Master Thesis

Question Generation With Minimal Recursion Semantics

Xuchen Yao

August, 2010

European Masters Program
in Language and Communication Technologies
University of Groningen & Saarland University

Supervisors: Prof. Hans Uszkoreit and Dr. Yi Zhang
Saarland University
Co-supervisor: Dr. Gosse Bouma
University of Groningen

Author's Declaration

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Declaration

I hereby confirm that the thesis presented here is my own work, with all assistance acknowledged.

Signature: Xuchen Yao

Date

Abstract

Question Generation (QG) is the task of generating reasonable questions from a text. It is a relatively new research topic and has its potential usage in intelligent tutoring systems and closed-domain question answering systems. Current approaches include template or syntax based methods. This thesis proposes a novel approach based entirely on semantics.

Minimal Recursion Semantics (MRS) is a meta-level semantic representation with emphasis on scope underspecification. With the English Resource Grammar and various tools from the DELPH-IN community, a natural language sentence can be interpreted as an MRS structure by parsing, and an MRS structure can be realized as a natural language sentence through generation.

There are three issues emerging from semantics-based QG: (1) sentence simplification for complex sentences, (2) question transformation for declarative sentences, and (3) generation ranking. Three solutions are also proposed: (1) MRS decomposition through a Connected Dependency MRS Graph, (2) MRS transformation from declarative sentences to interrogative sentences, and (3) question ranking by simple language models atop a MaxEnt-based model.

The evaluation is conducted in the context of the Question Generation Shared Task and Generation Challenge 2010. The performance of proposed method is compared against other syntax and rule based systems. The result also reveals the challenges of current research on question generation and indicates direction for future work.

Acknowledgement

I realized that this is the only place I can choose not to be serious in this thesis so decided to write something not that serious here. I hope people who get acknowledged do not get serious about what I've written in this page. I'm indebted to:

Dr. Yi Zhang, my direct supervisor, who helped me finally set down the thesis topic, held a help hotline for me all the time and always gave me enough freedom to stretch myself.

The European Masters Program in Language and Communication Technologies (LCT), its generous scholarship and the program's coordinators PD Dr. Valia Kordoni and Professor Hans Uszkoreit, who opened a fantastic window of computational linguistics to a former electronic engineer and most importantly, made me like it.

Dr. Gosse Bouma, my co-supervisor, whose humorous emails I enjoyed every time and who gave me even more freedom to stretch even more.

Professor Gisela Redeker, my former local coordinator at Groningen, who is always so patient, encouraging and thoughtful that every student just loves her.

Dr. Dan Flickinger, the main author of the English Resource Grammar, who wrote me incisive emails which I had to chew on for quite a while every time and who always encouraged me to develop my new ideas no matter how naïve they were.

Dr. Ann Copestake, Professor Stephan Oepen and other members from the DELPH-IN community, who were always there to explain things to me quickly and precisely.

Irina Borisova, who kept asking me "how many pages do you have so far?" and successfully made me keep wondering "how many pages do I have so far?" when writing this thesis (in order to tell her exactly how many pages I had every time she asked me). This turned out to be a very good supervision during the writing.

Pranesh Bhargava, who very kindly proof-read some part of this thesis and explained to me how to use English properly. Thus the part which Pranesh corrected is significantly better than the other parts.

Till Tantau, without whose amazing Pi*k*Z package I couldn't have drawn those confusing graphs. The credit for being amazing belongs to Till while the blame for being confusing goes to me.

Clusters' clusters, without which I could not have done any of the experiments. I never ran them on all wheels and always kept their load average below 5. According to Steven Levy's hacker ethic, "computers can change your life for the better". Thus I treated them well and hope they do the same for me in return.

John and Mary. One central theme in this thesis. Although they favor different animals, in my story they do not fight each other and they have a happy ending.

Bart, a very brave cat that chases dogs. The other central theme in this thesis. You can find a lot of these animals in the AFV.

To My Parents

Contents

List of Figures

List of Tables

1. Introduction

1.1. Question Generation: Definition and Status Quo

Question Generation (QG) is the task of generating reasonable questions from a text. It is a joint effort between Natural Language Understanding (NLU) and Natural Language Generation (NLG). Simply speaking, 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 of symbols for declarative sentences to symbols for interrogative sentences, as shown in Figure 1.1. Here we denote symbols as an organized data form that can represent natural languages and that can be processed by a machinery, artificial or otherwise.

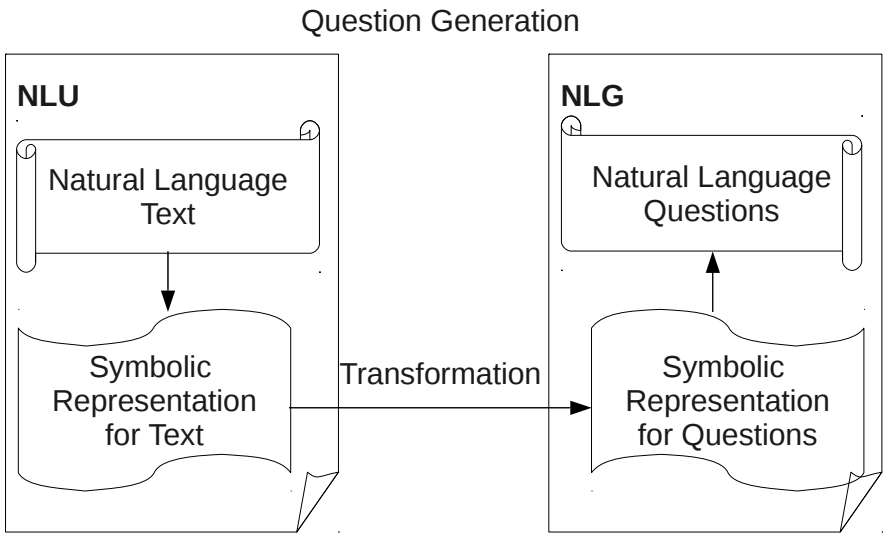


Figure 1.1.: The relation between Question Generation and its two components: Natural Language Understanding (NLU) and Natural Language Generation (NLG).

The task of question generation contains multiple subareas. Usually the purposes of QG determine the specific task definition and the best approaches. Generally speaking, a QG system can be helpful to the following areas:

- Intelligent tutoring systems. QG can ask learners questions based on learning materials in order to check their 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.
- Closed-domain Question Answering (QA) systems. Some closed-domain QA systems use pre-defined (sometimes hand-written) question-answer pairs to provide QA services. By employing a QG approach such systems could expand to other domains with a small 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.

According to the purpose of the usage, questions can be classified into different categories. For instance, in the question answering track of the text retrieval conference (Voorhees 2001), questions fall under three types: factoid, list and other. Factoid questions (such as "`How tall is the Eiffel Tower?`") ask for fact-based answers and list questions (such as "`What are army rankings from top to bottom?`") ask for a set of answer terms. In terms of target complexity, the type of QG can be divided into *deep* QG and *shallow* QG (Rus and Graesser 2009). Deep QG generates deep questions that involves 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).

The QG task has not aroused much attention until very recently[1]. The research on QG is also still in a preliminary stage, including methodology exploration, evaluation criteria selection and dataset preparation, etc. Concrete plans have been proposed (Rus and Graesser 2009) to push this area of research forward, which follows a road map from generating shallow questions to deep questions, from direct questions on explicit information (such as generating "`Who likes Mary?`" from "`John likes Mary.`") to inferred questions on implicit information (such as generating "`How much tip did John give?`" from "`John went to a restaurant.`"), from using single sentences as sources to using paragraphs or even multiple texts. Being in the early stage of QG research, this thesis concentrates on generating questions that can be answered by given texts, specifically, a text that contains only one sentence, rather than a paragraph, which would also need to take discourse cohesion into account.

The Question Generation Shared Task and Evaluation Challenge (QGSTEC) 2010[2] is one of the efforts the QG community has made to bring together both mind and will to approach this task in a form of challenge and competition. This challenge somehow has practically provided a way to tackle two very important issues researchers face when they enter a new area of Natural Language Processing (NLP): evaluation method and dataset allocation. As will be introduced in details later, as a challenge, QGSTEC2010 unifies the evaluation criteria, offers development set and test set, and finally organizes the

[1] `http://www.questiongeneration.org/`
[2] `http://www.questiongeneration.org/QGSTEC2010/`

overall evaluation. This greatly saves time for researchers from arguing about evaluation criteria and preparing datasets and therefore helps them to focus on methodologies and algorithms. Along with the evaluation results, a comparison of different approaches and discussion are also presented in this thesis.

1.2. Proposed Method: Motivation and Research Objectives

Thinking of generating a few simple questions from the sentence "`John plays football.`":

(1.1) `John plays football.`

(a) `Who plays football?`

(b) `What does John play?`

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

(1.2) play(John, football) $\Leftarrow$ `John plays football.`

(a) play(who, football) $\Leftarrow$ `Who plays football?`

(b) play(John, what) $\Leftarrow$ `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.

The author proposes a semantics-based method of transforming the Minimal Recursion Semantics (MRS, Copestake et al. 2005) representation of declarative sentences to that of interrogative sentences to perform question generation. The MRS analysis is obtained from PET (Callmeier 2000) while the generation function comes from the Linguistic Knowledge Builder (LKB, Copestake 2002). The underlying core component is the English Resource Grammar (ERG, Flickinger 2000). To help readers understand and differentiate the functions of the above components, an example is given in Figure 1.2.

The advantage of this approach has its basis in the semantic representation of meaning. Operations that are done at the semantics level makes it more language independent and void of syntactic heaviness in linguistic realization. But this does not mean it lacks syntactic expressive power during generation. On the contrary, as the semantic representation is more complex, there are even more syntactic interpretations that produce too many surface realizations, which raises another research question of ranking the generated questions.

Lexical semantics resources, such as ontologies, semantic networks, can be incorporated in this approach. For instance, given that "sport" is a hypernym of "football", we can have the following transformation:

(1.3) play(John, football) & hypernym(sport, football) $\Rightarrow$ play(John, which sport)

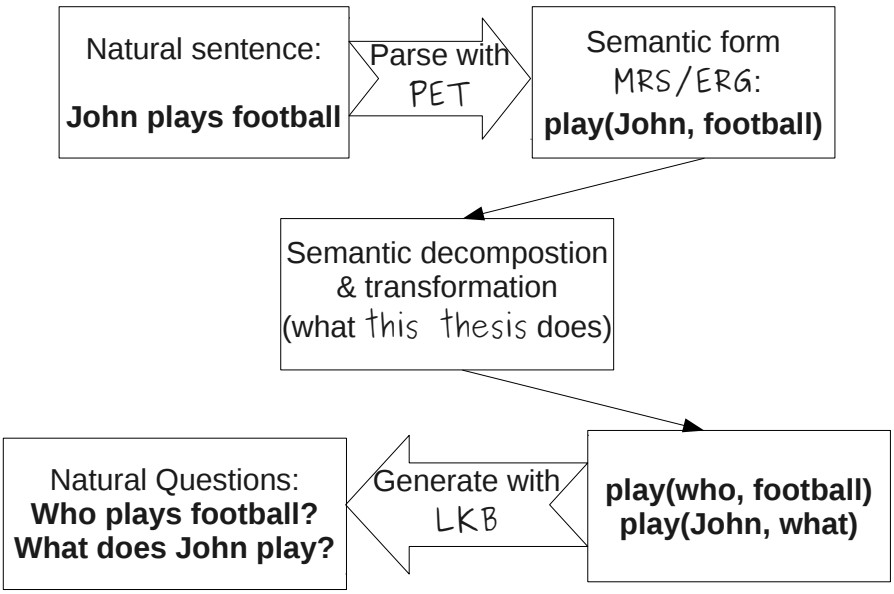


Figure 1.2.: A simple illustration of the functionalities of PET/MRS/ERG/LKB, along with the focus of this thesis in the task of question generations with semantics. Note the upper part of this graph corresponds to the NLU module in Figure 1.1, while the lower part corresponds to the NLG module.

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 work could potentially benefit the following areas of research:

1. Exploration of semantics-based approaches to QG, which, to the author's knowledge, has never been specified in detail before.

