Dialogue and Discourse 3(2) (2012) 11–42 doi: 0.5087/dad.2012.202

Semantics-based Question Generation and Implementation

Xuchen Yao XUCHEN@CS.JHU.EDU
Department of Computer Science, Johns Hopkins University, 3400 N. Charles Street, Baltimore, USA

Gosse Bouma G.BOUMA@RUG.NL
Information Science, University of Groningen, PO Box 716, 9700 AS Groningen, The Netherlands

Yi Zhang YZHANG@COLI.UNI-SB.DE
LT-Lab, German Research Center for Artificial Intelligence (DFKI GmbH)
Department of Computational Linguistics, Saarland University, 66123 Saarbrücken, Germany

Editors: Paul Piwek and Kristy Elizabeth Boyer

Abstract

This paper presents a question generation system based on the approach of semantic rewriting. State-of-the-art deep linguistic parsing and generation tools are employed to map natural language sentences into their meaning representations in the form of Minimal Recursion Semantics (MRS) and vice versa. By carefully operating on the semantic structures, we obtain a principled way of generating questions which avoids ad-hoc manipulation of syntactic structures. Based on the (partial) understanding of the sentence meaning, the system generates questions that are semantically grounded and purposeful. As the generator uses a deep linguistic grammar, the grammaticality of the generation results is licensed by the grammar. With a specialized ranking model, the linguistic realizations from the general purpose generation model are further refined for the question generation task. The evaluation results from QGSTEC2010 show promising results for the proposed approach.

1. Introduction

Question Generation (QG) is the task of generating reasonable questions from an input, which can be structured (e.g. a database) or unstructured (e.g. a text). In this paper, we narrow the task of QG down to taking a natural language text as input (thus textual QG), as it is a more interesting challenge that involves a joint effort between Natural Language Understanding (NLU) and Natural Language Generation (NLG). Simply put, if natural language understanding maps text to symbols and natural language generation maps symbols to text, then question generation maps text to text, through an inner mapping from symbols for declarative sentences to symbols for interrogative sentences, as shown in Figure 1. Here we use symbols as an organized data form that can represent the semantics of natural languages and that can be processed by a machinery, artificial or otherwise.

The task of question generation contains multiple subareas. Usually, the approach taken for QG depends on the purpose of the QG application. Generally speaking, a QG system can be helpful in the following areas:

- Intelligent tutoring systems. QG can ask questions based on learning materials in order to check learners' accomplishment or help them focus on the keystones in study. QG can also help tutors to prepare questions intended for learners or prepare for potential questions from learners.

Submitted 1/11; Accepted 1/12; Published online 3/12

- Closed-domain Question Answering (QA) systems. Some closed-domain QA systems use predefined (sometimes hand-written) question-answer pairs to provide QA services. By employing a QG approach such systems could be ported to other domains with little or no effort.
- Natural language summarization/generation systems. QG can help to generate, for instance, Frequently Asked Questions from the provided information source in order to provide a list of FAQ candidates.

In terms of target complexity, QG can be divided into *deep* QG and *shallow* QG (Graesser et al., 2009). Deep QG generates deep questions that involve more logical thinking (such as why, why not, what-if, what-if-not and how questions) whereas shallow QG generates shallow questions that focus more on facts (such as who, what, when, where, which, how many/much and yes/no questions). Given the current state of QG, most of the applications listed above have limited themselves to shallow QG.

We describe a semantics-based system, MrsQG[1], that generates questions from a given text, specifically, a text that contains only a single declarative sentence. This restriction was motivated by Task B of the Question Generation Shared Task and Evaluation Challenge (QGSTEC2010; Rus et al., 2010; Rus et al., this volume), which provides participants with a single sentence and asks them to generate questions according to a required target question type. By concentrating on single sentences, systems can focus on generating well-formed and purposeful questions, without having to deal (initially) with text analysis at the discourse level.

The basic intuition of MrsQG can be explained by the following example. Think of generating a few simple questions from the sentence "`John plays football.`":

Example 1 *`John plays football.`*
(a) `Who plays football?`
(b) `What does John play?`

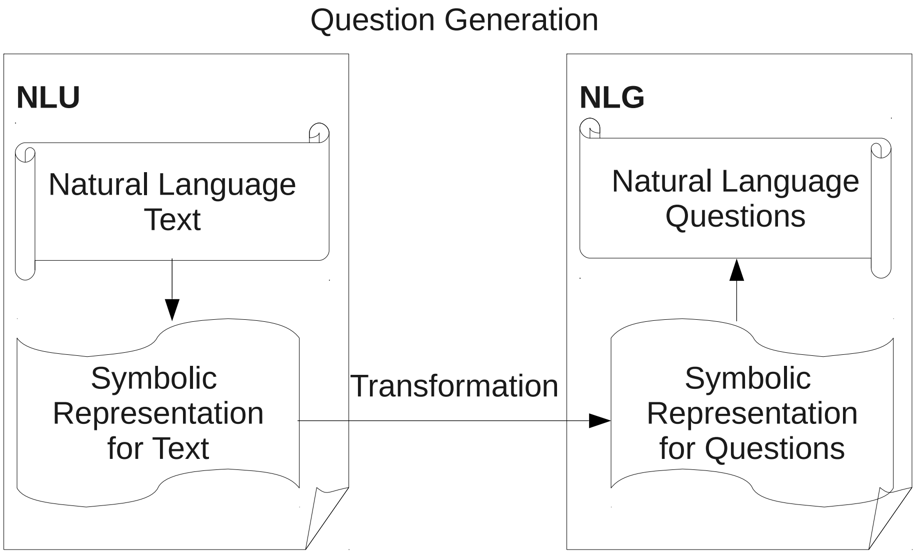

Question Generation
NLU
Natural Language Text
Symbolic Representation for Text
Transformation
NLG
Natural Language Questions
Symbolic Representation for Questions

Figure 1: The relation between Question Generation and its two components: Natural Language Understanding (NLU) and Natural Language Generation (NLG).

[1] `http://code.google.com/p/mrsqg/`

When people perform question generation, a transformation from declarative sentences to interrogatives happens. This transformation can be described at different levels of abstraction. An intuitive one is provided by predicate logic:

Example 2 *play(John, football)* ⇐ `John plays football.`
(a) play(who, football) ⇐ `Who plays football?`
(b) play(John, what) ⇐ `What does John play?`

If the above abstraction can be described and obtained in a formal language and transformation can be done according to some well-formed mechanism, then the task of question generation has a solution.

We propose a semantics-based method of transforming the Minimal Recursion Semantics (MRS, Copestake et al., 2005) representation of declarative sentences to that of interrogative sentences. The MRS analysis is obtained from PET (Callmeier, 2000) running with the English Resource Grammar (ERG, Flickinger, 2000) as the core linguistic component, while the generation function is delivered by the Linguistic Knowledge Builder (LKB, Copestake, 2002).

The advantage of this approach is that the mapping from declarative to interrogative sentence is done on the semantic representations. In this way, we are able to use an independently developed parser and generator for the analysis and generation stage. The generator will usually propose several different surface realizations of a given input due to its extensive grammatical coverage. This means that the system is able to produce more diverse questions, but also that ranking of various surface forms becomes an issue. Additional advantages of the semantic approach are that it is to a large extent language independent, and that it provides a principled level of representation for incorporating lexical semantic resources. For instance, given that "sport" is a hypernym of "football", we can have the following transformation:

Example 3 *play(John, football) & hypernym(sport, football)* ⇒ *play(John, which sport)*

The hypernym relation between "sport" and "football" can either be obtained from ontologies, such as a list of different sports, or semantic networks, such as WordNet (Fellbaum, 1998).

This paper is organized as follows. Section 2 reviews related work in question generation and explains why a semantics-based approach is proposed. It also identifies major challenges in the task of question generation. After a brief background introduction in Section 3, Section 4 provides corresponding solutions and describes our implemented system MrsQG. Specifically, Section 4.1 and 4.2 present the key idea in this paper. Evaluation results are presented in Section 5 and several aspects of the proposed method are discussed in Section 6. Finally, Section 7 concludes and addresses future work.

2. Related Work

Generally speaking, the following issues have been addressed in question generation:

1. Question transformation. As shown in Figure 1, this requires a theoretically-sound and practically-feasible algorithm to build a mapping from symbolic representation of declarative sentences to interrogative sentences.

2. Sentence simplification. Complex and long sentences widely occur in written languages. But questions are rarely very long. On the one hand, complex input sentences are hard to match against pre-defined patterns. On the other hand, most current question generation approaches transform the input sentence into questions, thus it is better to keep the input short and succinct in order to avoid lengthy and awkward questions. Thus sentence simplification is usually performed as a pre-processing step.

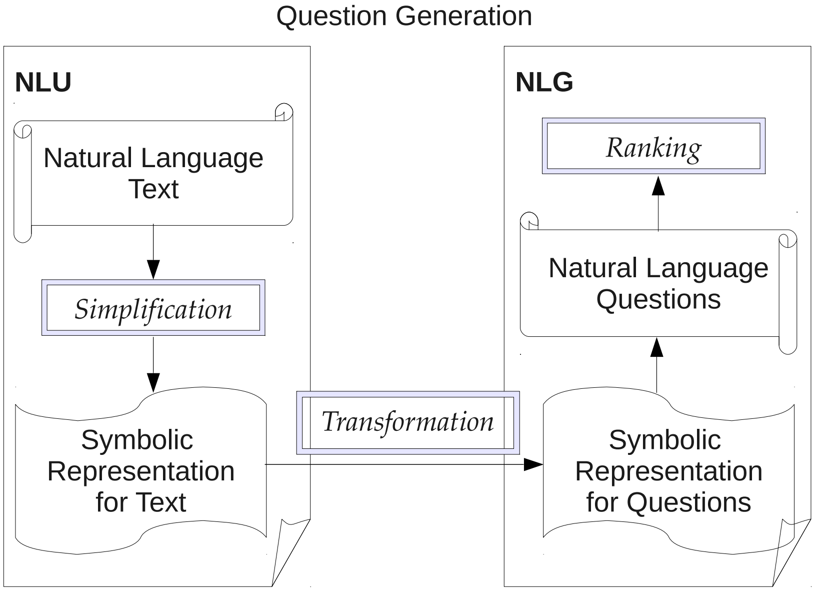

Question Generation
NLU
Natural Language Text
Simplification
Symbolic Representation for Text
Transformation
NLG
Ranking
Natural Language Questions
Symbolic Representation for Questions

Figure 2: Three major problems (*sentence simplification, question transformation* and *question ranking*) in the process of question generation, as shown in the framed boxes.

3. Question ranking. In the case of over generation, a ranking algorithm to grade the grammaticality and naturalness of questions must be developed. Furthermore, a good ranking algorithm should also select relevant and appropriate questions according to the content requirements.

Figure 2 shows these three problems in an overview of a complete question generation framework. The following section reviews research on these three issues in more detail.

2.1 Question Transformation

There are generally three approaches to question transformation and generation: template-based, syntax-based and semantics-based.

Mostow and Chen (2009) reported on a template-based system under a self-questioning strategy to help children generate questions from narrative fiction. Three question templates are used to produce what/why/how questions. Of 769 questions evaluated, 71.3% were rated acceptable. This work was further expanded by Chen et al. (2009) with 4 more templates to generate What-would-happen-if, When-would-x-happen, What-would-happen-when and Why-x questions from informational text questions. Template-based approaches are mostly suitable for applications with a special purpose, which sometimes come with a closed-domain. The trade-off between coverage and cost is hard to balance because human labor is required to produce high-quality templates. Thus, it is generally considered unsuitable for open-domain general purpose applications.

Syntax-based approaches include Wyse and Piwek (2009) and Heilman and Smith (2009), both of which use a very similar method for manipulating syntactic trees. The core idea is to transform a syntactic tree of a declarative sentence into that of an interrogative. Specific matching and transformation rules are defined by experienced linguists and operate on tree structures. All operations are straightforward from a syntactic point of view. Heilman and Smith (2009) reported 43.3% acceptability for the top 10 ranked questions and produced an average of 6.8 acceptable questions per 250 words on Wikipedia texts.

Semantics-based approaches are less explored in previous research. Schwartz et al. (2004) introduce a content question generator that uses the logical form to represent the semantic relationships of the arguments within a sentence and generate WH-questions. However, the paper only introduces the result of this generator whereas the inner mechanism is not presented. Sag and Flickinger (2008) discuss the possibility and feasibility of using the English Resource Grammar for generation under a Head-driven Phrase Structure Grammar (HPSG, Pollard and Sag, 1994) framework. Minimal Recursion Semantics is the input to ERG and linguistic realizations come from the Linguistic Knowledge Builder system. Successful applications are listed in support of their arguments for generating through ERG and LKB. In this paper, we present the implementation of a question generation system following their proposed approach.

2.2 Sentence Simplification

Sentence simplification reduces the average length and syntactic complexity of a sentence, which is usually marked by a reduction in reading time and an increase in comprehension.

