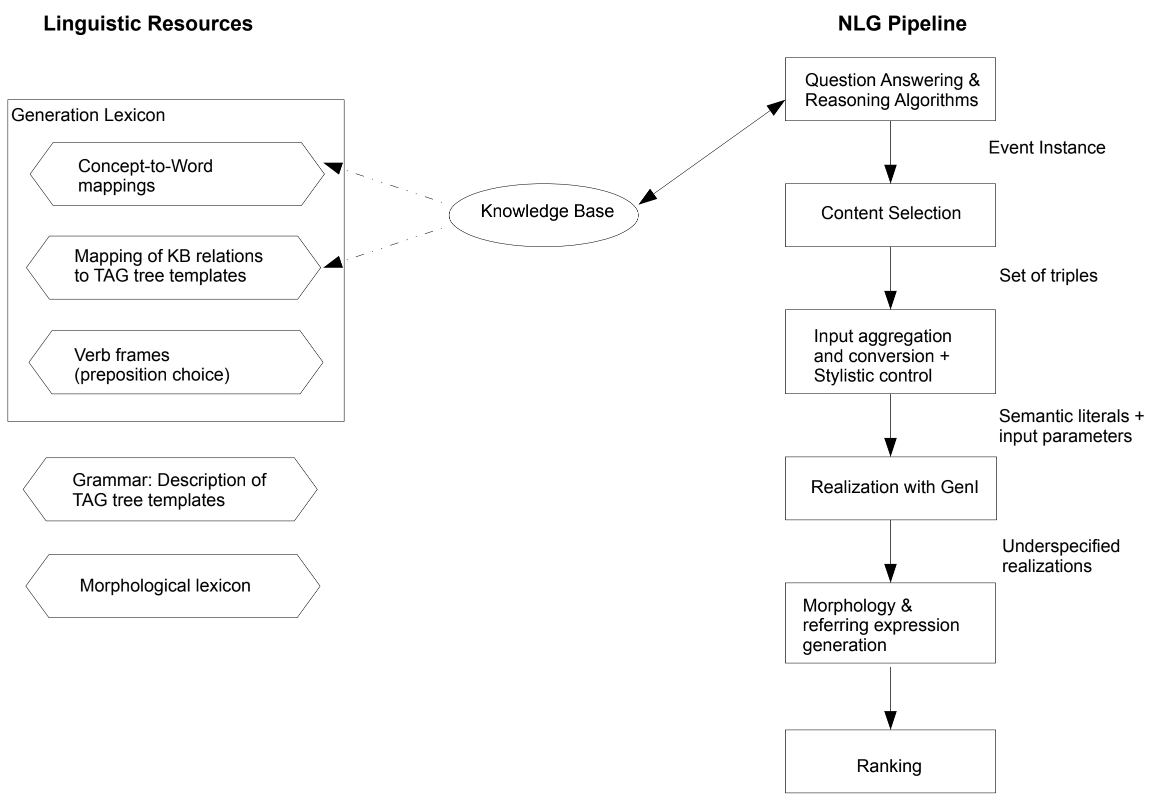
Linguistic Resources
NLG Pipeline
Generation Lexicon
Concept-to-Word mappings
Mapping of KB relations to TAG tree templates
Verb frames (preposition choice)
Grammar: Description of TAG tree templates
Morphological lexicon
Knowledge Base
Question Answering & Reasoning Algorithms
Event Instance
Content Selection
Set of triples
Input aggregation and conversion + Stylistic control
Semantic literals + input parameters
Realization with GenI
Underspecified realizations
Morphology & referring expression generation
Ranking

Figure 2: Linguistic resources and the generation pipeline

Our referring expression realization algorithm performs further semantic aggregation where necessary to produce cardinals ("two chromosomes"), and decides on a suitable determiner based on previous mentions of instance names and subclasses in the discourse context (definite/indefinite determiner, "another" or "the same"). For the input shown in Fig 1, our system will produce the following three realizations:

1. A sister chromatid detaches from another sister chromatid resulting in two chromosomes at a kinetochore.
2. A sister chromatid is detached from another sister chromatid resulting in two chromosomes at a kinetochore.
3. Detachment of a sister chromatid from another sister chromatid resulting in two chromosomes at a kinetochore

We rank the generated outputs based on their linguistic properties using optimality theoretic constraints (e.g., active sentences are ranked above passive sentences), where each constraint corresponds to a (set of) tree fragments that contributed to building the tree that appears in the output. Our system also allows for extra input parameters to be sent to GenI to restrict the set of generated outputs to fit a specific context (e.g., syntactic type or focused discourse entity). Our full natural language generation pipeline is illustrated in Fig 2.

5 Future Work

We are currently working on extending the system to handle more relations and other data types in the knowledge base. This involves extending the grammar to new sentence types and other linguistic constructions, and extending the content selection module to return more triples from the knowledge base. Our ultimate goal is to be able to generate arbitrary – but in some sense well-formed – paths from the knowledge base as coherent paragraphs of text.