2. As a subtask of semantics-based QG, the main efforts are put into developing algorithms of sentence simplification by decomposing complex semantic representations into small and individual ones, which can be reused by other NLP tasks.

3. Comparison of syntax-based and semantics-based approaches to QG. Most of the systems that have emerged from the QGSTEC2010 challenge employ a syntax-based approach. Since QGSTEC2010 has an open and unified evaluation standard, a detailed comparison of two distinct approaches can be done against different evaluation criteria.

1.3. Thesis Overview

Chapter 2 reviews the related work in question generation and explains why a semantics-based work is proposed given the related work. It also raises major difficulties in the task of question generation whilst Chapter 4 provides corresponding solutions. Specifically, Section 4.2 and 4.3 present the key idea in this thesis. For readers who are not familiar with various tools from the DELPH-IN community and the (dependency) MRS theory, Chapter 3 is an essential before moving on to Chapter 4. Evaluation results are presented in Chapter 5 and various aspects of the proposed method are discussed in Chapter 6. Finally, Chapter 7 concludes this thesis and addresses future work.

2. Related Work

2.1. Overview of Research Difficulties

At the crossroad of natural language understanding and natural language generation, question generation employs technologies from both NLU and NLG. At the front end, NLU parses a text into a semantic representation (Allen 1995), which can be done with a semantic parser along with a lexicon and a grammar. At the back end, the process of NLG is divided into several steps by Reiter and Dale (1997): *content determination, discourse planning, sentence aggregation, lexical choice, referring expression generation*, and *linguistic realization*. In a narrow scope of question generation from single sentences, taking the sentence "`John told Peter he liked Mary.`" as an example, a QG system can apply the following steps:

1. Content Determination. The focus words that can be asked questions about are determined, i.e. "`John`", "`Peter`", "`Mary`" and the relative clause "`he liked Mary`" serving as the direct object of "`told`".

2. Lexical Choice. The question words are chosen according the determined content, i.e. "`who`" or "`which person`" and "`what`".

3. (Optional) Referring Expression Generation. In complex sentences anaphora resolution might be needed to identify the correct coreferences, i.e. "`he`" refers to "`John`".

4. Linguistic Realization. The final questions are produced.

The steps for *discourse planning* and *sentence aggregation* are not present because this thesis only focuses on question generation from simple sentences, which does not concern cohesion from a discourse level. However, these steps must be considered if the input text exceeds a single sentence.

Because the research of question generation has just started, a big amount of literature discusses question taxonomy and evaluation criteria. This chapter, however, focuses on existing approaches to building a working question generation system.

Generally speaking, there are three major problems in question generation:

1. Question transformation. As shown in Figure 1.1, there must be a theoretically-sound and practically-feasible algorithm to build a mapping from symbolic representation of declarative sentences to interrogative sentences.

2. Sentence simplification. A common sense in both parsing and generation is that generally speaking the easier the input to a parser/generator, the better the quality

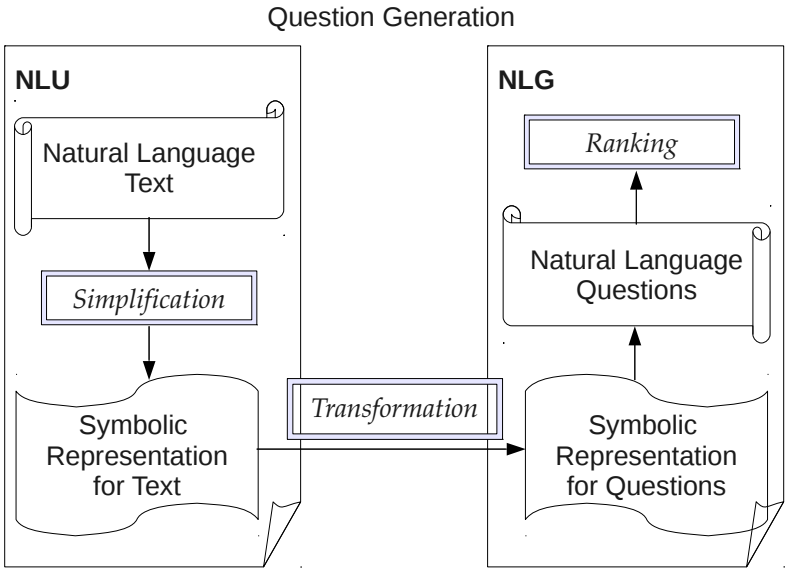


Figure 2.1.: Three major problems (*sentence simplification, question transformation* and *question ranking*) in the process of question generation, as shown in the framed boxes.

of the output. Complex sentences must be first broken into simple ones so the parser and NLU unit can better handle the input.

3. Question ranking. In the case of over generation, a ranking algorithm to grade the grammaticality and naturalness of questions must be developed.

Figure 2.1 has shown these three problems in a whole overview of a question generation framework. The following text reviews research literatures on these problems respectively.

2.2. Question Transformation

Silveira (2008) proposes a general architecture of a question generation system. It makes a difference between the generation of shallow questions and deep questions by employing different model representations. World knowledge is incorporated to help make inferences from input. A question design module determines the type of questions to be generated

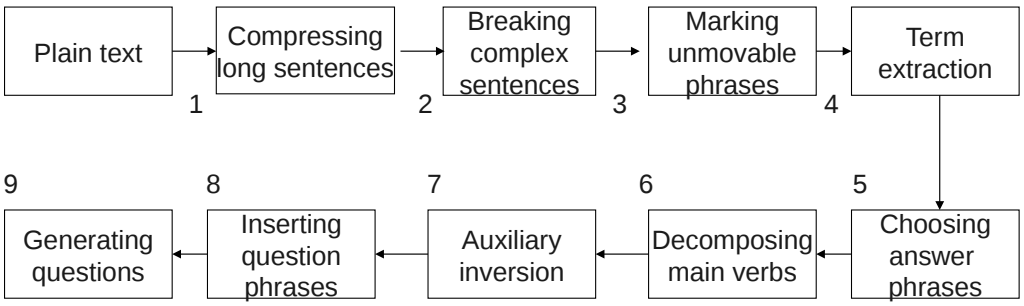


Figure 2.2.: Pipelines of a syntax transformation based question generation system.

and the final results are obtained from an NLG module. The work presents what could be done in QG but does not specify how. There are generally three approaches to do it: template-based, syntax-based and semantics-based.

2.2.1. Template-based

Mostow and Chen (2009) reported a system under a self-questioning strategy to help children generate questions from narrative fiction. Three question templates are used:

- What did <character> <verb>?
- Why/How did <character> <verb> <complement>?
- Why was/were <character> <past-participle>?

Of 769 questions evaluated, 71.3% was rated acceptable. Chen et al. (2009) expanded this system to generate questions from informational text. 4 more templates were added for What-would-happen-if, When-would-x-happen, What-would-happen-when and Why-x questions. 180 questions were generated from 444 sentences and 66% ~ 87% of them were rated acceptable, depending on the question types.

Template-based approaches are mostly suitable for applications with a special purpose, which sometimes comes within a closed-domain. The tradeoff between coverage and cost is hard to balance because human labors must be involved to produce high-quality templates. Thus, it has lost researchers' favor in open-domain general-purpose applications.

2.2.2. Syntax-based

Wyse and Piwek (2009) and Heilman and Smith (2009) use a very similar approach by manipulating syntactic trees from the Stanford parser (Klein and Manning 2003). The core idea is to transduce a syntactic tree of a declarative sentence into that of an interrogative.

As shown in Figure 2.2, the system consists of a pipeline of 11 steps. The following is not a complete description of the algorithm but is intended for a clear illustration with examples. Some steps are accompanied by some tree-based pattern matching rules. These rules follow a style of Tgrep2[1]. For instance, S < (NP .. VP) matches any S which immediately dominates an NP (which procedes a VP). Tree-based pattern matching and operation are done through the Stanford Tregex and Tsurgeon utility (Levy and Andrew 2006).

1. Compressing long sentences. Extra sentence-initial conjunctions, adjunct phrases are removed. For instance, in the second sentence of "`John plays basketball. And Bill plays soccer.`", the conjunction word "and" is matched by the rule ROOT < (S < CC=conj) and removed.

2. Breaking complex sentences. Simpler sentences are extracted from complex ones. For instance, the complex sentence "`John is tall and plays basketball.`" is divided into two simpler ones: "`John is tall. John plays basketball.`" by matching the rule VP < (CC=conj „ VP=vp1 .. VP=vp2) > (S > ROOT), in which vp1 and vp2 stays at the left and right side of the conjunction conj, and then using vp1 and vp2 to assemble two new sentences.

3. Marking unmovable phrases. Some phrases cannot be moved inside sentences. For instance, in order to prevent generating "`*John met who and Mary?`" from "`John met Bob and Mary.`", the whole phrase "`Bob and Mary`" is marked as unmovable by matching the pattern /\\.*/ < CC << NP=unmovable.

4. Term extraction. Terms are extracted as answer candidates. They are usually named entities.

5. Choosing answer phrases. Answer phrases are NPs and PPs tagged as Named Entities (NE). For instance, in "`Jackson was born on August 29, 1958 in Gary, Indiana.`", answer phrases are "Jackson" (NEperson), "on August 29, 1958" (NEdate) and "in Gary, Indiana" (NElocation).

6. Decomposing main verbs. The main verb of a sentence is decomposed to a form of auxiliary+lemma. For instance, the sentence "`John plays basketball.`" is transformed into "`John does play basketball.`"

7. Auxiliary inversion. The main auxiliary verb is inverted, if necessary. Following 6, "`John does play basketball.`" is transformed into "`Does John play basketball.`".

[1] `http://tedlab.mit.edu/~dr/Tgrep2/`

8. Inserting question phrases. If the answer phrase is the subject, then replace the subject with the answer phrase. Following 5, it generates "`who was born on August 29, 1958 in Gary, Indiana.`". If the answer phrase is non-subject, then insert the answer phrase to the front. Following 5, it generates "`when was Jackson born in Gary, Indiana.`"

9. Generating questions. The final questions are generated by adding a question mark to the end and capitalizing the first letter, etc.

This algorithm is reported to achieve 43.3% acceptability for the top 10 ranked questions and to produce an average of 6.8 acceptable questions per 250 words on Wikipedia texts (Heilman and Smith (2009)). The whole generation is based on tree nodes matching and operation. All operations are straight-forward from a syntactic point of view.

2.2.3. Semantics-based

Schwartz et al. (2004) introduces a content question generator based on a system called NLPwin (Heidorn 2000) used by Microsoft. It uses the logical form to represent the semantic relationships of the arguments within a sentence and generate WH-questions. However, the paper just introduces the result of this generator but the inner mechanism is not presented.

Sag and Flickinger (2008) discusses the possibility and feasibility to use 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. Only concrete ideas and implementation are lacking.

2.3. Sentence Simplification

Sentence simplification reduces the average length of a sentence and syntactic complexity, while the latter 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. In one method, chunk parsing is used while in the other one supertagging (Bangalore and Joshi 1999) from Lexicalized Tree Adjoining Grammar (LTAG, Schabes (1992)) is used to give a dependence analysis. An example rule taken from method one of the original paper is as follows:

(2.1) `X:NP, RelPron Y, Z` $\longrightarrow$ `X:NP Z. X:NP Y.`

(a) $(\texttt{The cat})_{\texttt{X:NP}}$, $(\texttt{who})_{\texttt{RelPron}}$ $(\texttt{chased the dog})_{\texttt{Y}}$, $(\texttt{was brave})_{\texttt{Z}}$. $\longrightarrow$

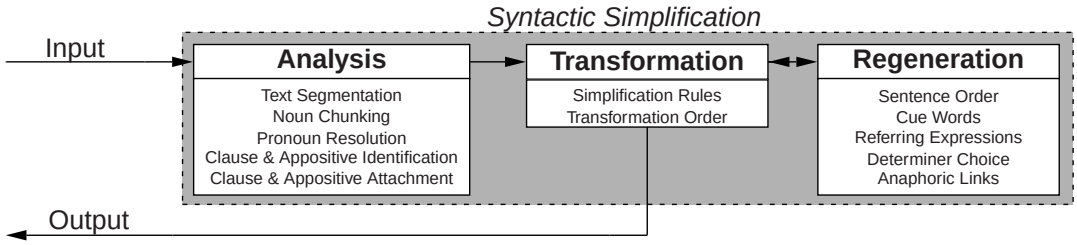


Figure 2.3.: A three-stage architecture for syntactic simplification from Siddharthan (2003).