Chandrasekar et al. (1996) reported two rule-based methods to perform text simplification. They take simplification as a two stage process. The first stage gives a structural representation of a sentence and the second stage transforms this representation into a simpler one, using handcrafted rules. The two methods differ in that the structural representation is different and thus rules also change accordingly (one uses chunk parsing and the other supertagging (Bangalore and Joshi, 1999)). In natural language processing, the methods are also used as a pre-processing technique to alleviate the overload of the parser, the information retrieval engine, etc. Thus the analysis of sentences is no deeper than a syntactic tree. In the context of question generation, the analyzing power is not confined to this level. For instance, Dorr et al. (2003) and Heilman and Smith (2010b) use a syntactic parser to obtain a tree analysis of a whole sentence and define heuristics over tree structures. Cohn and Lapata (2009) compress sentences by rewriting tree structures over synchronous tree substitution grammar (STSG, Eisner, 2003).

2.3 Question Ranking

Question ranking falls into the topic of realization ranking, which can be taken as the final step of natural language generation. In the context of generating questions from semantics, there can be multiple projections from a single semantic representation to surface realizations. Velldal and Oepen (2006) compared different statistical models to discriminate between competing surface realizations. The performance of a language model, a Maximum Entropy (MaxEnt) model and a Support Vector Machine (SVM) ranker is investigated. The language model is a trigram model trained on the British National Corpus (BNC) with 100 million words. Sentences with higher probability are ranked better. The MaxEnt model and SVM ranker use features defined over derivation trees as well as lexical trigram models. Their result shows that MaxEnt is slightly better than SVM, while both models significantly outperform the language model. This result is mostly based on declarative sentences. Our proposed question ranking module combines the MaxEnt model from Velldal and Oepen (2006) with a language model trained on questions. Details are in Section 4.3.

Heilman and Smith (2010a) worked directly on ranking questions. They employed an *overgenerate-and-rank* approach. The overgenerated questions were ranked by a logistic regression model trained on a human-annotated corpus. The features used by the ranker covered various aspects of the questions, including length, N-gram language model, WH-words, grammatical features, etc. While 27.3% of all test set questions were acceptable, 52.3% of the top 20% ranked questions were acceptable.

2.4 Perception of Related Work

Among the three subtasks of question generation, most of the work related to sentence simplification and question transformation is heavily syntax-based. The internal symbolic representation of languages is encoded with syntax and transformation rules are defined over syntax. Depending on the depth of processing, these syntactic structures can be either flat (with only POS or chunk information) or structured (with parse trees). However, these approaches also introduce syntax-specific limitations, including language dependencies, ad-hoc transformations and complex syntactic formalisms. Semantics-based approaches are to some extent amenable to these problems. They have more expressive power by utilizing lexical semantics resources and employing more flexible generators. Specifically, semantics-based methods involve both *parsing to semantics* and *generating from semantics*, as well as *transforming via semantics*. Question generation happens to be one application that requires all of these operations.

3. Background: Theory, Grammar and Tools

The proposed system consists of multiple syntactic and semantic processing components based on the DELPH-IN tool-chain (e.g. ERG/LKB/PET), while the theoretical support comes from Minimal Recursion Semantics. This section introduces all the components involved, but concentrates on the parts that are actually used in later sections.

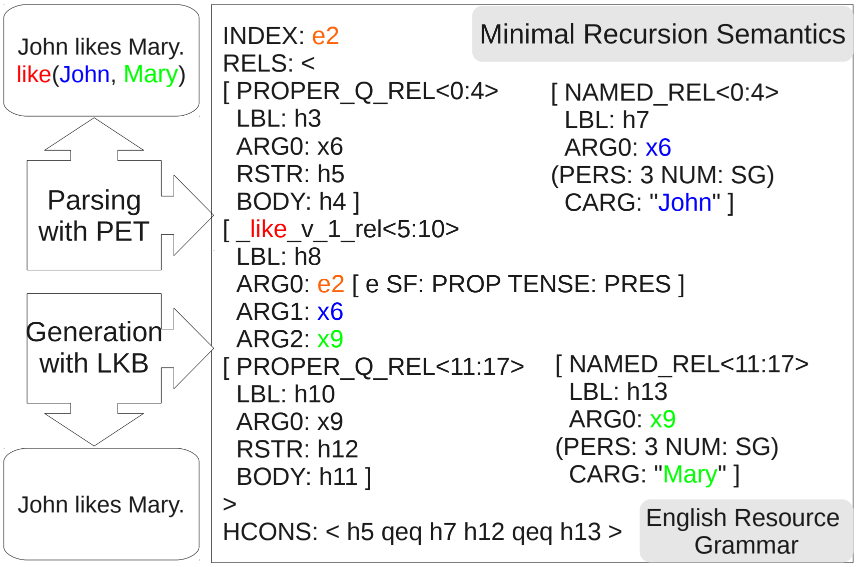
Figure 3 gives an overview of the functionalities of different components. Minimal Recursion Semantics is a theory of semantic representation of natural language sentences with a focus on the underspecification of scope ambiguities. PET is used as a parser to interpret a natural language sentence into MRS with the guidance of a hand-written grammar. In turn, the generation component of LKB takes in an MRS structure and produces various realizations as natural language sentences. Both directions of processing are guided by the English Resource Grammar, which includes a large scale hand-crafted lexicon and sophisticated grammar rules that cover most essential syntactic constructions of English, and connects natural language sentences with their meaning representations in MRS. The tools are developed in the context of the DELPH-IN[2] collaboration, and are freely available as an open source repository.

3.1 Minimal Recursion Semantics and English Resource Grammar

Minimal Recursion Semantics is a meta-level language for describing semantic structures in some underlying object language (Copestake et al., 2005). While the representation can be combined freely with various linguistic frameworks, MRS is particularly convenient to be encoded in typed feature structures. As it turns out to be also practically suitable for encoding semantic compositions in grammar engineering, it is no surprise to see MRS being adopted as the semantic representation for most of the DELPH-IN HPSG grammars. An MRS structure is composed of the following components: LTOP, INDEX, RELS, HCONS, as shown in Figure 4.

LTOP is the topmost label of this MRS. The INDEX usually contains an event variable "*e*", which is co-indexed with the ARG0 property of the main predicate (_like_v_rel in this case), i.e. its *bound variable*. RELS is a bag of Elementary Predications, or EPs, in which a single EP means a single relation with its arguments, such as _like_v_rel*(e2, x5, x9)*. Any EP with RSTR and BODY features corresponds to a generalized quantifier. It takes a form of rel*(*ARG0*,* RSTR*,* BODY*)* where ARG0 refers to the bound variable and RSTR puts a scopal restriction on some other relation by the "qeq" relation specified in HCONS (handle constraints). Here in Figure 4 the relation proper_q_rel*(x5, h4,*

[2] Deep Linguistic Processing with HPSG: http://www.delph-in.net/



```
INDEX: e2
RELS: <
[ PROPER_Q_REL<0:4>          [ NAMED_REL<0:4>
  LBL: h3                      LBL: h7
  ARG0: x6                     ARG0: x6
  RSTR: h5                   (PERS: 3 NUM: SG)
  BODY: h4 ]                   CARG: "John" ]
[ _like_v_1_rel<5:10>
  LBL: h8
  ARG0: e2 [ e SF: PROP TENSE: PRES ]
  ARG1: x6
  ARG2: x9
[ PROPER_Q_REL<11:17>         [ NAMED_REL<11:17>
  LBL: h10                      LBL: h13
  ARG0: x9                      ARG0: x9
  RSTR: h12                   (PERS: 3 NUM: SG)
  BODY: h11 ]                   CARG: "Mary" ]
>
HCONS: < h5 qeq h7 h12 qeq h13 >
```

Figure 3: Different components (PET/LKB/MRS/ERG) of the HPSG-centralized DELPH-IN community. The predicate logic form like(John, Mary) can be further represented by Minimal Recursion Semantics, a more powerful and complex semantic formalism, which is adopted in the English Resource Grammar.

$$\begin{bmatrix} \textbf{mrs} \\ \text{LTOP } \boxed{h1} \\ \text{INDEX } \boxed{e2} \\ \text{RELS } \left\langle \begin{bmatrix} \textbf{proper\_q} \\ \text{LBL } \boxed{h3} \\ \text{ARG0 } \boxed{x5} \\ \text{RSTR } \boxed{h4} \\ \text{BODY } \boxed{h6} \end{bmatrix}, \begin{bmatrix} \textbf{named} \\ \text{LBL } \boxed{h7} \\ \text{ARG0 } \boxed{x5} \\ \text{CARG "Mary"} \end{bmatrix}, \begin{bmatrix} \textbf{\_like\_v} \\ \text{LBL } \boxed{h8} \\ \text{ARG0 } \boxed{e2} \\ \text{ARG1 } \boxed{x5} \\ \text{ARG2 } \boxed{x9} \end{bmatrix}, \begin{bmatrix} \textbf{udef\_q} \\ \text{LBL } \boxed{h10} \\ \text{ARG0 } \boxed{x9} \\ \text{RSTR } \boxed{h11} \\ \text{BODY } \boxed{h12} \end{bmatrix}, \begin{bmatrix} \textbf{\_red\_a} \\ \text{LBL } \boxed{h13} \\ \text{ARG0 } \boxed{e14} \\ \text{ARG1 } \boxed{x9} \end{bmatrix}, \begin{bmatrix} \textbf{\_rose\_n} \\ \text{LBL } \boxed{h13} \\ \text{ARG0 } \boxed{x9} \end{bmatrix} \right\rangle \\ \text{HCONS } \left\langle \begin{bmatrix} \textbf{qeq} \\ \text{HARG } \boxed{h4} \\ \text{LARG } \boxed{h7} \end{bmatrix}, \begin{bmatrix} \textbf{qeq} \\ \text{HARG } \boxed{h11} \\ \text{LARG } \boxed{h13} \end{bmatrix} \right\rangle \end{bmatrix}$$

Figure 4: MRS representation of "`Mary likes red roses`" with some linguistic attributes omitted. *Types* are in **bold**. *Features* (or *attributes*) are in CAPITAL. *Values* are in □. Values in ⟨⟩ are *list values*. Note that some features are token-identical. The suffix **_rel** in relation names is dropped to save space.

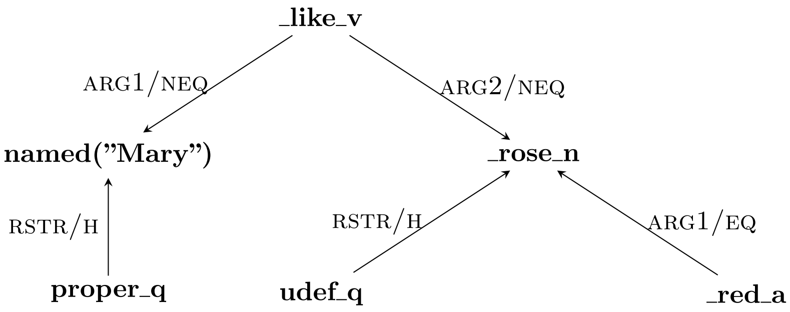


Figure 5: The Dependency MRS structure of "`Mary likes red roses`". Relations between EPs are indicated by the labels of arrowed arcs. Refer back to Figure 4 for the original MRS.

h6) with a "*h4* qeq *h7*" constraint means that proper_q_rel *outscope*s the named_rel, which has a label *h7*.

MRS to some extent preserves the ambiguities that are inherent to natural language, and represents them in a compact way using underspecification. For instance, for the sentence "`every man loves a woman`", there are at least two major readings: *every man loves the same woman* or *every man loves a different woman.* This sentence has only one MRS structure but two scopal readings: it can either be that the EP for "a" outscopes the EP for "`every`" (*every man loves the same woman*) or the other way around. Choosing MRS as our semantic formalism has gained a bonus for the present application: semantic transformation operates on MRSs that are not resolved to unambiguous logical representations, and generation also operates on such structures. Thus, the application can be realized without being forced to do complete ambiguity resolution for the input sentence.

Dependency MRS (DMRS, Copestake, 2008) serves as an interface between *flat* and *non-flat* structures. Recall that in Figure 4, the value of RELS is a bag of EPs. This flat structure is verbose and does not easily show how the EPs in an mrs are connected with each other. DMRS is designed to be more direct in representing the direct relations between EPs, but still preserves all of the original semantic information.

A DMRS is a connected acyclic graph. Figure 5 shows the DMRS of "`Mary likes red roses.`", originally from Figure 4. The directional arcs represent regular semantic dependencies (i.e. the semantic head points to its children) with the labels of arcs describing the relations in detail. A label (e.g. ARG1/NEQ) has two parts: the part before the slash is inherited from the feature name of the original MRS; the part after the slash indicates the type of a scopal relation. Possible values are: H (qeq relationship), EQ (label equality), NEQ (label non-equality), HEQ (one EP's argument is the other EP's label) and NULL (underspecified label relationships).