(b) (`The cat`)$_{\texttt{X:NP}}$ (`was brave`)$_{\texttt{Z}}$.

(c) (`The cat`)$_{\texttt{X:NP}}$(`chased the dog`)$_{\texttt{Y}}$.

Basically it says that in any non-restrictive relative clauses, if a noun phrase (`X:NP`) is followed by a pronoun (`RelPron`) and any word sequences of `Y`, comma and `Z`, then the sentence can be broken into two simpler ones: `X Z` and `X Y`.

Method two uses similar rules in an expressive fashion of tree adjoining grammars. It does not only use parts-of-speech information but also the non-flat dependence relations. Thus, it performs better than method one as reported in Chandrasekar et al. (1996).

The two stage simplification process is rephrased as *analysis* and *transformation* in Chandrasekar and Srinivas (1997). Preserving these two stages, Siddharthan (2003) added a third *regeneration* stage to address the issue of text cohesion. The main focus is put on anaphoric cohesive-relations. Broken anaphoric references are re-generated during simplification.

The three-stage architecture is shown in Figure 2.3. In natural language processing, it is 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.

2.4. 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. The following is an example.

(2.2) Input: Semantic representation of the sentence "`What does John say to Mary about the wedding?`"

Generations:

(a) `About the wedding what does John say to Mary?`

(b) `About the wedding, what does John say to Mary?`

(c) `To Mary, what does John say about the wedding?`

(d) `To Mary what does John say about the wedding?`

(e) `What does John say about the wedding to Mary?`

(f) `What does John say to Mary about the wedding?`

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 uses features defined over derivation trees as well as lexical trigram models. Different performance measures were conducted but here we only show the results on *exact match accuracy* and 5-best accuracy. Exact match accuracy is calculated from the portion of sentences which are assigned the highest score and which are in fact also the best ones according to a gold standard. 5-best accuracy measures how much of the top-5 scored result contains a gold standard sentence. Table 2.1 shows this comparison. Performance was tested on two datasets: "Jotunheimen" is based on high-quality tourism guide books; "Rondane" contains text gathered from a variety of web sites, with a purpose of cross-domain evaluation. "Jotunheimen" contains 2190 sentences in total and "Rondane" contains 634. Both of them have an average sentence length of 15.1 words.

The result reported by Velldal and Oepen (2006) is mostly on declarative sentences. 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 containing sample articles from Wikipedia, Wikipedia in simple English and Section 23 of the Wall Street Journal in the Penn Tree Bank. 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% of ranked questions were acceptable.

2.5. Perception of Related Work

Among the three subtasks of question generation, most of the work related to sentence simplification and question transformation boils down to be syntax-based. The internal

	Jotunheimen		Rondane	
Model	**accuracy**	**5-best**	**accuracy**	**5-best**
BNC LM	53.24	78.81	54.19	77.19
SVM	71.11	84.69	63.64	83.12
MaxEnt	72.28	84.59	64.28	83.60

Table 2.1.: Exact match accuracy and 5-best scores for the different models from Velldal and Oepen (2005). The results on "Jotunheimen" for SVM and MaxEnt are averaged from 10-fold cross-validation. A model trained on the entire "Jotunheimen" was tested on "Rondane".

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 not (with parse trees). With syntax-based method fully exploited, it is theoretically interesting and wanting to search for methods that can tackle these issues in a semantics-based fashion. This semantics-based method involves 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.

Machine learning methods are rarely used in the research of sentence simplification and question transformation. Chandrasekar and Srinivas (1997) is an exception, which tries to learn simplification rules from a parallel corpus containing both complex sentences and corresponding handcrafted simple ones. The authors claimed that it inherited the merit of good generalization from a machine learning point of view, as long as there is such a parallel corpus. But more than a decade has passed and there has not been trace of interest in creating such a big corpus that facilitates machine learning. Siddharthan (2003) argued that the time consumed to manually write rules is much less than the time to create a corpus. It is also unlikely that a system which learns from a simplified corpus outperforms a system where the rules themselves have been manually written. As currently there is no such a parallel corpus publicly available, it is natural that the proposed semantics-based method also stays rule-based.

However, the subtask of question ranking always takes a statistical approach. It is well studied (e.g. Walker et al. 2001; Langkilde and Knight 1998; Nakanishi et al. 2005) and shares similarities with parse selection (Collins and Koo 2005; Noord and Malouf 2004). Thus, this thesis is not intended for novel approaches on question ranking but focuses on sentence simplification and question transformation.

3. Background: Theory, Grammar and Tools

The proposed system consists of different components such as ERG/LKB/PET while the theoretical support comes from the MRS layer of HPSG. Thus this chapter introduces all the involved components for a better understanding of later chapters by focusing on the parts that would be practically used.

Figure 3.1 gives an overview of the functionalities of different components. In the scope of this thesis, Minimal Recursion Semantics is the theory that provides a semantic representation of natural language sentences. PET is used as merely a parsing tool to interpret a natural language sentence in MRS. LKB in turn takes in an MRS structure and produces a natural sentence. Both PET and LKB have to access the English Recourse Grammar, which encodes lexicon and grammar rules conforming to MRS.

As a comparison with the traditional Context Free Grammars (CFG), a parser of CFG would produce a syntactic tree rather than a semantic structure. Also, conventional CFG formalisms lack a generation function, making them only a "listener" but not a "speaker".

All the tools mentioned in this chapter along with more information can be found from the DELPH-IN[1] community.

3.1. Minimal Recursion Semantics

Minimal Recursion Semantics is a meta-level language for describing semantic structures in some underlying object language (Copestake et al. 2005). In a typed feature structure, an MRS is a type mrs with four features: LTOP, INDEX, RELS, HCONS[2], as shown in Figure 3.2.

LTOP is the topmost node of this mrs. INDEX usually starts with an "e" indicating this mrs represents an event and the main predicate (_like_v_rel in this case) carries it by its ARG0 feature, i.e. its *bound variable*. RELS is a set 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)*.

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

[2] This definition here is confined with ERG and is different from Copestake et al. (2005), which uses a HOOK feature to include GTOP/LTOP and INDEX. In ERG, GTOP is omitted.

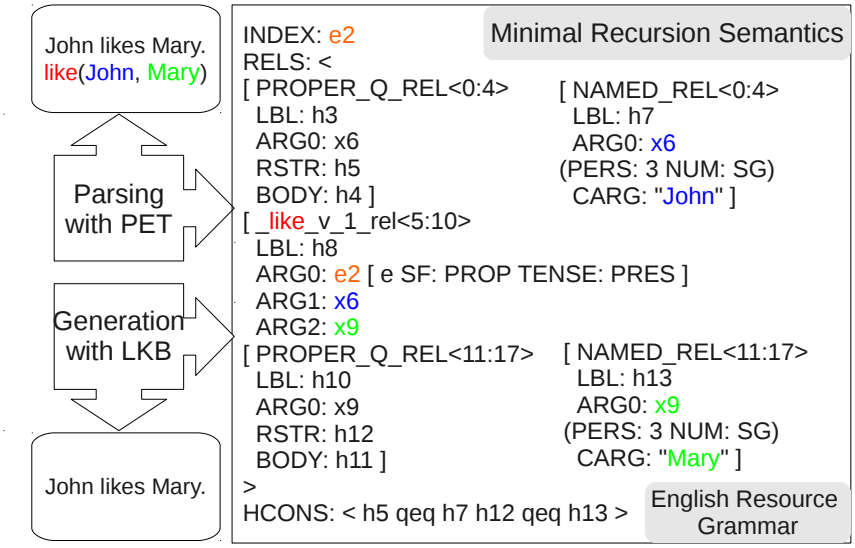
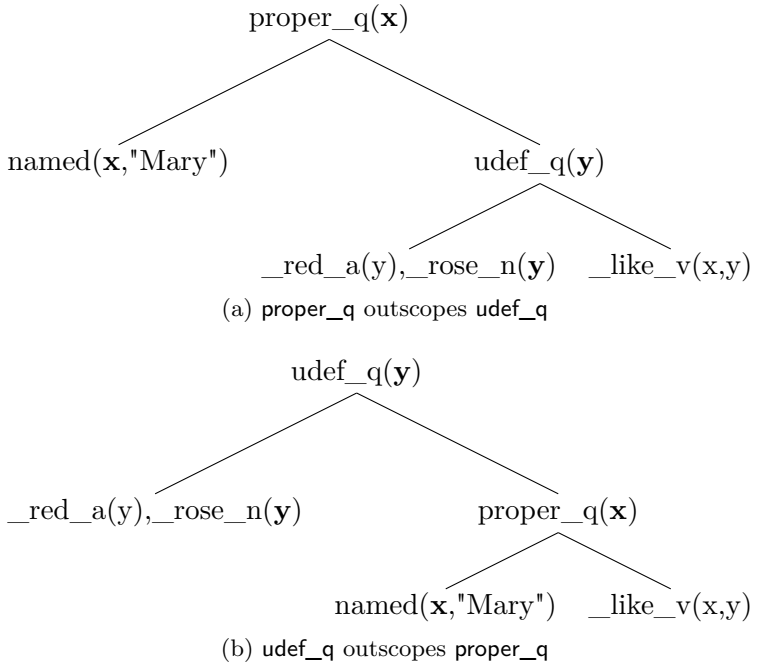


Figure 3.1.: Connecting the dots of different components (PET/LKB/MRS/ERG) of the HPSG-centralized DELPH-IN community. The predicate logic form like(John, Mary) can be further represented by Minimal Recursion Semantics, a more powerful and complex semantic formalism, which is encoded in the English Resource Grammar.

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

Figure 3.2.: MRS representation of "`Mary likes red roses`" with some linguistic attributes omitted. *Type*s are in **bold**. *Feature*s (or *attribute*s) are in CAPITAL. *Value*s are in □. Values in ⟨⟩ are *list value*s. Note that some features are token-identical. The suffix **_rel** in relation names are dropped to save space.



(a) proper_q outscopes udef_q

(b) udef_q outscopes proper_q

Figure 3.3.: Two scopal readings of the mrs from Figure 3.2. Higher tree nodes outscope lower tree node. EPs on the same node have equal scopes. The original mrs does not decide the scopal relations between proper_q and udef_q. Thus, there can be two readings. Variables in bold (e.g. **x**, **y**) are bound variables (ARG0), otherwise arguments (e.g. x, y)

Any EP with RSTR and BODY features correspond 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 3.2 the relation proper_q_rel*(x5, h4, h6)* with a "*h4* qeq *h7*" constraint means proper_q_rel *outscope*s the named_rel, which has a label *h7*.

Elementary predications with the same label have *equal scope*s. They appear on the same node in a scopal tree, such as _red_a and _rose_n in Figure 3.3. However, an mrs does not always have a unique scopal interpretation. Figure 3.3 illustrates this point.

3.1.1. Dependency Structures in MRS

MRS has two derivatives: Robust MRS (RMRS, Copestake 2004) and Dependency MRS (DMRS, Copestake 2008). RMRS serves as an interface between *deep* and *shallow* processing. It incorporates shallow processing techniques such as POS tagging and constructs

a semantic structure compatible with deeper processing. It is robust in a sense that the arguments of lexicon produced by shallow processing can be omitted or underspecified. Thus, it tolerates missing information. DMRS serves as an interface between *flat* and *non-flat* structures. Recall that in Figure 3.2, the values of RELS are a flat list of EPs. This flat structure is verbose and does not easily show how the EPs in an mrs are connected with each other. Thus, DMRS is designed to be succinct enough to show dependencies, but still preserves some of the original semantic information.

A DMRS is a connected acyclic graph. Figure 3.4 shows the DMRS of "`Mary likes red roses`.", originally from Figure 3.2. 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)
- NULL (underspecified label relationships)

An H relation is easily decided by qeq in HCONS. The difference between EQ and NEQ is that if two elementary predications share the same label, then they are in an EQ relation, otherwise an NEQ relation. For instance, _red_a and _rose_n in Figure 3.3 are on the same node, while _like_v is on a different node, thus _red_a governs _rose_n with a post-slash EQ relation while _like_v governs _rose_n with a post-slash NEQ relation.

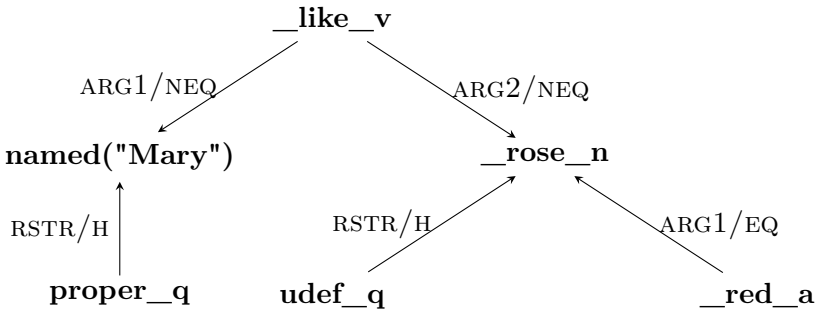


Figure 3.4.: The Dependency MRS structure of "`Mary likes red roses`". Relations between EPs are indicated by the labels of arrowed arcs. Refer back to Figure 3.2 for the original MRS.

HEQ and NULL are rare special cases and not discussed here. But examples of each can be found in Figure 4.7 and 4.12 separately.

MRS, RMRS and DMRS are all interconvertible. Note that MRS is generally more verbose. Thus converting RMRS and DMRS back to MRS needs the aid of SEM-I, the SEMantic Interface (Flickinger et al. 2005), for the lost information. Copestake (2008) provides the detailed algorithms.

3.2. English Resource Grammar

The English Resource Grammar is a general-purpose broad-coverage grammar implementation under the HPSG framework. It consists of a large set of lexical entries under a hierarchy of lexical types, with a modest set of lexical rules for production.

The ERG uses a form of *Davidsonian* representation in which all verbs introduce *event*s. This explains why the _like_v_rel relation in Figures 3.1 and 3.2 has *e2* as its ARG0. One effect of this is that adverbs are of two classes: *scopal* adverbs such as `probably`, which takes predicates as arguments and *non-scopal* adverbs such as `quickly`, which takes events as arguments[3]. The consequence is that special attention might be needed in these different cases during MRS transformation.

The ERG is in an ongoing development so structure encodings might change. One major observation from the author is that message relations described in Copestake et al. (2005) and Flickinger (2007) are missing and the purpose of sentence is encoded elsewhere. For instance, the relations prpstn_m_rel, int_m_rel and imp_m_rel were originally used to deliver a message that the sentence is declarative/exclamatory, interrogative and imperative. However, in the release of ERG (as of April 2010) the author is using, this is simplified into the SF (sentence force) attribute of the sentence event variable. Corresponding sentence forms to the previous sentence states are PROP (for proposition), QUES (for question) and COMM (for command).

3.3. Linguistic Knowledge Builder

The Linguistic Knowledge Builder is a grammar development environment for grammars in typed-feature structures and unification-based formalisms. It can examine the competence and performance of a grammar by the means of parsing and generation. As the name indicates, it is also a tool for construction of linguistic knowledge. Out of all the functionalities of LKB, the generation function is the way for linguistic realizations from the MRS representation of interrogative sentences.

3.3.1. Chart Generation in LKB

Kay (1996), Carroll et al. (1999), Carroll and Oepen (2005) describe the algorithm to use charts as in parsing charts to generate from a semantic representation. The latter two also tackle efficiency problems.

[3]Specific explanation can be found in Copestake et al. (2005) P312 and Flickinger (2007) P14.

As stated before, an MRS structure is mainly constructed by a bag of Elementary Predications, which in turn describe relations co-indexed in MRS. These relations can be traced back to a set of lexical entries and rules licenced by a grammar. Then a chart can be populated by all the retrieved lexical entries. With a bottom-up, head-first strategy, edges in a chart are instantiated and built up towards a whole sentence. The final generated output must agree with both the MRS representation and grammar rules. This double constraint, in the case of determining the order of multiple modifiers, is called *intersective modification*, which brings serious efficiency problems of both time and space.

Carroll et al. (1999), Carroll and Oepen (2005) proposed a two-phase generation algorithm. The first phase does normal chart generation but deliberately leaves out any modifiers that introduce intersective modification. In the second phase, intersective modifiers are added to the generated chart by adjunction. This two-phase algorithm brings down the worst-case complexity from exponential to polynomial.

3.4. Parsing with PET

Although ERG has a wide coverage of lexicons, there are always unknown words in real text. With LKB, parsing by symbolic processing fails whenever unknown words are present. Thus, a statistical and robust system is needed as an alternative. PET is a platform for experimentation with efficient processing of unification-based grammars (Callmeier 2000). It consists of a pre-processor called `flop` to convert handwritten grammars into a compact binary format, and a parser called `cheap` to load this binary grammar for HPSG lexicon and rules, along with some PCFG models to perform chart parsing.

`Cheap` 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 takes on and gives full parsing derivations based on some more relaxed and thus robust rules (Zhang et al. 2007a). Also, from the very beginning, efficiency has been taken into serious consideration (Callmeier 2000 and Zhang et al. 2007b). A system with PET running will not be easily overloaded even that it performs deep linguistic processing.

4. Proposed Method

Recall in Section 2.1 we addressed three major problems in question generation: question transformation, sentence simplification and question ranking. This chapter is aimed 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.

4.1. System Architecture

A semantics-based system, MrsQG[1], is developed to perform automatic question generation. It serves the following purposes:

1. It checks whether question generation based on MRS transformation is possible, in both theory and practice.

2. It constructs a general framework and test field for question generation on MRS, including modulized pre-processing, MRS manipulation, parsing and generation etc.

3. It reveals potential implementation problems that cannot be easily foreseen on paper.

Figure 4.1 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[2] 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. Thus each term, no matter how long it is, is treated as an un-splittable token in the initial parsing chart. This is extremely helpful when the ERG grammar fails to recognize unknown words, especially unusual proper nouns. For instance, generating from the MRS parsing result of the sentence "`Al Gore is a man.`" in LKB gives "`Alaska Gore is a man.`". By annotating the term "`Al Gore`" as a single token with a POS of NNP it avoids such funny and harmful mistakes.

[1] `http://code.google.com/p/mrsqg/`
[2] `http://wiki.delph-in.net/moin/PetInputFsc`

3. Parsing with PET. The chart mapping (Adolphs et al. 2008) functionality of PET accepts FSC input and outputs MRS structures. The issue of parsed MRS selection is also briefly addressed in Section 4.5.

4. MRS decomposition. Complex sentences need to be first broken into shorter ones with valid and meaningful semantic representation. Also, this shorter semantic representation must be able to generate output. This is the key point in this semantics-based approach. Details in Section 4.3.

5. MRS transformation. With a valid MRS structure of a sentence, there must be ways to replace EPs for terms with EPs for (WH) question words. Section 4.2 gives detailed description with examples.

6. Generating with LKB. Section 3.3.1 has already introduced the algorithm for generation in LKB.

7. Output selection. From a well-formed MRS structure, LKB might give multiple output. Depending on the generalization of ERG, some output might not sound fluent or even grammatical. Thus there must be ranking algorithms to select the best one. Details in Section 4.4.

8. Output to console/XML. Depending on the purpose, MrsQG outputs to console for user interaction or XML files for formal evaluation.

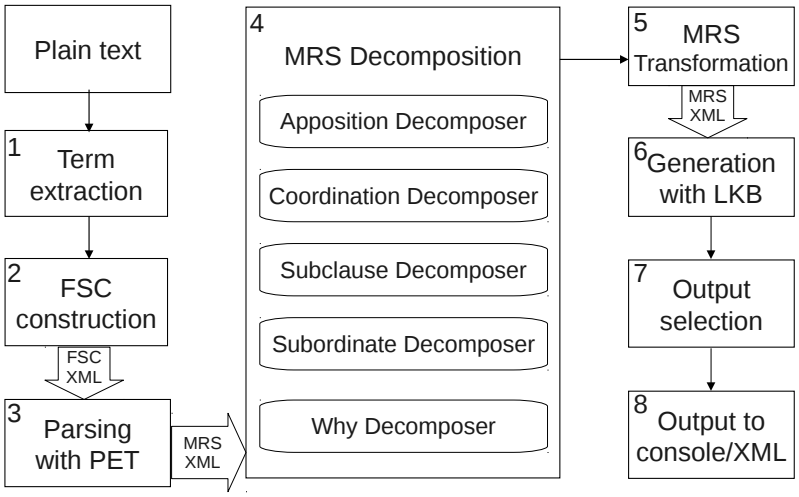


Figure 4.1.: Pipelines of an MRS transformation based question generation system.

We end this section by presenting readers some actual questions generated from MrsQG:

(4.1) `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 question

(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?`

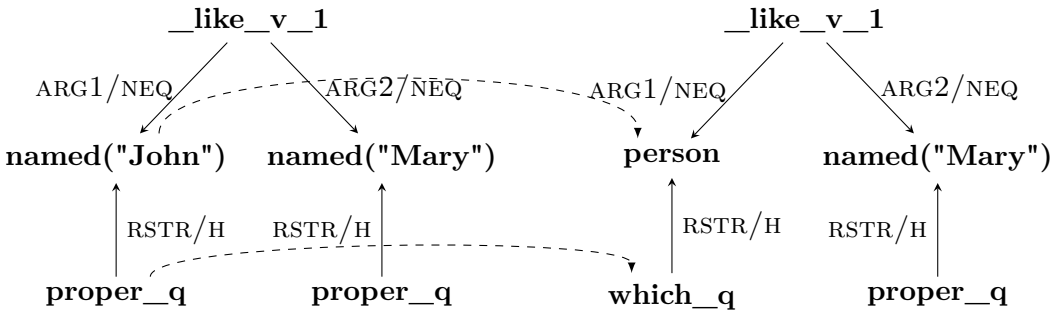
(l) `On August 29 , 1958, was Jackson born in Gary , Indiana?`

Obviously, it overgenerates. Section 4.4 addresses this issue. Following sections present more details on steps that need further elaboration.

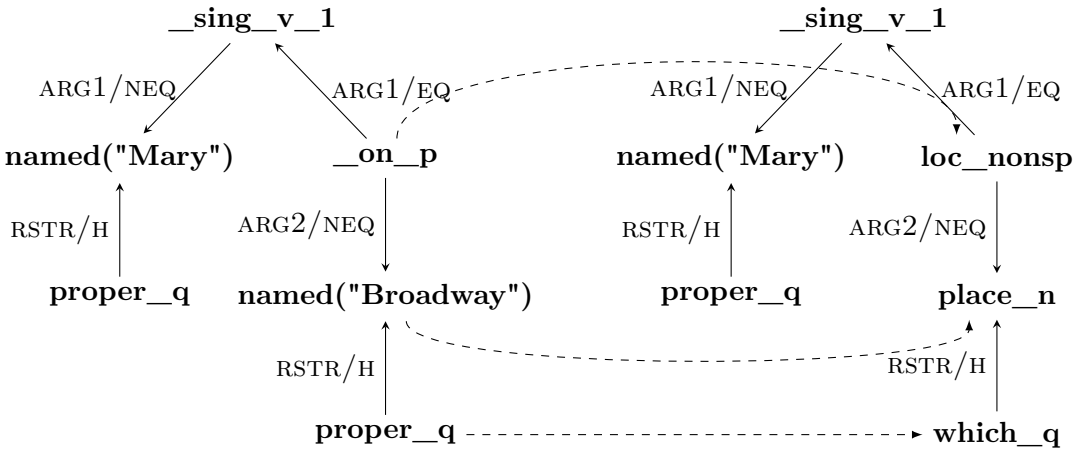
4.2. MRS Transformation for Simple Sentences

The transformation from declarative sentences into interrogatives follows a mapping between elementary predications (EPs) of relations. Figure 4.2 has shown this mapping. Most 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 parallel. For instance, the EPs of "`who`" consists 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.