The semantic composition rules are encoded in the English Resource Grammar, together with the thorough modeling of the syntax. The grammar is based on the HPSG framework, and through over 15 years of continuous development, has achieved broad coverage while maintaining high linguistic accuracy. It consists of a large set of lexical entries under a detailed hierarchy of lexical types, with a modest set of lexical rules for production. The ERG uses a *Davidsonian* representation in which all verbs introduce *event*s. This explains why the _like_v_rel relation in Figures 3 and 4 has *e2* as its ARG0.

3.2 Linguistic Knowledge Builder

We use the generation component of LKB for sentence realization. The Linguistic Knowledge Builder is a grammar engineering platform for developing linguistic grammars in typed-feature structures and unification-based formalisms. It can examine the competence and performance of a grammar by means of parsing and generation. The generation component takes as input a valid MRS structure, and tries to find all possible realizations of the same meaning representation in natural language sentences according to the grammar. The generator uses a chart-based algorithm as described in Kay (1996), Carroll et al. (1999), Carroll and Oepen (2005). The latter two also tackle efficiency problems. In case multiple realizations are available, the generator ranks them with a statistical disambiguation model trained with large scale treebanks.

3.3 Parsing with PET

Although ERG has a large lexicon, there are always unknown words in real text. Thus, a robust processing strategy is needed as fallback. PET is a platform for doing experiments with efficient processing of unification-based grammars (Callmeier, 2000). It employs a two-stage parsing model. Firstly, HPSG rules are used in parsing. Since these rules are sometimes too restrictive and only produce a partial chart, a second stage with a permissive PCFG backbone predicts a full-spanning pseudo-derivation, and the fragmented MRS structures are extracted from these partial analyses. (Zhang et al., 2007). The fragmented MRS does not have full feature structures for all constituents because of rule conflict, otherwise it could have been parsed in the first stage.

4. Proposed Method

Recall in Section 2 we addressed three major problems in question generation: question transformation, sentence simplification and question ranking. A semantics-based system, MrsQG, is developed to tackle these problems by corresponding solutions: MRS transformation for simple sentences, MRS decomposition for complex sentences and automatic generation with rankings. Also, practical issues such as robust generation with fallbacks are addressed.

Figure 6 shows the processing pipelines of MrsQG. The following is a brief description of each step.

1. Term extraction. The Stanford Named Entity Recognizer (Finkel et al., 2005), a RegEx NE tagger, an Ontology NE tagger and WordNet (Fellbaum, 1998) are used to extract terms.

2. FSC construction. The Feature Structure Chart (FSC) format[3] is an XML-based format that introduces tokenization and external annotation to the ERG grammar and PET parser. Using FSC makes the terms annotated by named entity recognizers known to the parser.

3. Parsing with PET. The chart mapping (Adolphs et al., 2008) functionality of PET accepts FSC input and outputs MRS structures.

4. MRS decomposition. Complex sentences first need to be broken into shorter ones with valid and meaningful semantic representation. Also, it must be possible to generate a natural language sentence from this shorter semantic representation. This is the key point in our semantics-based approach. Details in Section 4.2.

5. MRS transformation. Given a valid MRS structure of a sentence, we replace EPs for terms with EPs for (WH) question words. Section 4.1 gives a detailed description with examples.

6. Generating with LKB. Given an MRS for a question, the generator of LKB produces possible surface realizations.

7. Output selection. Given a well-formed MRS structure, LKB might give multiple outputs. Depending on how ERG does generation, some output strings might not sound fluent or even

[3] http://wiki.delph-in.net/moin/PetInputFsc

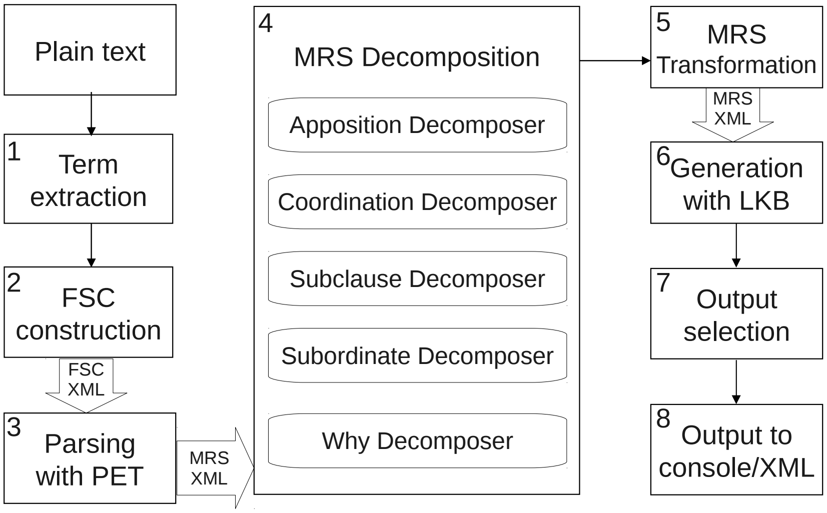


Figure 6: Pipelines of an MRS transformation based question generation system.

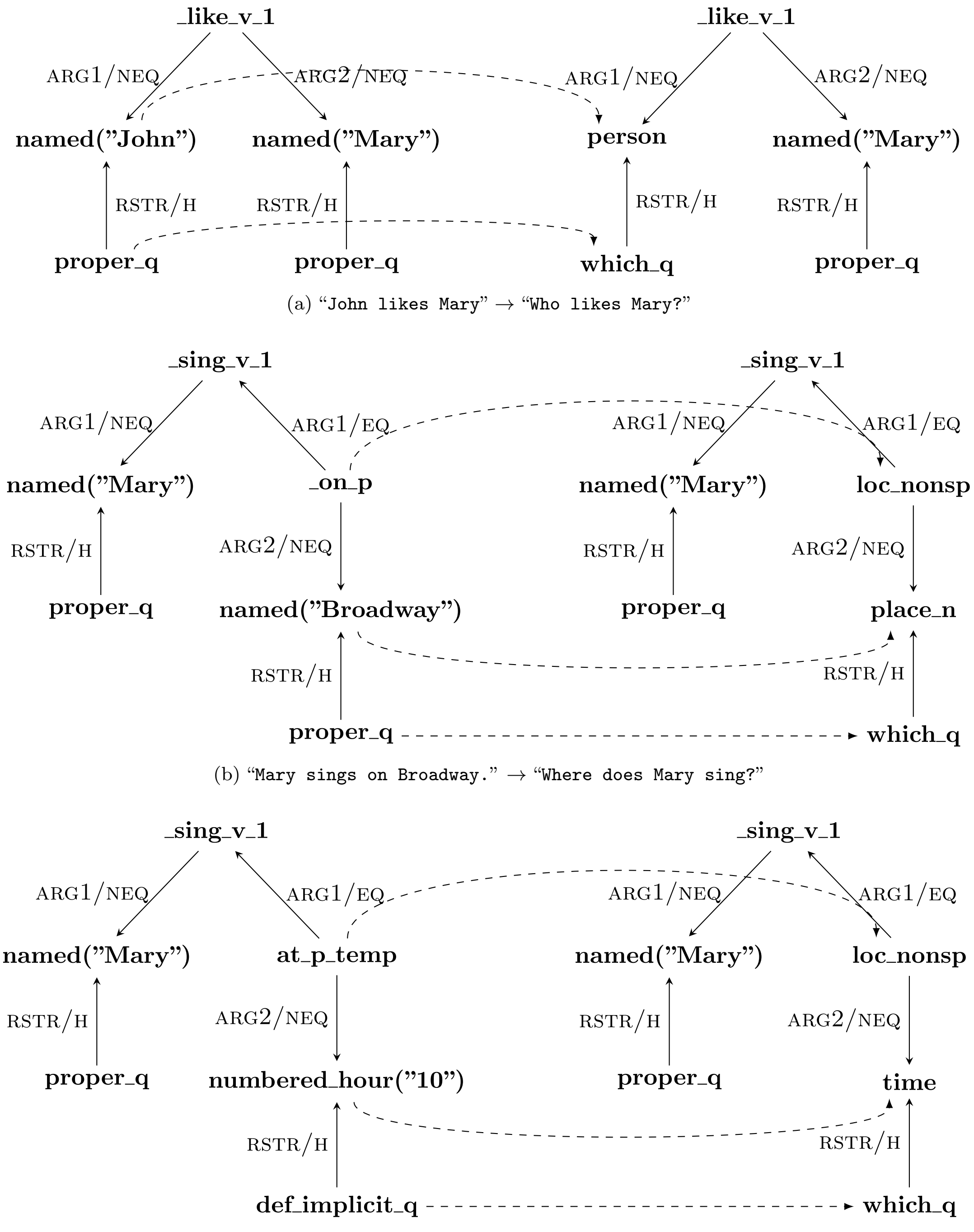
grammatical. Thus there must be ranking algorithms to select the best one. Details are in Section 4.3.

8. Output to console/XML. MrsQG can output its results either to a console for user interaction, or (to a file) in XML format for formal evaluation.

By way of illustration, we now present some actual questions generated by MrsQG:

Example 4 *Jackson was born on August 29, 1958 in Gary, Indiana.*
Generated WHO questions:
(a) Who was born in Gary , Indiana on August 29 , 1958?
(b) Who was born on August 29 , 1958 in Gary , Indiana?
Generated WHERE questions:
(c) Where was Jackson born on August 29 , 1958?
Generated WHEN questions:
(d) When was Jackson born in Gary , Indiana?
Generated YES/NO questions:
(e) Jackson was born on August 29 , 1958 in Gary , Indiana?
(f) Jackson was born in Gary , Indiana on August 29 , 1958?
(g) Was Jackson born on August 29 , 1958 in Gary , Indiana?
(h) Was Jackson born in Gary , Indiana on August 29 , 1958?
(i) In Gary , Indiana was Jackson born on August 29 , 1958?
(j) In Gary , Indiana, was Jackson born on August 29 , 1958?
(k) On August 29 , 1958 was Jackson born in Gary , Indiana?
(l) On August 29 , 1958, was Jackson born in Gary , Indiana?

The examples show that the system generates various question types. Different word orders are allowed by the general ERG used in generation. Not all of these sound equally natural. Section 4.3 addresses this issue. The following sections present more details on steps that need further elaboration.

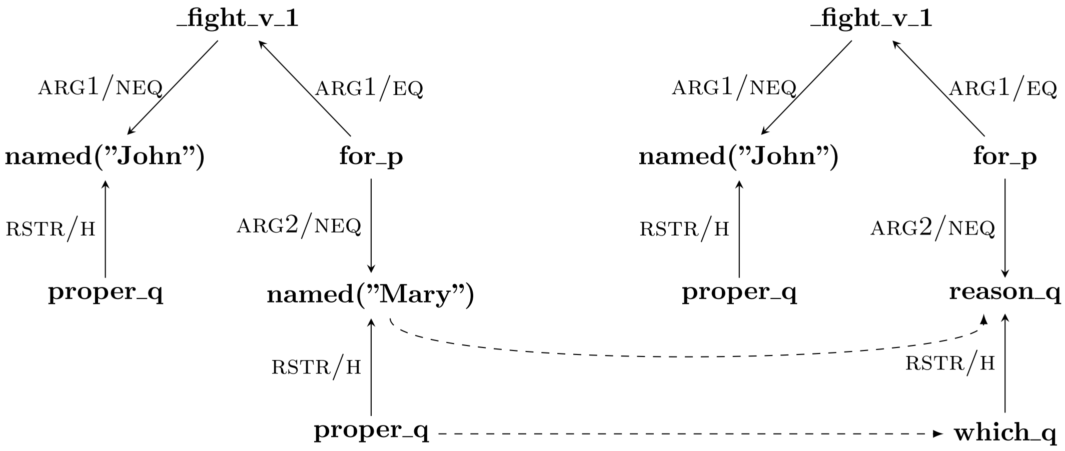


(a) "John likes Mary" → "Who likes Mary?"

(b) "Mary sings on Broadway." → "Where does Mary sing?"

(c) "Mary sings at 10." → "When does Mary sing?"

Figure 7: (continued in the next page)



(d) "`John fights for Mary.`" → "`Why does John fight?`"

Figure 7: MRS transformation from declarative sentences to WH questions in a form of dependency graph.

4.1 MRS Transformation for Simple Sentences

The transformation from declarative sentences into interrogatives follows a mapping between elementary predications (EPs) of relations. Figure 7 shows this mapping. Many terms in preprocessing are tagged as proper nouns (NNP or NNPS). Thus the EPS of a term turns out to consist of two EPs: proper_q_rel (a quantification relation) and named_rel (a naming relation), with proper_q_rel outscoping and governing `named_rel`. The EPs of WH-question words have a similar structure. For instance, the EPs of "`who`" consist of two relations: which_q_rel and person_rel, with which_q_rel outscoping and governing person_rel. Changing the EPs of terms to EPs of WH-question words naturally results in an MRS for WH-questions.

Similarly, in WHERE/WHEN/WHY questions, the EPs for the WH question word are which_q_rel and place_rel/time_rel/reason_rel. Special attention must be paid to the preposition word that usually comes before location/time. In a dependency tree, a preposition word governs the head of the phrase with a post-slash EQ relation (as shown in Figure 7(bcd)). The EP of the preposition must be changed to a loc_nonsp_rel EP (an implicit locative which does not specify a preposition) which takes the WH word relation as an argument in both cases of WHEN/WHERE. This EP avoids generating non-grammatical phrases such as "`in where`" and "`on when`".

Generally, there is one rule per question type. The choice for a transformation rule is triggered by the named entity output. Most questions regarding person/location/time can be asked in two ways: either by WHO/WHERE/WHEN questions or by WHICH PERSON/LOCATION/TIME questions. We encoded both types of these rules in the system. For instance, "`on August 29, 1958`" can be transformed to either "`when`" or "`on which date`". Both questions are generated at this point, and we leave it to the question ranker to select the best one based on properties of the whole question. As we also use an *overgenerate-and-rank* approach, we always try to generate as many questions as possible.

Changing the SF (Sentence Force) attribute of the main event variable from PROP to QUES generates YES/NO questions. This is the simplest case in question generation.

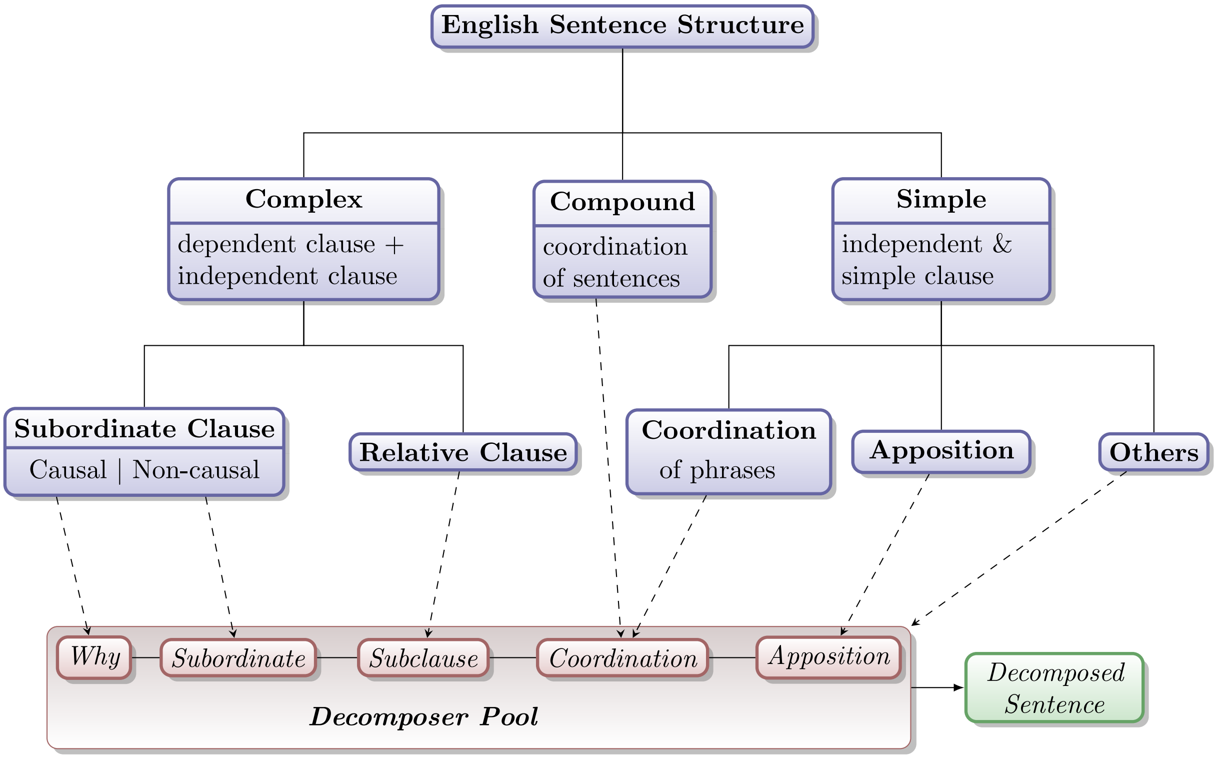


Figure 8: The structure of English sentences and corresponding decomposers (in red boxes). Sentence decomposition does not know the type of sentence beforehand thus all sentences will go through the pool of decomposers. The dashed arrows just indicate which decomposer works on which type of sentence.

4.2 MRS Decomposition for Complex Sentences

4.2.1 Overview

The MRS mapping between declarative and interrogative sentences only works for simple sentences. It generates lengthy questions from complex sentences, as illustrated in Example 4. This is not a desirable result, as too much unnecessary information is provided in the questions. Thus methods must be developed to obtain partial but intact semantic representations from complex sentences, so we can generate from simpler MRS representations. Note, however, that the production of lengthy sentences is not a weakness of the generator, but in essence a problem of sentence simplification. This is what our MRS decomposition rules intend to tackle from the semantic level.

MrsQG employs four decomposers for apposition, coordination, subclause and subordinate clauses. An extra WHY decomposer splits a causal sentence into two parts, reason and result, by extracting the arguments of the causal conjunction word, such as “because”, “the reason”, etc. The distribution of these decomposers is not random but depends on the structure of English sentences.

English sentences are generally categorized into three types: *simple*, *compound* and *complex* depending on the type of clauses they contain. Simple sentences do not contain dependent clauses. Compound sentences are composed of at least two independent clauses. Complex sentences must have at least one dependent clause and one independent clause.

Dependent clauses can be subordinate or relative clauses. Subordinate clauses are typically introduced by a subordinating conjunction, while relatives are typically introduced by relative pronouns.

Other phenomena that call for sentence simplification are coordination and apposition. Coordination can be either clausal or phrasal.

Each type of sentence or grammar construction has a corresponding decomposer, shown in Figure 8:

- Apposition is formed by two adjacent nouns describing the same reference in a sentence. An *apposition decomposer* simplifies the sentence "`Search giant Google was found guilty`" to "`Search giant was found guilty`" and "`Google was found guilty`" in order to prevent an ungrammatical replacement "`*Search giant who was found guilty?`". Note that it is not usually easy to directly replace the compound "`Search giant Google`" with a question word since a named entity recognizer does not work well on compounds or noun phrase chunks.
- Coordination is formed by two or more elements connected with coordinators such as "`and`", "`or`". A *coordination decomposer* simplifies the sentence coordination "`John likes cats and Mary likes dogs`" to "`John likes cats`" and "`Mary likes dogs`" then generates from each simpler sentence. However, it avoids splitting coordination of nouns. For instance, "`the cat and the dog live in the same place`" is not split into "`the cat live in the same place`", which is nonsense and ungrammatical.
- A *subordinate decomposer* works on sentences containing dependent clauses. A dependent clause "depends" on the main clause, or an independent clause. Thus it cannot stand alone as a complete sentence. It starts with a subordinate conjunction and also contains a subject and a predicate. For instance, the sentence "`given that Bart chases dogs, Bart is a brave cat`" is decomposed into two parts: "`Bart chases dogs`" from the subordinate clause and "`Bart is a brave cat`" from the independent clause. Note that in some cases the proposition of the subordinate clause might be changed after decomposition. For instance, extracting "`Bart chases dogs`" from "`if Bart chases dogs, Bart is a brave cat`" makes the proposition "Bart chases dogs" true, while its true value is undetermined in the original sentence.

The following subsection takes the subclause decomposer as an example and illustrates how it works. The order of applying these decomposers is not considered since in question generation from single sentences text cohesion is not important.

4.2.2 Subclause Decomposer

A subclause decomposer works on sentences that contain relative clauses, such as this one. A relative clause is mainly indicated by relative pronouns, i.e., *who*, *whom*, *whose*, *which*, *whomever*, *whatever*, and *that*. Extracting relative clauses from a sentence helps to ask better questions. For instance, given the following sentence:

Example 5 *(a)* *`Bart is the cat that chases the dog.`*

Extracted relative clause after decomposition:

(b) *`The cat chases the dog.`*

Parsing the above clause into logic form:

(c) chase(cat, dog)

Replacing named entities with question words:

(d) chase(which animal, dog) and chase(cat, which animal)

Generated questions from the transformed logic form:

(e) *`Which animal chases the dog?`*

(f) *`Which animal does the cat chase?`*

It is impossible to ask a short question such as (e) and (f) directly from the original sentence (a) without dropping the main clause. A subclause decomposer serves to change this situation.

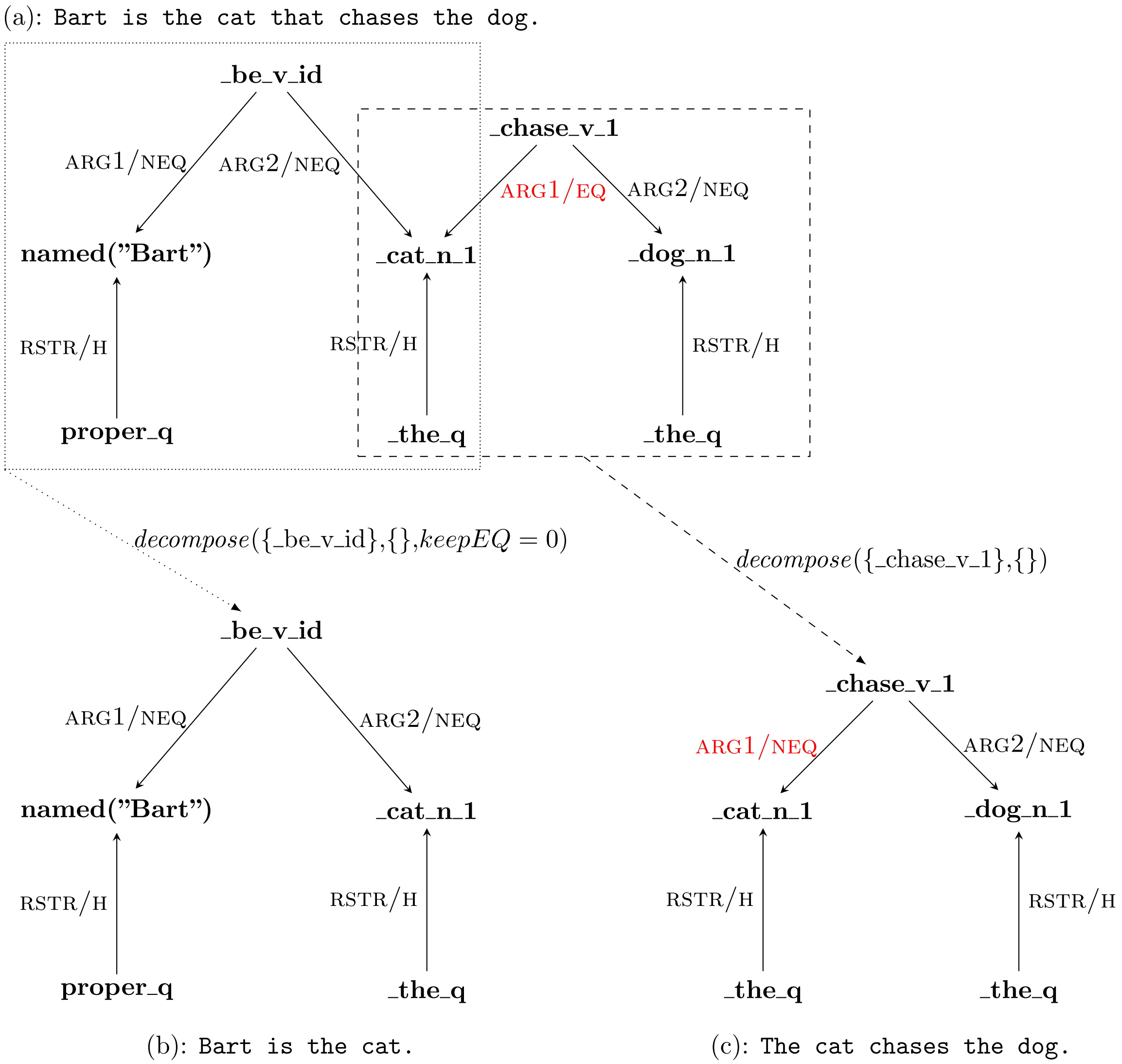
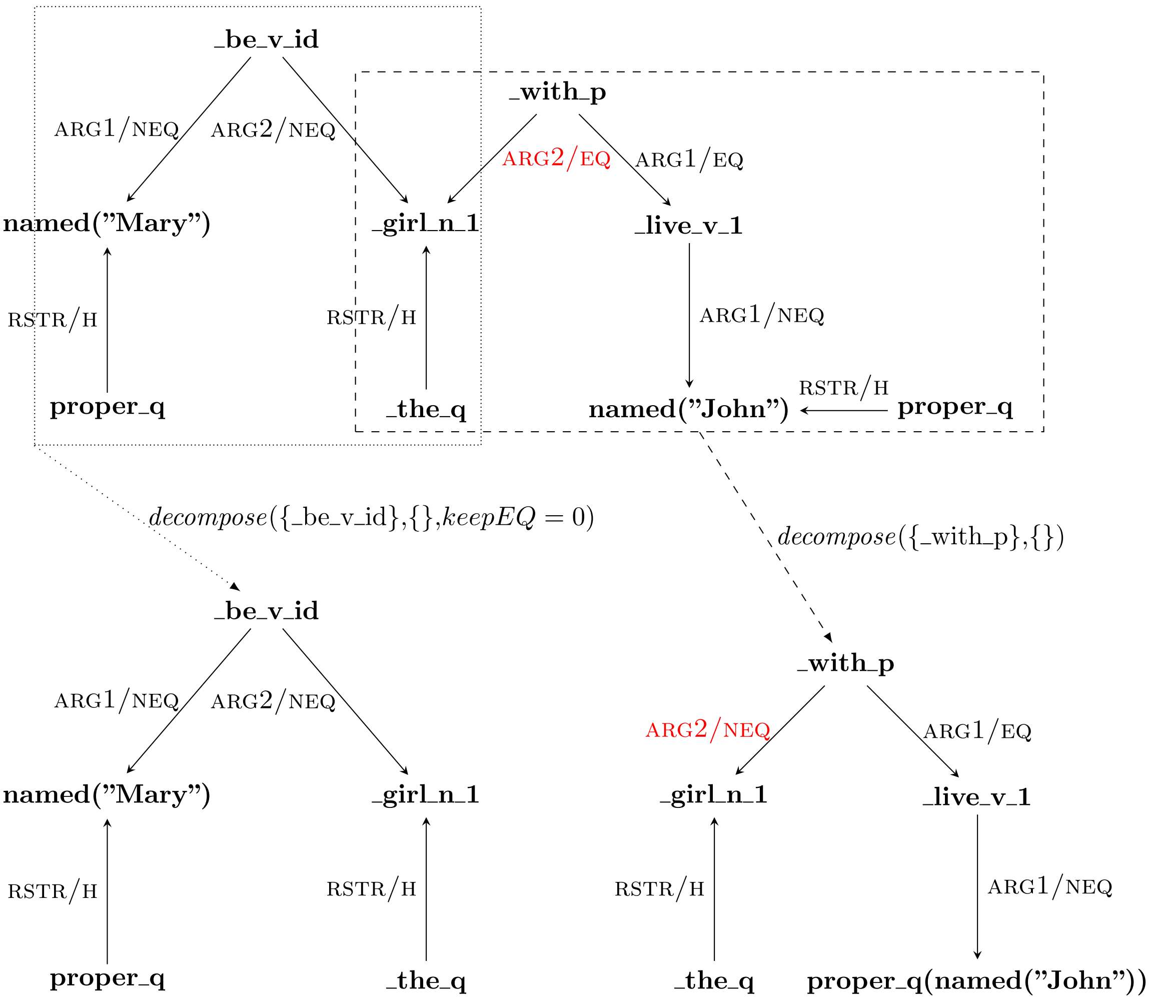