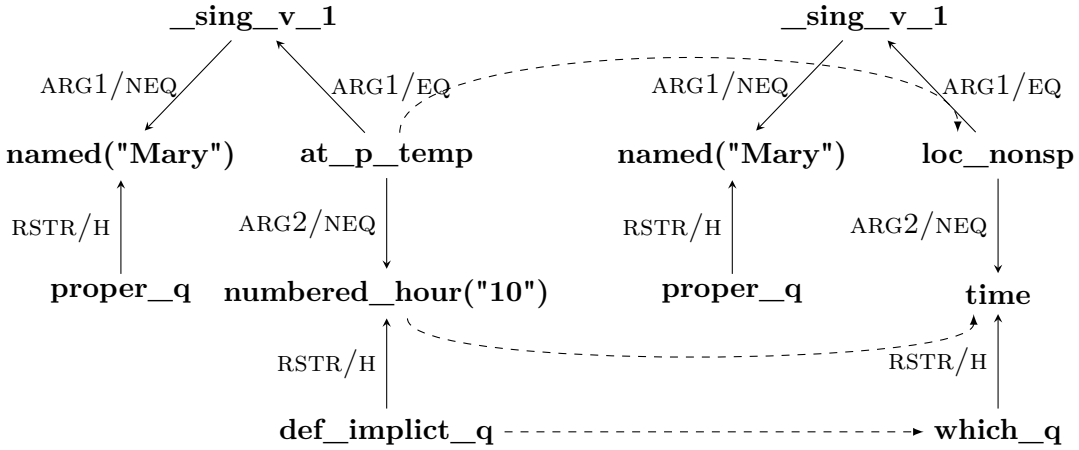
Similarly, in WHERE/WHEN/WHY questions, the EPs for the WH question word are which_q_rel and place_rel/time_rel/reason_rel. Special attention must be paid to the preposition word that usually comes before location/time. In a scoped tree, a preposition word stays on the same node of a semantic tree as the head of the phrase this PP is



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

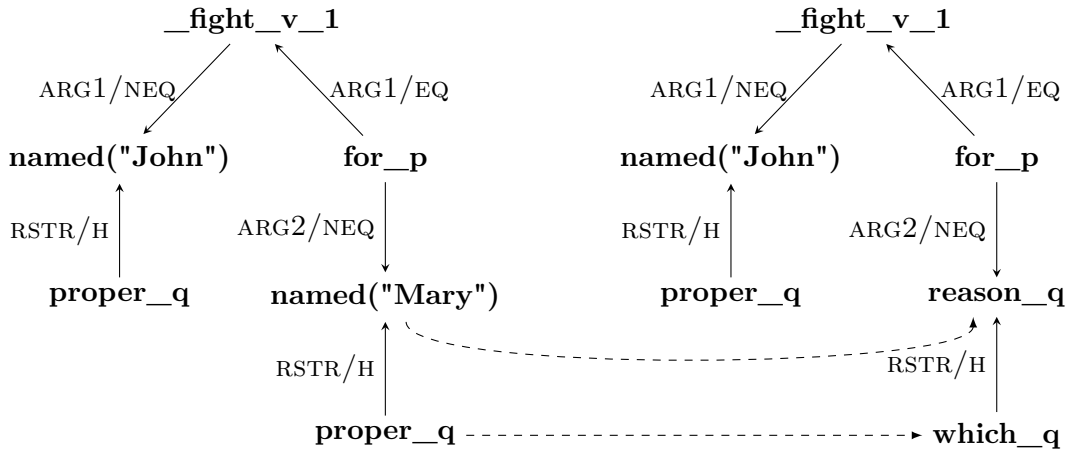


(b) "Mary sings on Broadway." → "Where does Mary sing?"



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

Figure 4.2.: (continued in the next page)



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

Figure 4.2.: MRS transformation from declarative sentences to WH questions in a form of dependency graph.

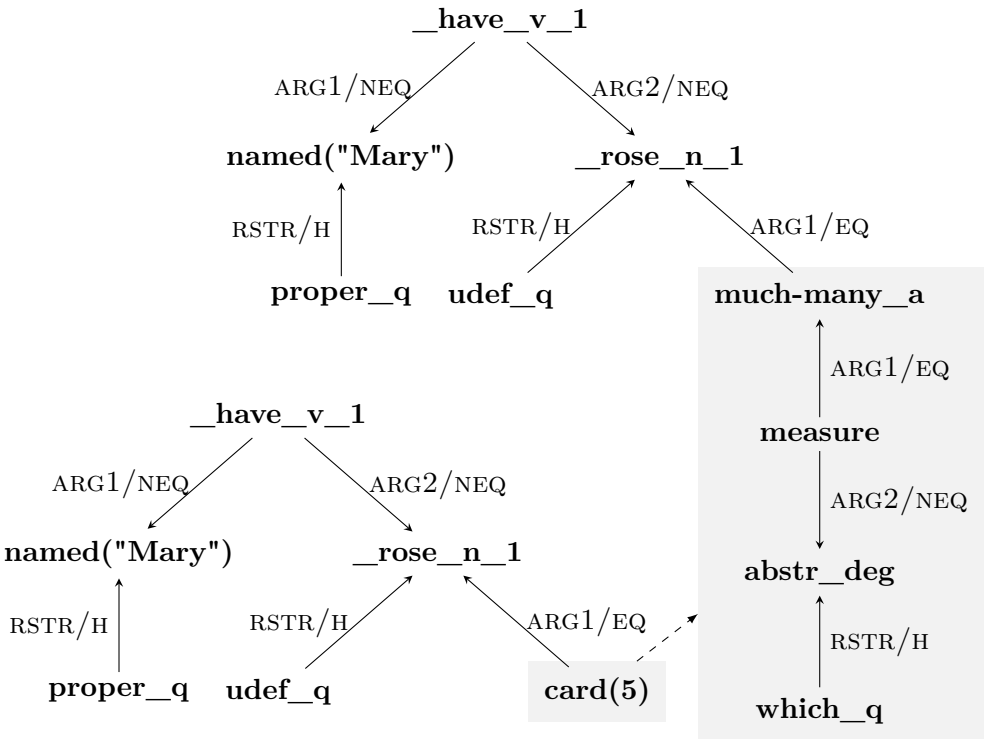


Figure 4.3.: MRS transformation from declarative sentences to HOW MANY/MUCH questions in a form of dependency graph. "`Mary has five roses.`" → "`How many roses does Mary have?`"

attached to; In a dependency tree, a preposition word governs the head of the phrase with a post-slash EQ relation (as shown in Figure 4.2(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 question words such as "`in where`" and "`on when`". In a normal parsing operation, the loc_nonsp_rel is inserted by some HPSG rule with a legitimate reason but for the simplicity of question generation, it is added by the replacement of the EP of the preposition.

Transforming EPs of numbers to EPs of HOW MANY/MUCH question words involves the addition of more elementary predications. Figure 4.3 tells asking a question on numbers needs a degree specifier for measures, i.e. measure_rel, which takes a MANY/MUCH relation much-many_a_rel as its ARG1 and an abstract "how" degree specifier abstr_deg_rel as its ARG2. Thus the final question word comes out as `how many` or `how much`, which is finally decided by the type of noun (whether it is singular, third-person, countable, etc) it modifies through the ERG.

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. However, since this is only a matter of auxiliary fronting, the generated YES/NO question always has an answer of YES.

4.3. MRS Decomposition for Complex Sentences

4.3.1. Overview

The MRS mapping between declarative and interrogative sentences only works for simple sentences. They generate lengthy questions from complex sentences, which is a not desirable result. For instance:

(4.2) `ASC takes a character as input, and returns the integer giving the ASCII code of the input character.`

Desired question:

(a) `What does ASC take as input?`

(b) `What does ASC return?`

Actual questions that could have been generated from MRS transformation:

(c) `What does ASC take as input and returns the integer giving the ASCII code of the input character?`

(d) `ASC takes a character as input and returns what giving the ASCII code of the input character?`

Questions in (c) and (d) are affiliated with more information which does not help the question itself. They sound unnatural and disturb the original intention of questions. Thus methods must be developed to obtain partial but intact semantic representations from complex sentences, such as one for "`ASC takes a character as input`" and the

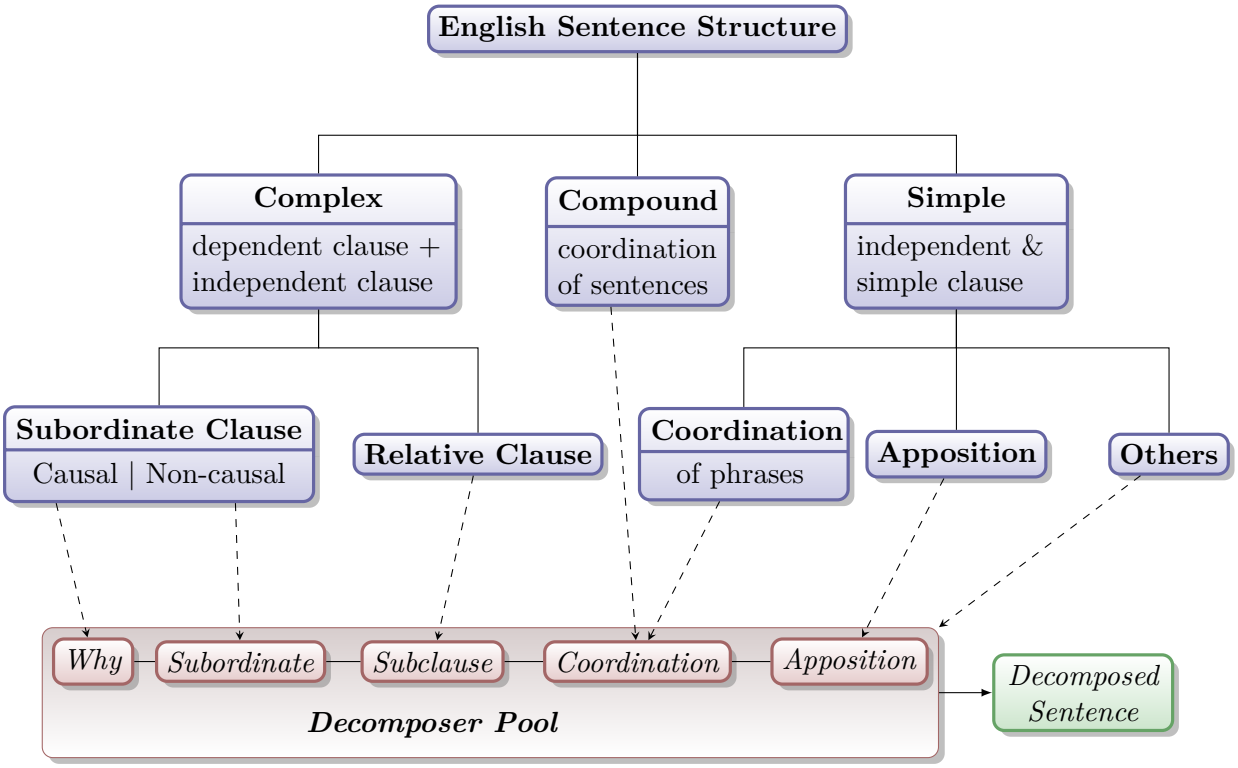


Figure 4.4.: The structure of English sentences and corresponding decomposers (in red boxes). Sentence decomposition does not know the type of sentence before hand thus all sentences will go through the pool of decomposers. The dashed arrows just indicate which decomposer works on which type of sentence.

other for "`ASC returns the integer giving the ASCII code of the input character.`". Then generate from simpler MRS representations.

MrsQG employs four decomposers for apposition, coordination, subclause and subordinate clause. 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*. They are determined by the types 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 also be classified into two types: subordinate clause and relative clause. Subordinate clauses are signaled by subordinate conjunctions and relative clauses

are introduced by relative pronouns. All kinds of sentences can be characterized by some linguistic phenomena, such as coordination and apposition. Compound sentences are attributed to coordination of sentences in this work, since it shares structure similarity with coordination of phrases.

Each type of sentence or grammar construction has a corresponding decomposer, shown in Figure 4.4. The order of applying these decomposers is not considered since in question generation from single sentences text cohesion is not important.

The following subsections walk through these decomposers and present the steps to construct a generic decomposing algorithm gradually.

4.3.2. Apposition Decomposer

Apposition is formed by two adjacent nouns describing the same reference in a sentence. In the ERG, it is identified by the type name appos_rel. The following is two types of apposition, the first one in restrictive form and the second one in non-restrictive form:

(4.3) `The boy John likes Mary.`

`John, a boy, likes Mary.`

Apposition is a relatively simple linguistic phenomenon but without proper treatment it harms question generation. For instance, in the preprocessing stage of the above example, the term extractor is only able to recognize "`John`" as a person name from the subject. After replacing the EP of "`John`" with the EP of "`who`", the sentence literately becomes "`The boy who likes Mary.`", which does not generate. If an apposition decomposer breaks the original sentence apart as follows:

(4.4) `The boy likes Mary.`

`John likes Mary.`

Then the second sentence can be transformed into a question: "`John likes Mary.`" → "`Who likes Mary?`". This is the purpose of an apposition decomposer in question generation.

Apposition in sentences can be precisely captured by the ERG. It is assigned an appos_rel EP that takes the two adjacent nouns as its arguments. Figure 4.5 shows how the apposition decomposer works. The basic idea is to replace the ARG1 of _like_v_1_rel with each of appos_rel's two arguments and build two new mrs's. In the ERG, it's always the ARG1 of appos_rel that is dependent on another EP (such as the verb), let's call it the governor EP. Thus in a more general sense the apposition decomposer first removes the appos_rel EP as well as its ARG2 to form an mrs1, then it builds the other mrs2 by replacing appos_rel's ARG1 with its ARG2 and removing appos_rel and its ARG1 afterwards. In mrs1 and mrs2, the governor EP takes a different EP as its argument, which is appos_rel's ARG1 in mrs1 and appos_rel's ARG2 in mrs2 respectively.

Note that in Figure 4.5 there is a dotted line between appos_rel and _like_v_1_rel. It is a NULL/EQ relation. The absence of the pre-slash part indicates neither of the two involved EPs depends on the other one. Thus the dotted line does not have an arrow

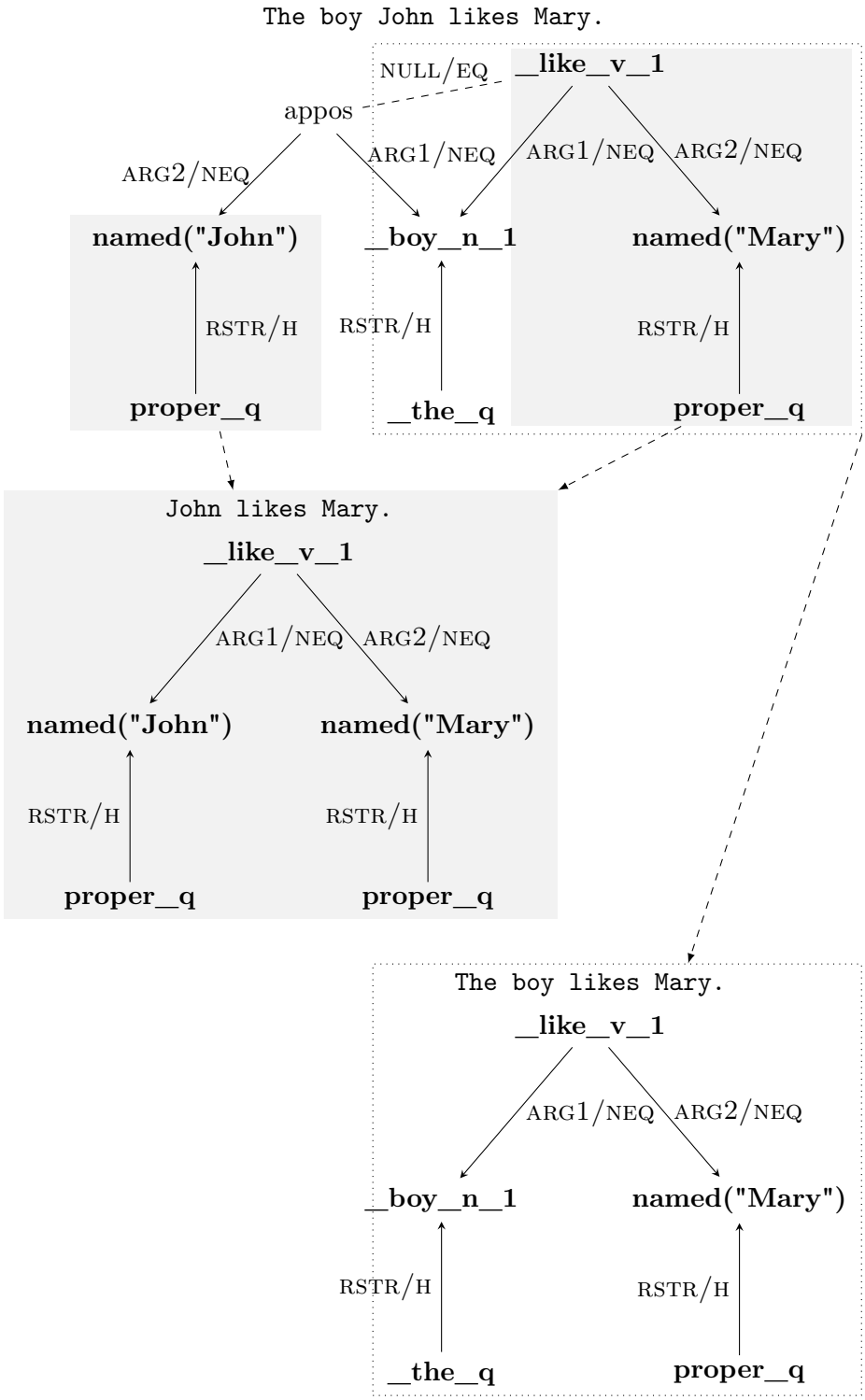


Figure 4.5.: A DMRS analysis of decomposing the apposition in "`The boy John likes Mary.`" to form two new DMRS's. The two shaded area at the top is combined to a new DMRS of "`John likes Mary.`". The EPs in the dotted box at the top is extracted to form the other DMRS for "`The boy likes Mary.`".

indicating a dependency relation. The post-slash part EQ indicates the two involved EPs have the same label, i.e. the same scope. The relation NULL/EQ is rare but does exist. It also shows a DMRS graph is only a connected acyclic graph, but not a directed acyclic graph.

4.3.3. Coordination Decomposer

Coordination is formed by two or more elements connected with coordinators such as "`and`", "`or`". In the ERG, it is identified by any relation that has a _c_rel suffix, such as _and_c_rel and _or_c_rel. It is a more complex grammatical construction than apposition in the sense that the elements involved are not confined to nouns. For instance,

(4.5) (a) `John likes cats and dogs.` (*coordination of* NP*s*)

(b) `John likes cats but hates dogs.` (*coordination of* VP*s*)

(c) `John likes cats and Mary likes dogs.` (*coordination of* S*s*)

Strictly speaking, coordination describes the linguistic phenomenon within the scope of simple sentences. Thus the targets of coordination should only contain phrases but not sentences. Coordination of sentences exceeds the definition of coordination. It constructs a *compound* sentence, rather than a simple sentence. However, since compound sentences share very similar patterns with coordination of phrases in the task of sentence decomposition, a coordination decomposer does not differentiate between them. We treat compound sentences as coordination of Ss here.

Example (4.2) has already shown the desired effects of a coordination decomposer. It should deal with different types of coordination. But not all decomposed sentences preserve the original meaning. For instance, a forceful disassembly of NP coordination can result in a counter effect,

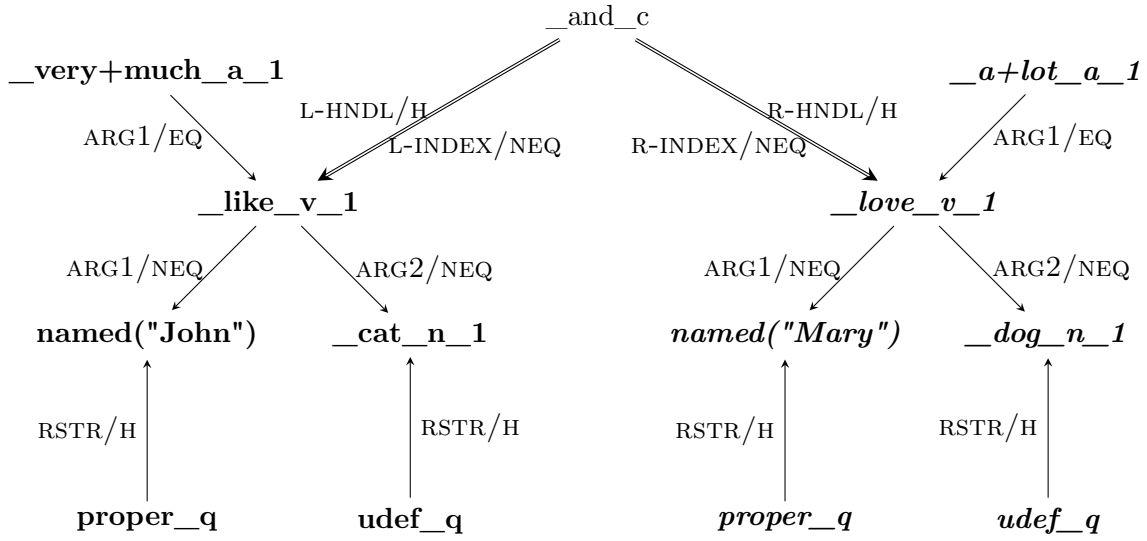
(4.6) `The cat and the dog live in the same place.`

`* The cat live in the same place.`

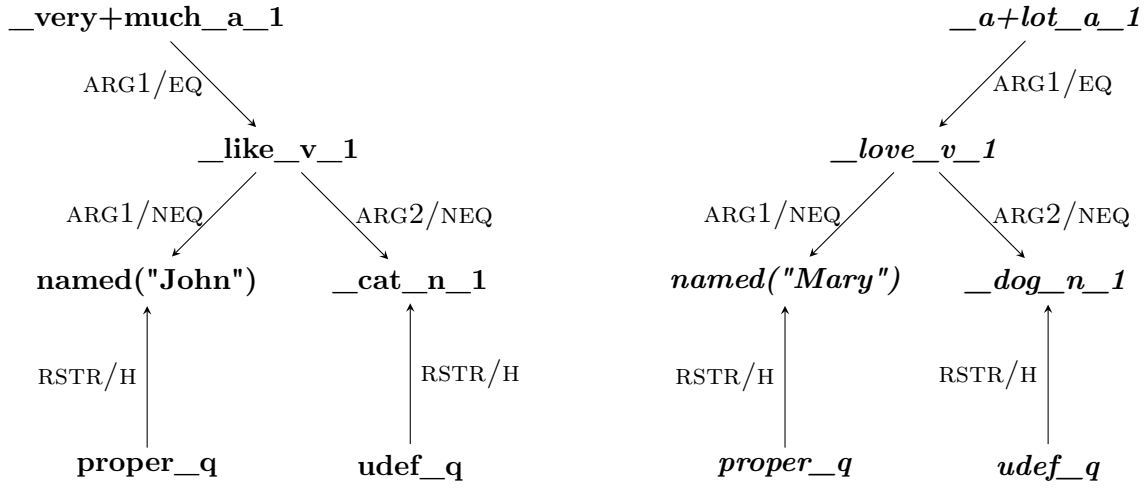
`* The dog live in the same place.`

Syntactically, the above decomposition violates the grammaticality of the sentence. Even if it can be fixed in a certain degree, the semantic meaning is not sound. Coordination of VPs and Ss is less problematic in this case and is our focus in the coordination decomposer.