Figure 9: A subclause decomposer extracts relative clauses by finding all non-verb EPs that are directly related to a verb EP. It also relaxes scope constraint from a relative clause. A strict EQ relation in the top graph represents an NP "the cat that chases the dog". Relaxing it to NEQ generates a sentence "the cat chases the dog." in the bottom right graph.

(a): `Mary is the girl John lives with.`



(b): `Mary is the girl.`

(c): `John lives with the girl.`

Figure 10: Subclause decomposition for relative clauses with a preposition. The preposition EP _with_p connects the subclause with the main clause. After decomposition, the dependency relation between _with_p and _girl_n_1 is relaxed from ARG2/EQ in (a) to ARG2/NEQ in (c).

Though relative pronouns indicate relative clauses, in an MRS structure, these relative pronouns are not explicitly represented. For instance, in Figure 9(a), there is no EP for the relative pronoun ”`that`”. However, the verb EP _chase_v_1 governs its subject by a post-slash EQ relation. This indicates that _chase_v_1 and _cat_n_1 share the same label and have the same scope. After decomposing the sentence, this constraint of the same scope should be relaxed. Thus in the MRS of “`The cat chases the dog.`”, _chase_v_1 and _cat_n_1 have different scopes, indicated by a post-slash NEQ relation. A generic decomposition algorithm should have a scope-relaxing step at the final stage.

It is also possible to form a relative clause by leaving out the object of a preposition. Although there is usually a relative pronoun in such cases (Example 6a), it can also be left out (Example 6b):

Example 6 *(a) Mary is the girl with whom John lives.* (with a relative pronoun)
(b) Mary is the girl John lives with. (zero relative pronoun)

The ERG produces an identical MRS for such cases. The preposition is the word that connects the relative clause to the main clause. Thus the subclause decomposer first starts from the preposition rather than the verb in the relative clause, as shown in Figure 10.

4.2.3 General Algorithm

In this subsection we describe the general algorithm for sentence decomposition. This generic algorithm is the key step for decomposing complex MRS structures. Before describing it, we first sum up the dependencies between EPs and their intricacies. Note that the following is only a list of examples and is incomplete.

- ARG*/EQ: a special case that indicates scope identity.
  - Adjectives govern nouns, such as _brave_a_1 → _cat_n_1 from “`a brave cat`”.
  - Adverbs govern verbs, such as _very+much_a_1 → _like_v_1 from “`like ... very much`”.
  - Prepositions govern and attach to verbs, such as _with_p → _live_v_1 from “`live with ...`”.
  - Passive verbs govern and modify phrases, such as _stir_v_1 → _martini_n_1 from “`a stirred martini`”.
  - In a relative clause:
    - Verbs govern and modify phrases that are represented by relative pronouns, such as _chase_v_1 → _cat_n_1 from “`the cat that chases ...`” in Figure 9.
    - Prepositions govern phrases that are represented by relative pronouns, such as _with_p → _girl_n_1 from “`the girl whom John lives with`”.
- ARG*/NEQ: the most general case that underspecifies scopes.
  - Verbs govern nouns (subjects, objects, indirect objects, etc), such as _chase_v_1 → named(“Bart”) and _chase_v_1 → _dog_n_1 from “`Bart chases dogs`”.
  - Prepositions govern phrases after them, such as _with_p → _girl_n_1 from “`lives with the girl`” in Figure 10(c).
  - Some grammatical construction relations govern their arguments, such as compound_name → named(“Google”) from “`Search giant Google`”.
- ARG*/NULL: a special case where an argument is empty.
  - Passive verbs govern and modify phrases, e.g. in _chase_v_1 → _dog_n_1 from “`the chased dog`” and “`the dog that was chased`”, _chase_v_1 has no subject.

- ARG*/H: a rare case where an argument qeq an EP.
  - Subordinating conjunctions govern their arguments, such as _given+that_x_subord → _be_v_id from "`Given that` ARG2, ARG1.".
  - Verbs govern verbs, such as _tell_v_1_rel → _like_v_1_rel from "`John told Peter he likes Mary.`".
- NULL/EQ: a rare case where two EPs share the same label but preserve no equalities.
- RSTR/H: a common case for every noun phrase.
  - A noun is governed by its quantifier through a qeq relation, such as proper_q → named("Bart") from "`Bart`".
- L|R-INDEX/NEQ: a common case for every coordinating conjunction.
  - Coordinating conjunctions govern their arguments, such as _and_c → _like_v_1 from "L-INDEX `and` R-INDEX".
- L|R-HNDL/HEQ: a special case for coordination of verb phrases. Coordination of noun phrases do not have these relations.
  - The governor's argument is its dependent's label, such as _but_c → _like_v_1 from "L-HNDL `but` R-HNDL".

We first give a formal definition of a connected DMRS graph, then a generic algorithm for DMRS decomposition.

Connected DMRS Graph
A Connected DMRS Graph is a tuple $G = (N, E, L, S_{pre}, S_{post})$ of:
a set N, whose elements are called *nodes;*
a set E of connected pairs of vertices, called *edges;*
a function L that returns the associated label for edges in E;
a set S_{pre} of pre-slash labels and a set S_{post} of post-slash labels.
Specifically,
N is the set of all Elementary Predications (EPs) defined in a grammar;
S_{pre} contains all pre-slash labels, namely {ARG*, RSTR, L-INDEX, R-INDEX, L-HNDL, R-HNDL, NULL};
S_{post} contains all post-slash labels, namely {EQ, NEQ, H, HEQ, NULL};
L is defined as: $L(x, y) = [pre/post, \ldots]$. For every node $x, y \in N$, L returns a list of pairs *pre*/*post* that $pre \in S_{pre}, post \in S_{post}$. If $pre \neq$ NULL, then the edge between (x, y) is directed: x is the governor, y is the dependant; otherwise the edge between x and y is not directed. If $post =$ NULL, then $y =$ NULL, x has no dependant by a *pre* relation.

Algorithm 1 shows how to find all related EPs for some target EPs. It is a graph traversal algorithm starting from a set of target EPs for which we want to find related EPs. It also accepts a set of exception EPs that we always want to exclude. Finally it returns all related EPs to the initial set of target EPs.

There are two optional parameters for the algorithm that define different behaviors of graph traversal. *relaxEQ* controls whether we want to relax the scope constraints from /EQ in a subclause to /NEQ in a main clause. It works on both verbs and prepositions that head a relative clause. Examples of this option taking effect can be found in Figure 9(c) and 10(c). *keepEQ* controls whether we want to keep a subclause introduced by a verb or preposition. It is set to true by

Algorithm 1 A generic decomposing algorithm for connected DMRS graphs.

```
function decompose($rEPS$, $eEPS$, $relaxEQ = 1$, $keepEQ = 1$)

parameters:
$rEPS$: a set of EPs for which we want to find related EPs.
$eEPS$: a set of exception EPs.
$relaxEQ$: a boolean value of whether to relax the post-slash value from EQ to NEQ for verbs and prepositions (optional, default:1).
$keepEQ$: a boolean value of whether to keep verbs and prepositions with a post-slash EQ value (optional, default:1).
returns: a set of EPs that are related to $rEPS$
;; assuming concurrent modification of a set is permitted in a for loop
$aEPS \leftarrow$ the set of all EPs in the DMRS graph
$retEPS \leftarrow \emptyset$ ;; initialize an empty set
for $tEP \in rEPS$ and $tEP \notin eEPS$ do
  for $ep \in aEPS$ and $ep \notin eEPS$ and $ep \notin rEPS$ do

    $pre/post \leftarrow L(tEP, ep)$ ;; $ep$ is the dependant of $tEP$
    if $pre \neq$ NULL then ;; $ep$ exists
      if $relaxEQ$ and $post =$ EQ and ($tEP$ is a verb EP or ($tEP$ is a preposition EP and $pre =$ ARG2)) then
        assign $ep$ a new label and change its $qeq$ relation accordingly
      end if
      $retEPS$.add($ep$) , $aEPS$.remove($ep$)
    end if

    $pre/post \leftarrow L(ep, tEP)$ ;; $ep$ is the governor of $tEP$
    if $pre \neq$ NULL then ;; $ep$ exists
      if $keepEQ = 0$ and $ep$ is a (verb EP or preposition EP) and $post =$ EQ and $ep$ has no empty ARG* then
        continue ;; continue the loop without going further below
      end if
      if not ($ep$ is a verb EP and $post =$ NEQ or $post =$ H) then
        $retEPS$.add($ep$) , $aEPS$.remove($ep$)
      end if
    end if

  end for
end for

if $retEPS \neq \emptyset$ then
  return decompose($rEPS \cup retEPS$, $eEPS$, $relaxEQ = 0$) ;; the union of two
else
  return $rEPS$
end if
```

default. If set to false, the relative clause will be removed from the main clause. Examples can be found in Figure 9(b) and 10(b).

All decomposers except the apposition decomposer employ this generic algorithm. Figures 9 to 10 are marked with corresponding function calls of this algorithm that decompose one complex MRS structure to two simpler ones.

4.3 Automatic Generation with Rankings

MrsQG usually produces too much output. The reasons are twofold: firstly, PET and ERG "overparse". Sometimes even a simple sentence has tens of parsed MRS structures due to subtle distinctions in the grammar. Secondly, LKB and ERG overgenerate. Sometimes even a simple MRS structure has tens of realizations, due to word order freedom, lexical choice, etc. Thus ranking is needed to select the best output.

Generation from LKB has already incorporated the MaxEnt model by Velldal and Oepen (2006), which works best for declarative sentences as its training corpus does not include questions. To rank questions, we could have trained the MaxEnt model on questions. But this requires a syntactically annotated (in HPSG treebank style) question corpus, which is time consuming and expensive to create. Thus we took a shortcut to simply use language models trained on questions only.

To construct a corpus consisting of only questions, data from the following sources was collected:

- the Question Answering track (1999-2007)[4] of the Text REtrieval Conference (TREC, Voorhees, 2001)
- the Multilingual Question Answering campaign (2003-2009)[5] from the Cross-Language Evaluation Forum (CLEF, Braschler and Peters, 2004)
- a Question Classification (QC) dataset (Li and Roth, 2002, Hovy et al., 2001)[6]
- a collection of Yahoo!Answers (Liu and Agichtein, 2008)[7]

The whole language model training procedure follows a standard protocol of "*build-adapt-prune-test*". Firstly we *build* a small language model for questions only. Then this language model is *adapt*ed with the whole English Wikipedia[8] to increase lexicon coverage. Finally we *prune* it for rapid access and *test* it for effectiveness. The IRST Language Modeling Toolkit (Federico and Cettolo, 2007) was used.

To combine the scores from both the MaxEnt model and the language model, we first project the log-based MaxEnt score to linear space and then we use a weighted formula for linear scores to combine the linear MaxEnt score with the sentence probability from the language model.

Suppose for a single MRS representation there are N different realizations $R = [r_1, r_2, \dots r_N]$. $P(r_i|\textsc{me})$ is the normalized MaxEnt score for the i-th realization under a Maximum Entropy model ME. $P(r_i|\textsc{lm})$ is the probability of each of the N different realizations given the language model LM. With two rankings $P(r_i|\textsc{me})$ and $P(r_i|\textsc{lm})$ we can borrow the idea of *F-measure* from information retrieval and combine them:

$$R(r_i) = F_\beta = (1+\beta^2)\frac{P(r_i|\textsc{me})P(r_i|\textsc{lm})}{\beta^2 P(r_i|\textsc{me}) + P(r_i|\textsc{lm})} \tag{1}$$

With $\beta = 1$ the ranking is unbiased. With $\beta = 2$ the ranking weights $P(r_i|\textsc{lm})$ twice as much as $P(r_i|\textsc{me})$.

[4] http://trec.nist.gov/data/qamain.html

[5] http://celct.isti.cnr.it/ResPubliQA/index.php?page=Pages/pastCampaigns.php

[6] http://l2r.cs.uiuc.edu/~cogcomp/Data/QA/QC/

[7] http://ir.mathcs.emory.edu/shared/

[8] http://meta.wikimedia.org/wiki/Wikipedia_Machine_Translation_Project

	Interrogatives					Declaratives
	TREC	CLEF	QC	YahooAns	All	Wikipedia
Sentence count	4,950	13,682	5,452	581,348	605,432	30,716,301
Word count (K)	36.6	107.3	61.1	5,773.8	5,978.9	722,845.6
Words/Sentence	7.41	7.94	11.20	9.93	9.88	23.53

Table 1: Statistics of the data sources. "All" is the sum of the first four datasets. Wikipedia is about 120 times larger than "All" in terms of words.

Figure 11 illustrates how question ranking works. Since the MaxEnt model is only trained on declarative sentences, it prefers sentences with a declarative structure. However, the language model trained with questions prefers auxiliary fronting. The weighted score can help to select the best interrogative sentences.

5. Evaluation

The evaluation of question generation with semantics was conducted as part of the Question Generation Shared Task and Evaluation Challenge (QGSTEC2010; Rus et al., 2010; Rus et al., this volume). Participants are given a set of inputs consisting of an input sentence + question type and their system should generate two questions for each type. Question types include yes/no, which, what, when, how many, where, why and who. Input sources are Wikipedia, OpenLearn[9] and Yahoo! Answers. Each source contributes 30 input sentences. There will be 360 questions generated in total.

Evaluation was conducted by independent human raters (but not necessarily native speakers). They follow the following criteria:

1. Relevance. Questions should be relevant to the input sentence. Best/worse score: 1/4.
2. Question type. Questions should be of the specified target question type. Best/worse score: 1/2.
3. Syntactic correctness and fluency. The syntactic correctness is rated to ensure systems can generate sensible output. Best/worse score: 1/4.
4. Ambiguity. The question should make sense when asked more or less out of the blue. Best/worse score: 1/3.
5. Variety. Pairs of questions in answer to a single input are evaluated on how different they are from each other. Best/worse score: 1/3.

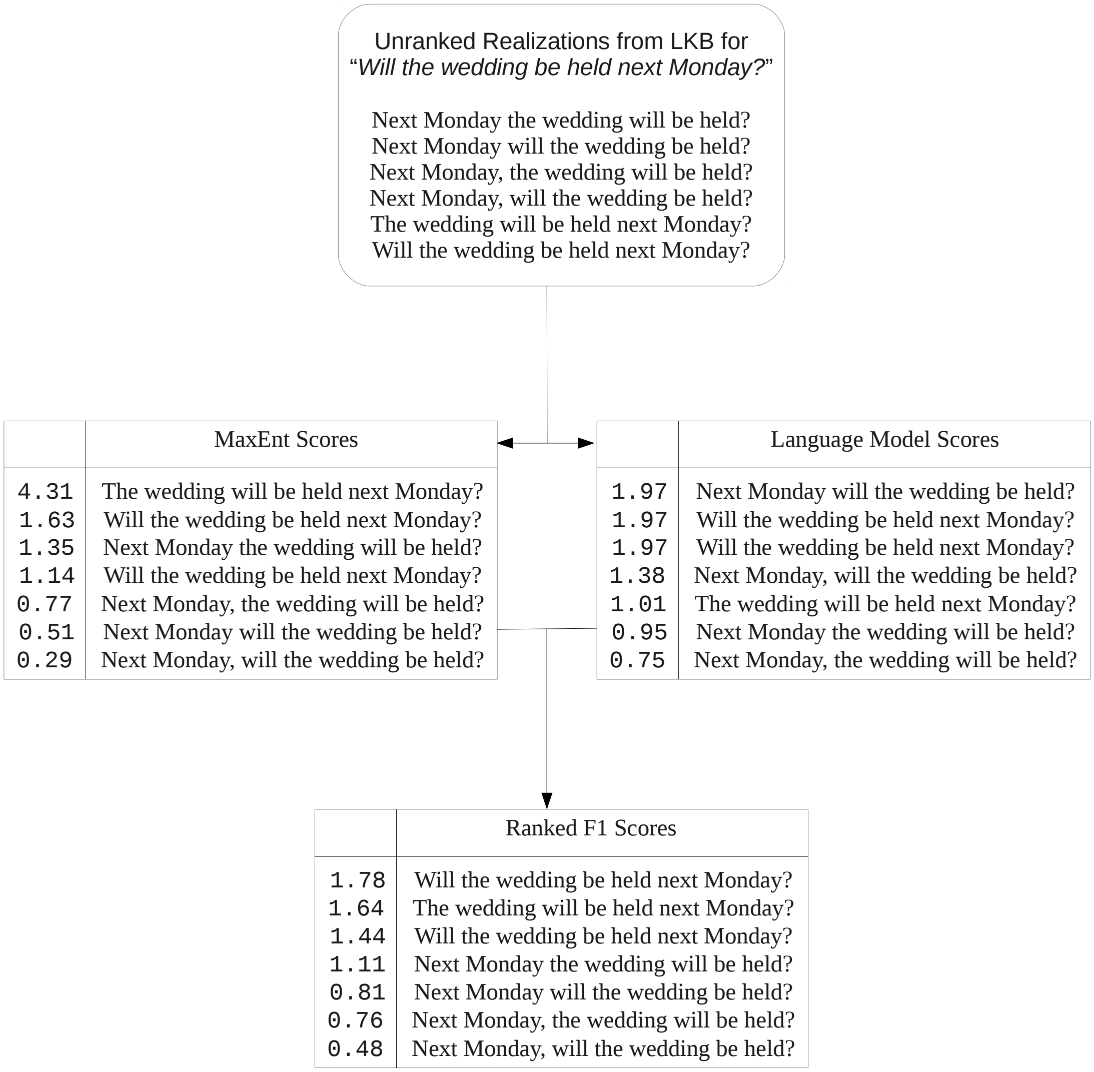
Note that all participants were asked to generate two questions of the same type. If only one question was generated, then the variety ranking of this question receives the lowest score and the missing question receives the lowest scores for all criteria.

5.1 Evaluation Results

Four systems participated in Task B of QGSTEC2010:

- Lethbridge (Ali et al., 2010), University of Lethbridge, Canada

[9] http://openlearn.open.ac.uk



Unranked Realizations from LKB for
“*Will the wedding be held next Monday?*”

Next Monday the wedding will be held?
Next Monday will the wedding be held?
Next Monday, the wedding will be held?
Next Monday, will the wedding be held?
The wedding will be held next Monday?
Will the wedding be held next Monday?

	MaxEnt Scores
4.31	The wedding will be held next Monday?
1.63	Will the wedding be held next Monday?
1.35	Next Monday the wedding will be held?
1.14	Will the wedding be held next Monday?
0.77	Next Monday, the wedding will be held?
0.51	Next Monday will the wedding be held?
0.29	Next Monday, will the wedding be held?

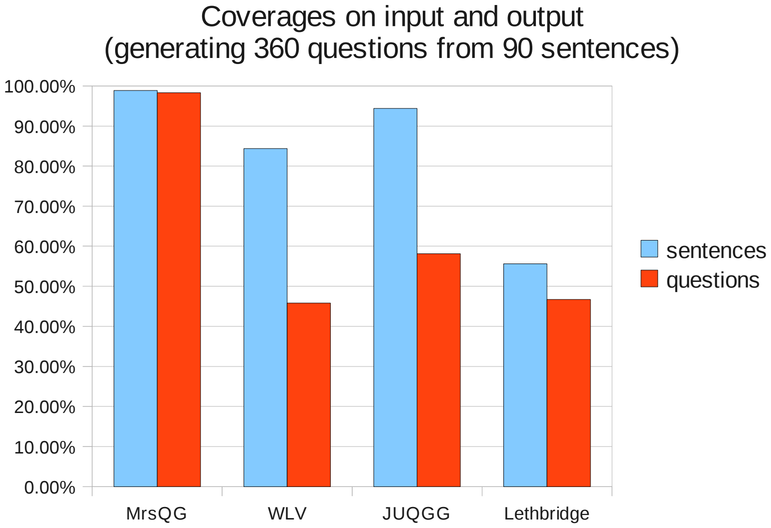
	Language Model Scores
1.97	Next Monday will the wedding be held?
1.97	Will the wedding be held next Monday?
1.97	Will the wedding be held next Monday?
1.38	Next Monday, will the wedding be held?
1.01	The wedding will be held next Monday?
0.95	Next Monday the wedding will be held?
0.75	Next Monday, the wedding will be held?

	Ranked F1 Scores
1.78	Will the wedding be held next Monday?
1.64	The wedding will be held next Monday?
1.44	Will the wedding be held next Monday?
1.11	Next Monday the wedding will be held?
0.81	Next Monday will the wedding be held?
0.76	Next Monday, the wedding will be held?
0.48	Next Monday, will the wedding be held?

Figure 11: Combined ranking using scores from a MaxEnt model and a language model. The final combined scores come from Equation 1 with $\beta = 1$. All scores are multiplied by 10 for better reading. Note that the question “`Will the wedding be held next Monday?`” appears twice and has different scores with the MaxEnt model. This is because the internal generation chart is different and thus it is regarded as different, even though the lexical wording is the same.

	input sentences				output questions			
	count	%(/90)	mean	std	count	%(/360)	mean	std
MrsQG	89	98.9	19.27	6.94	354	98.3	12.36	7.40
WLV	76	84.4	19.36	7.21	165	45.8	13.75	7.22
JUQGG	85	94.4	19.61	6.91	209	58.1	13.32	7.34
Lethbridge	50	55.6	20.18	5.75	168	46.7	8.26	3.85

(a)



(b)

Table 2: Generation coverage of four participants. Each was supposed to generate 360 questions in all from 90 sentences. “mean” and “std” indicate the average and standard deviation of the input sentence length and output question length in words.

- MrsQG, Saarland University, Germany
- JUQGG (Pal et al., 2010), Jadavpur University, India
- WLV (Varga and Ha, 2010), University of Wolverhampton, UK

In this section we describe the evaluation result of MrsQG in comparison with other systems.

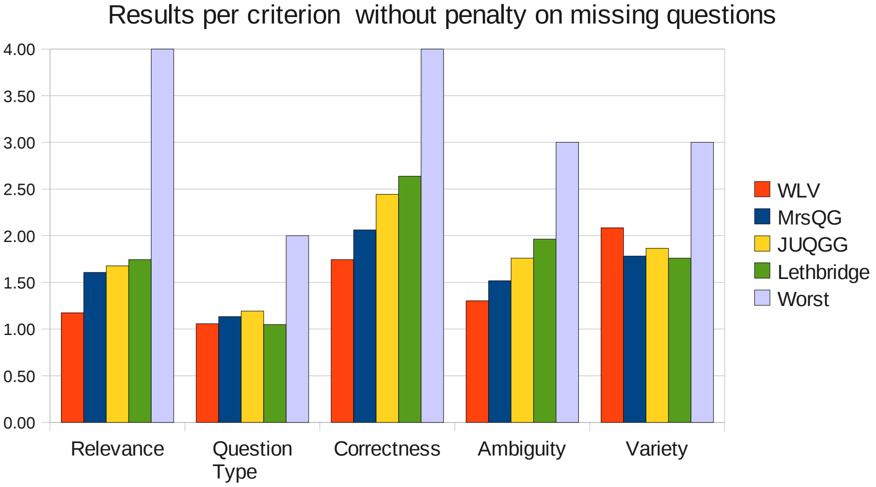
5.1.1 Generation Coverage

The low coverage number in most cases of Table 2 shows that generation from all input sentences and from all required question types was not an easy task. None of the systems reached 100%. Among them, MrsQG, WLV and JUQGG generated from more than 80% of the 90 sentences but Lethbridge only managed to generate from 55.6% of them. As for the required 360 questions, WLV, JUQGG and Lethbridge only generated around 40% ˜ 60% of them. Most systems can respond to the input sentences but only MrsQG has a good coverage on required output questions. Figure 2b in Table 2 illustrates this.