Coordination of Ss is a simpler case than coordination of VPs because elements are less correlated. Figure 4.6 gives an example similar to Example (4.5c). It is clear-cut that there is no overlap between the L-INDEX (like_v_1_rel) and R-INDEX (love_v_1_rel) of the coordinator _and_c_rel. A naïve algorithm could have collected all EPs related to like_v_1_rel and built a new mrs based on that. However, this simple example is a bit deceptive because the DMRS representation of a sentence is unaware of the type of coordination (whether it is coordination of Ss or coordination of VPs). The naïve algorithm only works for coordination of Ss and fails when the L-INDEX and R-INDEX share some EP (such as in the case of coordination of VPs). A more complex but general example is given in Figure 4.7.

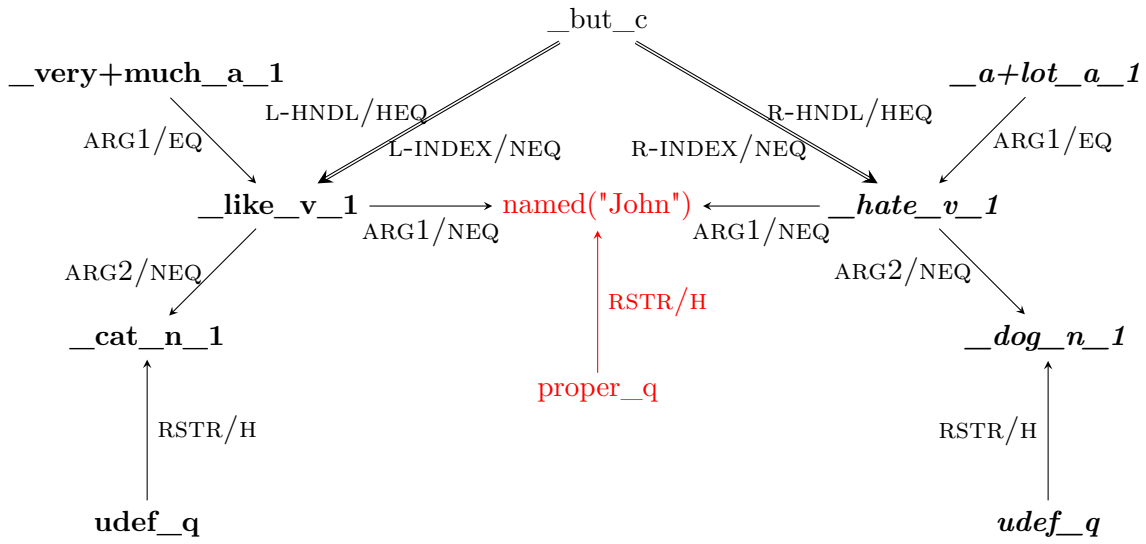


(a) "`John likes cats very much and Mary loves dogs a lot.`"

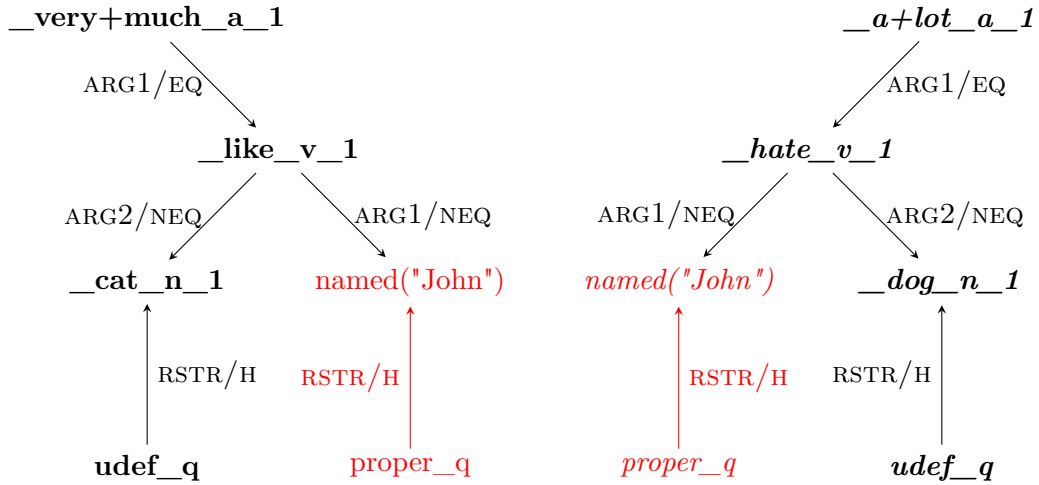


(b) left: "`John likes cats very much.`" right: "`Mary loves dogs a lot.`"

Figure 4.6.: An S coordination decomposer finds out the EPs that the coordinator's L-INDEX and R-INDEX refer and assemble two new mrs's. In (a) _and_c_rel's L-INDEX refer to _like_v_1_rel. All EPs related to _like_v_1_rel (highlighted in bold) are taken out and assembled to a new mrs in the left of (b). Similarly _but_c_rel's R-INDEX refer to _love_v_1_rel. All EPs related to _love_v_1_rel (in bold italics) are taken out and assembled to a new mrs in the right of (b). Note that _but_c_rel governs its arguments by double relations: L/R-HANDLE and L/R-INDEX.
in (a) *decompose*({_like_v_1}, {_and_c}) → (b) left
in (a) *decompose*({_love_v_1}, {_and_c}) → (b) right (c.f. Algorithm 4.1)



(a) "John likes cats very much but hates dogs a lot."



(b) left: "John likes cats very much." right: "John hates dogs a lot."

Figure 4.7.: A VP coordination decomposer works similarly as the S coordination decomposer in Figure 4.6. The difference is highlighted in red: both _like_v_1_rel and _hate_v_1_rel refer to the same subject proper_q_rel(named("John")). Thus when the decomposer tries to find all EPs related to _and_c_rel's L-INDEX (_like_v_1_rel), it must not include _hate_v_1_rel and its related EPs, as shown in the left of (b).
in (a) *decompose*({_like_v_1}, {_but_c}) → (b) left
in (a) *decompose*({_hate_v_1}, {_but_c}) → (b) right (c.f. Algorithm 4.1)

In Figure 4.7 the subject named("John") is governed by both verbs in the sentence, _like_v_1_rel and _hate_v_1_rel. Recall that in *Davidsonian* representation of semantics all verbs introduce *event*s. Thus the two verb EPs introduce two different events. A decomposer should separate them by keeping one and ruling out the other. Thus starting from the L/R-INDEX (name it as the INDEX-EP) of the coordinator, a coordination decomposer travels through all the governors and dependants (name them as the REF-EPS) of INDEX-EP, collects them and excludes any verb who governs REF-EPS by its argument, then rebuilds a new mrs.

The algorithm described above does not take scopes into consideration and thus is problematic in more complicated cases. The following example involving a relative clause shows this:

(4.7) (a) `The man who adopted Bart likes cats but hates dogs.`

Sentences extracted after coordination decomposition:

(b) `The man likes cats.`

(c) `The man hates dogs.`

Sentences that should have been extracted after coordination decomposition:

(d) `The man who adopted Bart likes cats.`

(e) `The man who adopted Bart hates dogs.`

During the process of coordination decomposition, the relative clause "`who adopted Bart`" is lost. We will explain why the man adopted Bart and revise the algorithm in subsection 4.3.5.

4.3.4. Subordinate Decomposer

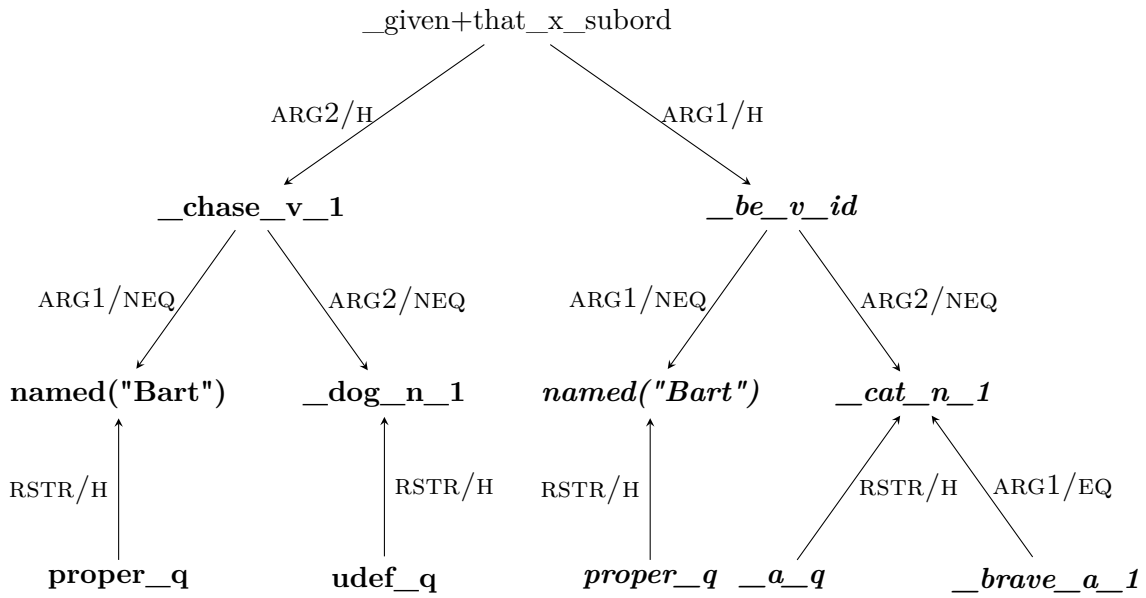
A subordinate decomposer works on sentences containing dependent clauses. A dependent clause "depends" on the main clause, or an independent clause. Thus it cannot stand alone as a complete sentence. It starts with a subordinate conjunction and also contains a subject and a predicate. For instance,

(4.8) `Given that Bart chases dogs, Bart is a brave cat.`

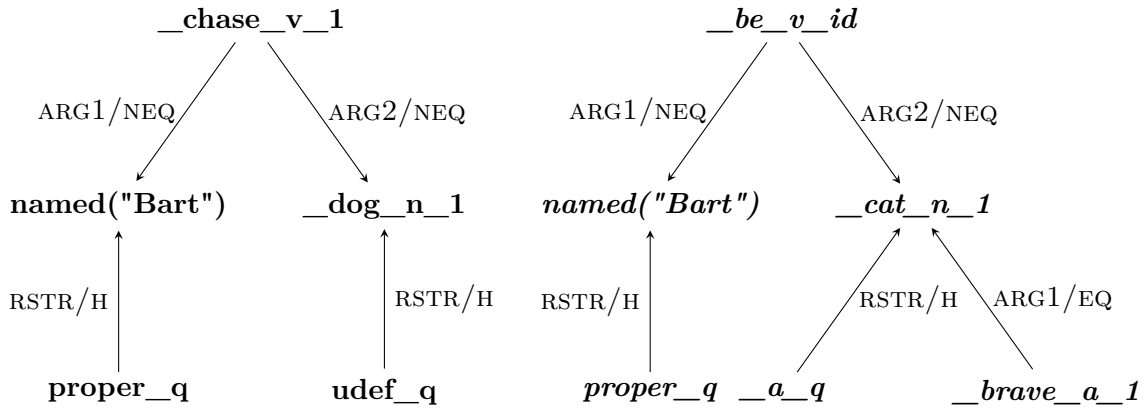
The part before the comma is the dependent clause, which starts with a subordinate conjunction ("`given that`") and followed by an independent clause ("`Bart is a brave cat.`"). In the ERG, the subordinate conjunction is mainly identified by a _subord_rel suffix, such as _given+that_x_subord_rel, _once_x_subord_rel, or sometimes a _x_.*rel pattern, such as _although_x_rel, _unless_x_rel, _if_x_then_rel.

The algorithm for a subordinate decomposer is the same as the one for a coordination decomposer. The ARG1 of the subordinate EP refers to the main clause while the ARG2 refers to the dependent clause. Thus the decomposer extracts all EPs related to ARG1/2 separately and builds two new mrs's. Figure 4.8 illustrates this.

Although the dependent clause contains a subject and a predicate, which in turn form a sentence, the proposition of this sentence can be either true or false:



(a) "`Given that Bart chases dogs, Bart is a brave cat.`"



(b) left: "`Bart chases dogs.`" right: "`Bart is a brave cat.`"

Figure 4.8.: A subordinate clause decomposer. It works in the same way as the coordination decomposer.
in (a) *decompose*({_chase_v_1}, {_given+that_x_subord}) → (b) left
in (a) *decompose*({_be_v_id}, {_given+that_x_subord}) → (b) right (c.f. Algorithm 4.1)

(4.9) (a) `Given that Bart chases dogs, Bart is a brave cat.`

(b) `If Bart chases dogs, Bart is a brave cat` (in a world where cats fear to chase dogs).