Good coverage on required questions usually depends on whether the employed named entity recognizer is able to identify corresponding terms for a question type and whether the reproduction

	Relevance	Question Type	Correctness	Ambiguity	Variety
MrsQG	1.61	1.13	2.06	1.52	1.78
WLV	1.17	1.06	1.75	1.30	2.08
JUQGG	1.68	1.19	2.44	1.76	1.86
Lethbridge	1.74	1.05	2.64	1.96	1.76
best/worst	1/4	1/2	1/4	1/3	1/3
Agreement	63%	88%	46%	55%	58%

(a)



(b)

Table 3: Results per participant without penalty for missing questions. Lower grades are better. Agreement closer to 100% indicates better inter-rater reliability. ”Worst” indicates the worst possible scores per evaluation category.

rule covers enough structural variants. It also depends on whether the systems have been tuned on the development set and the strategy they employed. For instance, in order to guarantee high coverage, MrsQG can choose to sacrifice some performance in sentence correctness. Some systems, such as WLV, seemed to focus on performance rather than coverage. The next subsection shows this point.

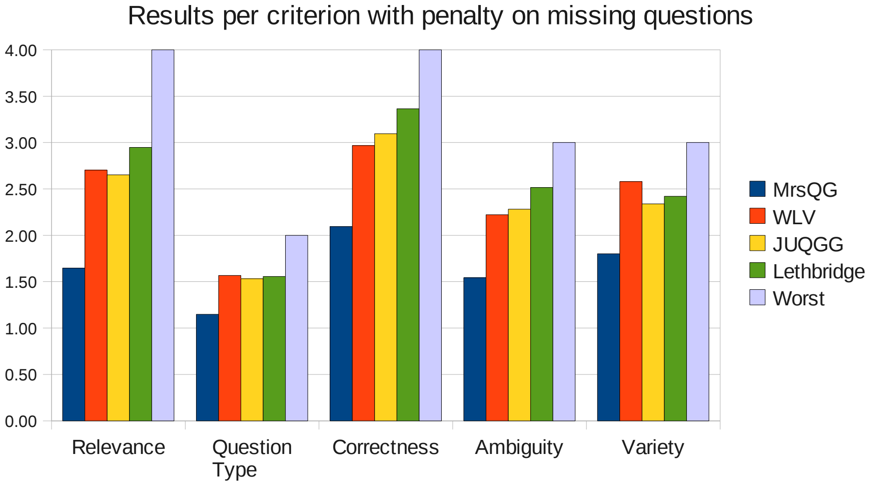
5.1.2 Overall Evaluation Grades

Two human raters gave grades to each question according to the established criteria. Then all grades were averaged and inter-rater agreement was calculated. Due to the fact that most systems do not have a good coverage on required questions (c.f. Table 2), the final grades were calculated with and without penalty on missing questions.

Table 3 presents the results without penalty on missing questions. Grades were calculated based only on *generated* questions from each system. Out of all grading criteria, syntactic correctness and fluency appears to be the hardest for all systems. Apparently, syntactic or semantic transformations do not guarantee grammaticality and fluency of the generated question. The best scores are obtained for question type and relevance. This is not surprising, as the question type is given and the questions are generated on the basis of a single sentence.

	Relevance	Question Type	Correctness	Ambiguity	Variety
MrsQG	1.65	1.15	2.09	1.54	1.80
WLV	2.70	1.57	2.97	2.22	2.58
JUQGG	2.65	1.53	3.10	2.28	2.34
Lethbridge	2.95	1.56	3.36	2.52	2.42

(a)



(b)

Table 4: Results per participant with penalty for missing questions. Lower grades are better. ”Worst” indicates the worst possible scores per evaluation category.

Table 4 shows the results if all questions that should be generated are taken into account. If a system failed to generate a question, it is assumed this system generates a question with the worst scores in all criteria. Since WLV, JUQGG and Lethbridge have relatively low coverage (between 40% ˜ 60%), their scores deteriorate considerably. The scores for MrsQG are not affected that much, as it has a coverage of 99%.

The inter-rater agreement for the scores shown by Table 3 is not satisfactory. A score of over 80% usually indicates good agreement but only question type has achieved this standard. Landis and Koch (1977) have argued values between 0–.20 as slight, .21–.40 as fair, .41–.60 as moderate, .61–.80 as substantial, and .81–1 as almost perfect agreement. According to this standard, all the agreement numbers for other criteria only show a moderate or weakly substantial agreement between raters. These values reflect the fact that the rating criteria were ambiguously defined.

5.1.3 Evaluation Grades per Question Type

QGSTEC2010 requires eight types of questions: yes/no, which, what, when, how many, where, why and who. Figure 12 shows the individual scores of MrsQG on these questions. The quality of these questions mostly depends on the term extraction component. For instance, if a named entity recognizer fails or the ontology cannot provide a hypernym for a term, a which question cannot be properly generated. In general, MrsQG did the worst for which questions and the best for who questions. This reveals the strong and weak points of the term extraction component employed in

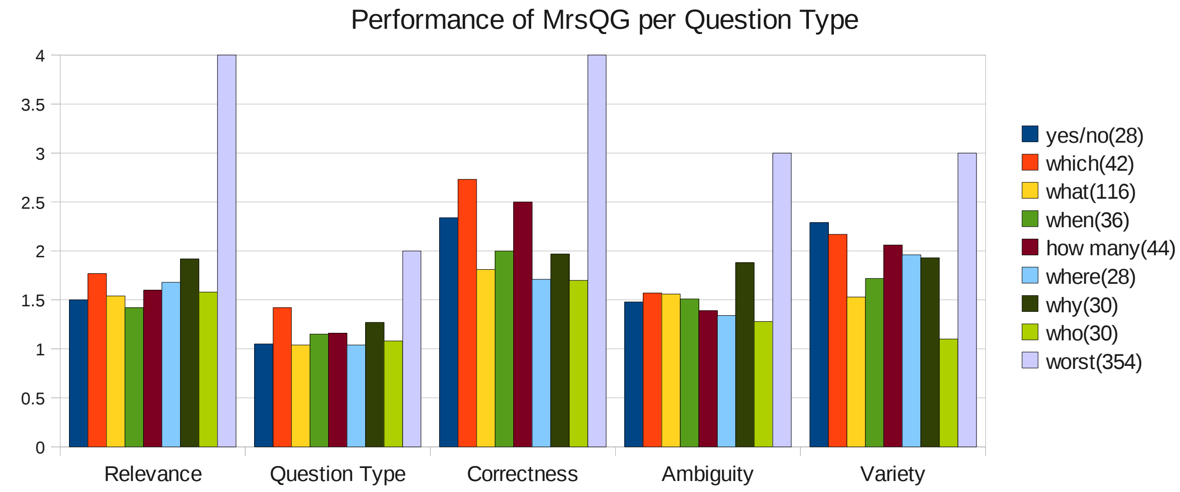


Figure 12: Performance of MrsQG per question type. Numbers in () indicate how many questions were generated regarding each question type. The number in worst() is the sum of all questions. ”Worst” indicates the worst possible scores per evaluation category.

MrsQG. Also, why questions involve more reasoning than others. Since MrsQG does not have a reasoning component, it received the worst score in “ambiguity” on why questions.

6. Discussion

This section discusses various aspects in question generation with semantics. Some of the discussion is based on a theoretical perspective while some involves implementation issues. The merits of deep processing with semantics are emphasized but their disadvantages are also addressed.

6.1 Generation with Semantics can Produce Better Sentences with Fewer Rules

One semantic representation can lead to several surface realizations. With a good ranking mechanism the best realization can be selected, which makes the generated question even more natural than the input sentence in some cases. Take the English active and passive voice as an example:

Example 7 *(a)* *The dog was chased by Bart.*
Question generated from syntactic transformation:
(b) *By whom was the dog chased?*
Extra question generated from semantic transformation:
(c) *Who chased the dog?*

By replacing the term with question words and fronting auxiliary verbs and question words, a syntax-based system will normally only generate questions as good as (b). A semantics-based system, generating from chase(who, the dog), can produce both the passive form in (b) and the active form in (c). In most contexts, (c) would be preferred. (c) is shorter than (b) and sounds more natural.

One might argue that with special treatment and carefully tested transformation rules, a syntax-based system is capable of generating questions both in active and passive voices. But then again, similar problems arise in the case of di-transitive verbs:

Example 8 *(a)* *John gave the waitress a one-hundred-dollar tip.*

Asking a question on the tip giver:
(b) `Who gave the waitress a one-hundred-dollar tip?`
Extra question generated from semantics-based system:
(c) `Who gave a one-hundred-dollar tip to the waitress?`

A syntax-based system should have no problems producing (b). However, an additional rule is required to generate (c). For a semantics-based system, both (b) and (c) can be generated from the logical form give(who, one-hundred-dollar tip, waitress) and no special rules are required.

Generally in a semantics-based system, the generator is responsible for realizing a sentence in all the ways a language's grammar permits. The question transformer only needs to address the semantic divergence between different questions. Thus, a semantics-based system in general requires fewer rules to generate more questions (i.e. more word order variation) than a syntax-based system.

6.2 Interface to Lexical Semantics Resources

Although one semantic representation can produce multiple syntactic realizations, this procedure is handled internally through chart generation with ERG. Also, different syntactic realizations of the same meaning can be symbolized into the same semantic representation through parsing with ERG. Thus MrsQG is relieved of the burden to secure grammaticality and is able to focus on the semantics of languages. This in turn makes MrsQG capable of incorporating other lexical semantics resources seamlessly. For instance, given the predicate logic form have(John, dog) from the sentence "`John has a dog`" and applying ontologies to recognize that a dog is an animal, a predicate logic form have(John, what animal) for the question "`what animal does John have`" can be naturally derived.

Apart from using hypernym-hyponym[10] relations to produce which questions, other lexical semantic relations can be employed as well. Holonym-meronym[11] relations can help ask more specific questions. For instance, given that a clutch is a part of a transmission system and a sentence "`a clutch helps to change gears`", a more specific question "`what part of a transmission system helps to change gears?`" instead of merely "`what helps to change gears?`" can be asked. Also, given the synonym of the cue word the task of lexical substitution can be performed to produce more lexical variations in a question.

Using lexical semantics resources needs the attention of a word sense disambiguation component. A misidentification of the sense of word might lead to nonsense questions. So expanding the range of questions with lexical semantics resources should be used with caution.

Note that there is no distinct difference between semantics-based system and syntax-based system in the ability to incorporate lexical semantics resources. But in terms of expressive simplicity and ease of use, a semantics-based system seems to us a more natural way to utilize lexical semantics resources.

6.3 Language Independence and Domain Adaptability

In theory, MrsQG is language-neutral as it is based on semantic transformations. As long as there is a grammar[12] conforming with the HPSG structure and LKB, adapting it to other languages should require little or no modification. However, the experience in multi-lingual grammar engineering has shown that although MRS offers a higher level of abstraction than syntax, it is difficult to guarantee absolute language independence. As a syntax-semantics interface, part of the MRS representation will inevitably carry some language specificity. As a consequence, the MRS transfer rules need to be adapted for the specific grammars, similar to the situation in MRS-based machine translation (Oepen et al., 2004).

The domain adaptability is confined to the following parts:

[10] Y is a hypernym of X if every X is a (kind of) Y; Y is a hyponym of X if every Y is a (kind of) X.

[11] Y is a holonym of X if X is a part of Y; Y is a meronym of X if Y is a part of X.

[12] For a list of available grammars, check `http://wiki.delph-in.net/moin/MatrixTop`

1. Named entity recognizers. For a different domain, the recognizers must be re-trained. MrsQG also uses an ontology-based named entity recognizer. Thus collections of domain-specific named entities can be easily plugged-in to MrsQG.

2. HPSG parser. The PET parser needs to be re-trained on a new domain with an HPSG treebank. However, since the underlying HPSG grammars are mainly hand-written, they normally generalize well and have a steady performance on different domains.

6.4 Limitations of Proposed Method

A semantics-based question generation system is theoretically sound and intuitive. But the implementation is limited to tools that are currently available, such as the grammar, the parser, the generator and the preprocessors. The central theme of MrsQG is MRS, which is an abstract syntactic-semantic interface employed by ERG. The parser and generator cannot work without the ERG either. Thus the ERG is indeed the backbone of the whole system. The heavy machinery employed by a deep precision grammar decreases both the parsing and generation speed and requires large memory footprints. Thus MrsQG needs more resources and time to process the same sentence than a syntax-based system does.

On the parsing side, MrsQG is not robust against ungrammatical sentences, due to the fact that ERG is a rule-based grammar and only accepts grammatical sentences. The grammar coverage also decides the system performance in terms of recall value. But since ERG has been developed for over ten years with great effort, robustness against ungrammaticality and rare grammatical constructions is only a minor limitation to MrsQG.

On the generation side, all parsed MRS structures in theory should generate. But there exists the problem of overgeneration. As shown by Velldal and Oepen (2006), the MaxEnt model achieves a 64.28% accuracy on the best sentence and 83.60% accuracy on the top-5 best sentences. Thus the question given by MrsQG might not be the best one in some cases.

On the preprocessing side, the types of questions that can be asked depend on the named entity recognizers and ontologies. If the named entity recognizer fails to recognize a term, MrsQG is only able to generate a `yes/no` question that requires no term extraction. Even worse, if the named entity recognizer mistakenly recognizes a term, MrsQG generates wrong questions that might be confusing to people.

On the side of sentence simplification, the decomposed sentences are not necessarily grammatical. Even grammatical sentences might not make sense due to lack of information. For instance, given a restrictive relative clause, "`Bart is the cat that chases the dog`", one does not ask "`who is the cat?`" but rather "`Who is the cat that chases the dog?`". In this case, sentence simplification produces questions that are too vague and that cannot be understood without a context. Our proposed system can only rely on the question ranking module to select a best one. In some other cases, such as the hypothetical clause "`if Bart chases dogs, Bart is a brave cat`", the subordinate decomposer might extract a false statement "`Bart chases dogs`" or "`Bart is a brave cat`", then a question that cannot find its answer from the input would be generated. Ruling out these types of questions require either some extra rules, or a textual entailment module to judge whether the extracted simple clause can be inferred from the original sentence.

From the theoretical point of view, the theory underlying MrsQG is Dependency Minimal Recursion Semantics, a variant of MRS. DMRS provides a connecting interface between a semantic language and a dependency structure. Although still under development, it has been successfully applied to question generation via MrsQG . However, there are still redundancies in DMRS (such as the non-directional NULL/EQ "dependence"). Some of the redundancies are even crucial: a mis-manipulation of the MRS structure inevitably leads to generation failure and thus makes the MRS transformation process fragile. Fixing this issue is not trivial however, which requires the joint effort from the MRS theory, the ERG grammar and the LKB generator.

From the application point of view, MrsQG limits itself to only generating from single sentences. Expanding the input range to paragraphs or even articles is more interesting for applications but needs more sophisticated processing. Thus MrsQG currently only serves as a starting point for the full task of question generation.

7. Conclusion and Future Work

This paper introduces the task of question generation and proposes a semantics-based method to perform this task. Generating questions from the semantics of languages is intuitive but also has its difficulties, namely sentence simplification, question transformation and question ranking. This paper proposes three methods to address these issues: MRS decomposition for complex sentences to simplify sentences, MRS transformation for simple sentences to convert the semantic form of declarative sentences into that of interrogative sentences, and hybrid ranking to select the best questions. The underlying theoretical support comes from a dependency semantic representation (DMRS) while the backbone is an English deep precision grammar (ERG) based on the HPSG framework. The core technology used in this paper is MRS decomposition and transfer. A generic decomposition algorithm is developed to perform sentence simplification, which boils down to solving a graph traversal problem following the labels of edges that encode linguistic properties.

Evaluation results reveal some of the fine points of this semantics-based method and also challenges that indicate future work. The proposed method works better than most other syntax/rule-based systems in terms of the correctness and variety of generated questions, mainly benefiting from the underlying precision grammar. However, the quality of yes/no questions is not the best among other question types. This indicates that the sentence decomposer does not work very well. The main reason is that it does not take text cohesion and context into account. Thus sometimes the simplified sentences are ambiguous or even not related to the original sentences. Enhancing the sentence decomposer to simplify complex sentences but still preserving enough information and semantic integrity is one of the future works.

The low inter-rater agreement shows that the evaluation criteria are not well defined. Also, the evaluation focuses on standalone questions without putting the task of question generation into an application scenario. This disconnects question generation from the requirements of actual applications. For instance, an intelligent tutoring system might prefer precise questions (achieving high precision by sacrificing recall) whilst a closed-domain question answering system might need as many questions as possible (achieving high recall by sacrificing precision). Since the research of question generation has just started, efforts and results are still in a preliminary stage. Combining question generation with specific application requirements has been put into the long-term schedule.

To sum up, the method proposed by this paper produces so far the first open-source semantics-based question generation system. It properly employs a series of deep processing steps which in turn lead to better results than most other shallow methods. Theoretically, it describes the linguistic structure of a sentence as a dependency semantic graph and performs sentence simplification algorithmically, which has its potential usage in other fields of natural language processing. Practically, the developed system is open-source and provides an automatic application framework that combines various tools for preprocessing, parsing, generation and MRS manipulation. The usage of this method will be further tested in specific application scenarios, such as intelligent tutoring systems and closed-domain question answering systems.

Acknowledgments

This article is based on the first author's Master thesis (Yao, 2010).

References

Peter Adolphs, Stephan Oepen, Ulrich Callmeier, Berthold Crysmann, Dan Flickinger, and Bernd Kiefer. Some Fine Points of Hybrid Natural Language Parsing. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, May 2008. European Language Resources Association (ELRA).

Husam Ali, Yllias Chali, and Sadid A. Hasan. Automation of Question Generation From Sentences. In Kristy Elizabeth Boyer and Paul Piwek, editors, *Proceedings of the Third Workshop on Question Generation*, Pittsburgh, Pennsylvania, USA, 2010.

S. Bangalore and A.K. Joshi. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–265, 1999.

M. Braschler and C. Peters. Cross-language evaluation forum: Objectives, results, achievements. *Information Retrieval*, 7(1):7–31, 2004.

Ulrich Callmeier. PET – a platform for experimentation with efficient HPSG processing techniques. *Nat. Lang. Eng.*, 6(1):99–107, 2000. ISSN 1351-3249.

J. Carroll and S. Oepen. High efficiency realization for a wide-coverage unification grammar. *Lecture notes in computer science*, 3651:165, 2005.

John Carroll, Ann Copestake, Dan Flickinger, and Victor Poznanski. An efficient chart generator for (semi-)lexicalist grammars. In *Proceedings of the 7th European Workshop on Natural Language Generation (EWNLG'99)*, pages 86–95, Toulouse, France, 1999.

R. Chandrasekar, C. Doran, and B. Srinivas. Motivations and methods for text simplification. In *Proceedings of the 16th conference on Computational linguistics*, volume 2, pages 1041–1044, 1996.

Wei Chen, Gregory Aist, and Jack Mostow. Generating Questions Automatically from Informational Text. In *Proceedings of the 2nd Workshop on Question Generation In Craig, S.D. & Dicheva, S. (Eds.) (2009) AIED 2009: 14th International Conference on Artificial Intelligence in Education: Workshops Proceedings*, 2009.

Trevor Cohn and Mirella Lapata. Sentence Compression as Tree Transduction. *J. Artif. Intell. Res. (JAIR)*, 34:637–674, 2009.

A. Copestake, D. Flickinger, C. Pollard, and I.A. Sag. Minimal Recursion Semantics: An Introduction. *Research on Language & Computation*, 3(4):281–332, 2005.

Ann Copestake. *Implementing Typed Feature Structure Grammars.* CSLI: Stanford, 2002.

Ann Copestake. Dependency and (R)MRS. `http://www.cl.cam.ac.uk/~aac10/papers/dmrs.pdf`, 2008.

B. Dorr, D. Zajic, and R. Schwartz. Hedge trimmer: A parse-and-trim approach to headline generation. In *Proceedings of the HLT-NAACL 03 on Text summarization workshop*, volume 5, pages 1–8, 2003.

Jason Eisner. Learning non-isomorphic tree mappings for machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 2*, ACL '03, pages 205–208, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

Marcello Federico and Mauro Cettolo. Efficient handling of N-gram language models for statistical machine translation. In *StatMT '07: Proceedings of the Second Workshop on Statistical Machine Translation*, pages 88–95, Morristown, NJ, USA, 2007.

Christiane Fellbaum, editor. *WordNet: An electronic lexical database*. MIT press Cambridge, MA, 1998.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by Gibbs sampling. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370, Morristown, NJ, USA, 2005.

Dan Flickinger. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6(1):15–28, 2000. ISSN 1351-3249.

A. Graesser, J. Otero, A. Corbett, D. Flickinger, A. Joshi, and L. Vanderwende. Guidelines for Question Generation Shared Task and Evaluation Campaigns. In V. Rus and A. Graesser, editors, *The Question Generation Shared Task and Evaluation Challenge Workshop Report*. The University of Memphis, 2009.

M. Heilman and N. A. Smith. Question Generation via Overgenerating Transformations and Ranking. Technical report, Language Technologies Institute, Carnegie Mellon University Technical Report CMU-LTI-09-013, 2009.

M. Heilman and N. A. Smith. Good Question! Statistical Ranking for Question Generation. In *Proc. of NAACL/HLT*, 2010a.

Michael Heilman and Noah A. Smith. Extracting Simplified Statements for Factual Question Generation. In *Proceedings of the 3rd Workshop on Question Generation.*, 2010b.

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. Toward Semantics-Based Answer Pinpointing. In *HLT '01: Proceedings of the first international conference on Human language technology research*, pages 1–7, Morristown, NJ, USA, 2001.

Martin Kay. Chart generation. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 200–204, Morristown, NJ, USA, 1996.

J. R. Landis and G. G. Koch. The Measurement of Observer Agreement for Categorical Data. *Biometrics*, 33(1):159–174, March 1977.

Xin Li and Dan Roth. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics*, pages 1–7, Morristown, NJ, USA, 2002.

Yandong Liu and Eugene Agichtein. You've Got Answers: Towards Personalized Models for Predicting Success in Community Question Answering. In *Proceedings of Annual Meeting of the Association of Computational Linguistics (ACL)*, 2008.

Jack Mostow and Wei Chen. Generating Instruction Automatically for the Reading Strategy of Self-Questioning. In *Proceeding of the 2009 conference on Artificial Intelligence in Education*, pages 465–472, Amsterdam, The Netherlands, 2009. IOS Press.

Stephan Oepen, Helge Dyvik, Jan Tore Lønning, Erik Velldal, Dorothee Beermann, John Carroll, Dan Flickinger, Lars Hellan, Janne Bondi Johannessen, Paul Meurer, Torbjørn Nordgård, and Victoria Rosén. Som å kapp-ete med trollet? Towards MRS-Based Norwegian-English Machine Translation. In *Proceedings of the 10th International Conference on Theoretical and Methodological Issues in Machine Translation*, Baltimore, MD, October 2004.

Santanu Pal, Tapabrata Mondal, Partha Pakray, Dipankar Das, and Sivaji Bandyopadhyay. QG-STEC System Description – JUQGG: A Rule based approach. In Kristy Elizabeth Boyer and Paul Piwek, editors, *Proceedings of the Third Workshop on Question Generation*, Pittsburgh, Pennsylvania, USA, 2010.

C.J. Pollard and I.A. Sag. *Head-Driven Phrase Structure Grammar.* University of Chicago Press, 1994.

V. Rus, B. Wyse, P. Piwek, M. Lintean, Stoyanchev S., and C. Moldovan. The First Question Generation Shared Task Evaluation Challenge. In *Proceedings of the 6th International Natural Language Generation Conference (INLG 2010)*, Dublin, Ireland, 2010.

Ivan A. Sag and Dan Flickinger. Generating Questions with Deep Reversible Grammars. In *Proceedings of the First Workshop on the Question Generation Shared Task and Evaluation Challenge*, Arlington, VA: NSF, 2008.

Lee Schwartz, Takako Aikawa, and Michel Pahud. Dynamic Language Learning Tools. In *Proceedings of the 2004 InSTIL/ICALL Symposium*, 2004.

Andrea Varga and Le An Ha. WLV: A Question Generation System for the QGSTEC 2010 Task B. In Kristy Elizabeth Boyer and Paul Piwek, editors, *Proceedings of the Third Workshop on Question Generation*, Pittsburgh, Pennsylvania, USA, 2010.

E. Velldal and S. Oepen. Statistical ranking in tactical generation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 517–525, 2006.

Ellen M. Voorhees. The TREC question answering track. *Nat. Lang. Eng.*, 7(4):361–378, 2001.

Brendan Wyse and Paul Piwek. Generating Questions from OpenLearn study units. In *Proceedings of the 2nd Workshop on Question Generation In Craig, S.D. & Dicheva, S. (Eds.) (2009) AIED 2009: 14th International Conference on Artificial Intelligence in Education: Workshops Proceedings*, 2009.

Xuchen Yao. Question Generation with Minimal Recursion Semantics. Master's thesis, Saarland University & University of Groningen, 2010.

Yi Zhang, Valia Kordoni, and Erin Fitzgerald. Partial Parse Selection for Robust Deep Processing. In *Proceedings of ACL 2007 Workshop on Deep Linguistic Processing*, pages 128–135, 2007.