The proposition "Bart chases dogs" is true in (a) but could be false (it can not be inferred) in (b). Thus if a subordinate decomposer extracts a sentence "`Bart chases dogs`." from (b), it does not truly inherit the original meaning. The boolean value of the proposition of the extracted sentence depends mostly on a more fine-grained classification of the types of subordinate conjunction. However, this has been left as a future work to the subordinate decomposer.

4.3.5. 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*, *who*m, *who*se, *which*, *whomever*, *whatever*, and *that*. Extracting relative clauses from a sentence helps asking better questions. For instance, given the following sentence:

(4.10) (a) `Bart is the cat that chases the dog.`

Extracted relative clause after decomposition:

(b) `The cat chases the dog.`

Questions asked on the relative clause:

(c) `Which animal chases the dog?`

(d) `Which animal does the cat chase?`

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

Though relative pronouns indicate relative clauses, in an MRS structure, these relative pronouns are not explicitly represented. For instance, in Figure 4.9(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.

Relative clauses with prepositions are less frequent but also common. Usually a preposition is used before the relative pronoun. Sometimes the preposition also appears after the verb if there is no relative pronoun in the sentence:

(4.11) (a) `Mary is the girl with whom John lives.` (with a relative pronoun)

(b) `Mary is the girl John lives with.` (zero relative pronoun)

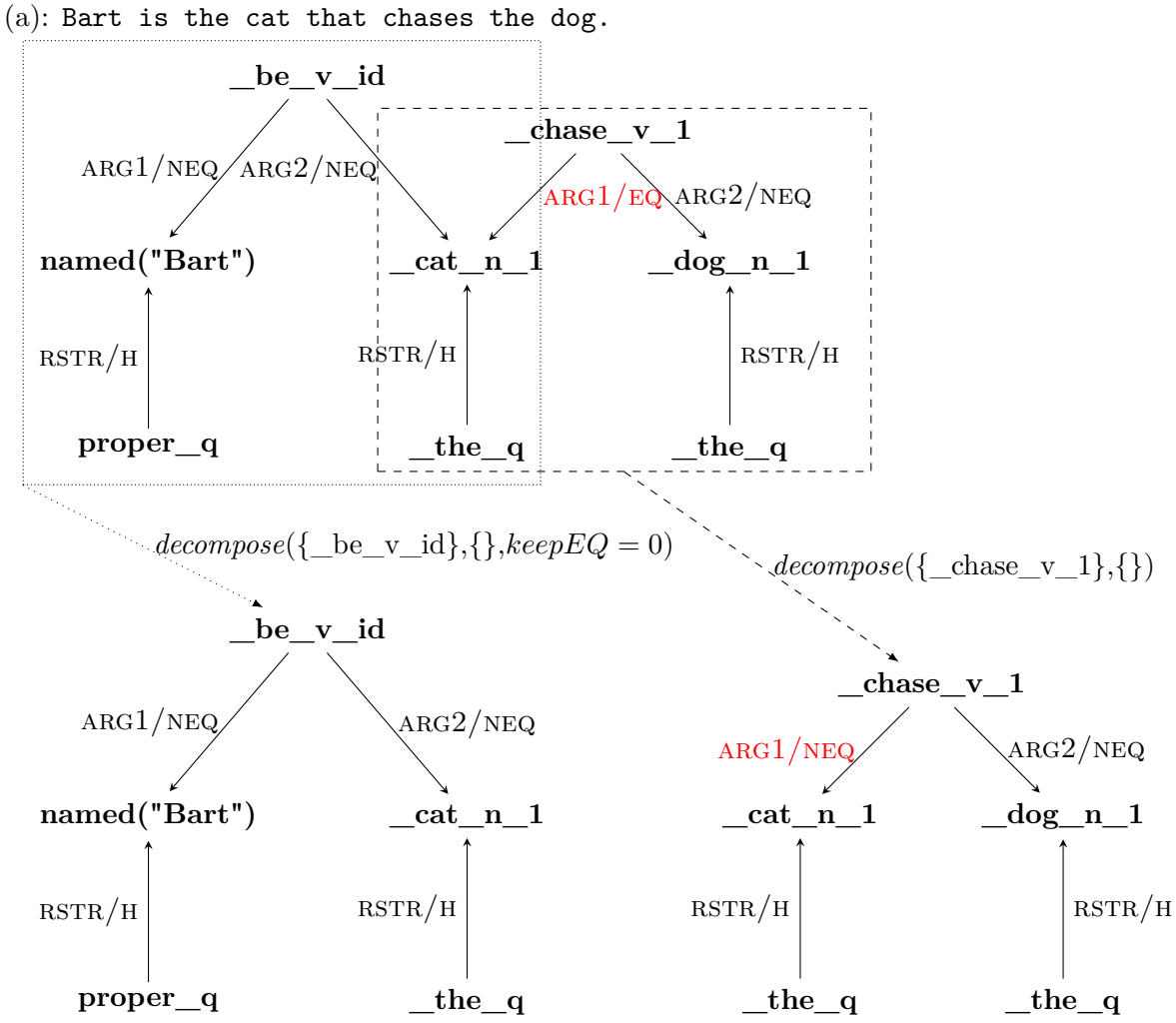


Figure 4.9.: A subclause decomposer extracts relative clauses by finding all non-verb EPs that are directly related to a verb EP. It also relaxes scope constraint from a relative clause. A strict EQ relation in the top graph represents an NP "`the cat that chases the dog`". Relaxing it to NEQ generates a sentence "`the cat chases the dog.`" in the bottom right graph.

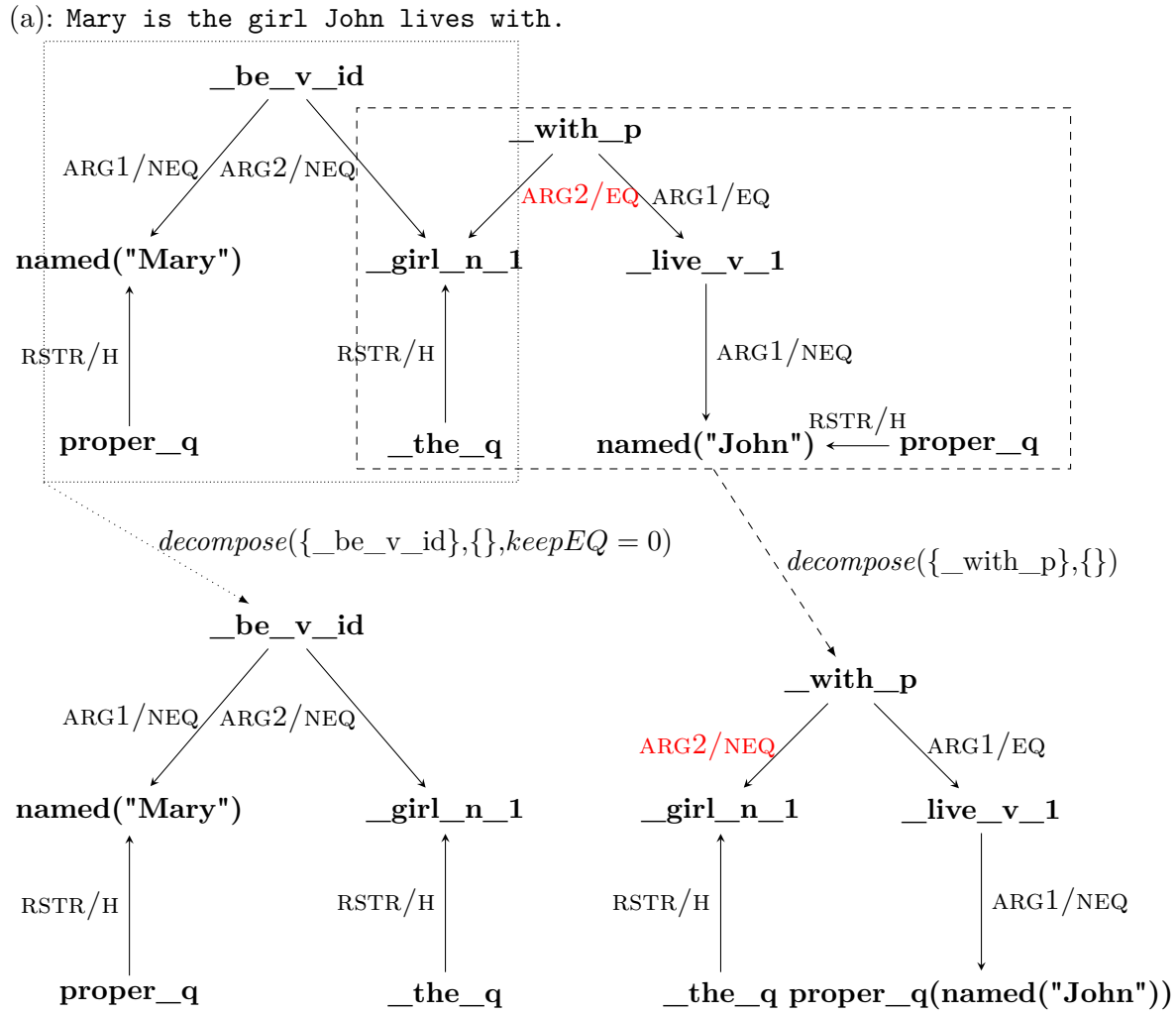
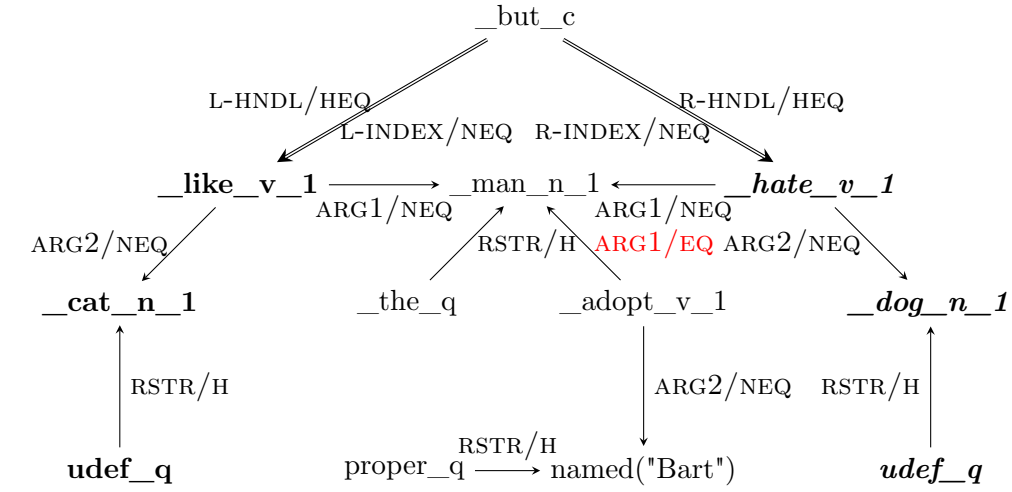
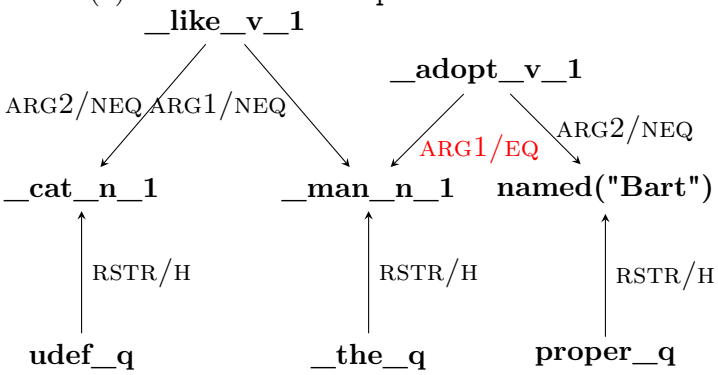


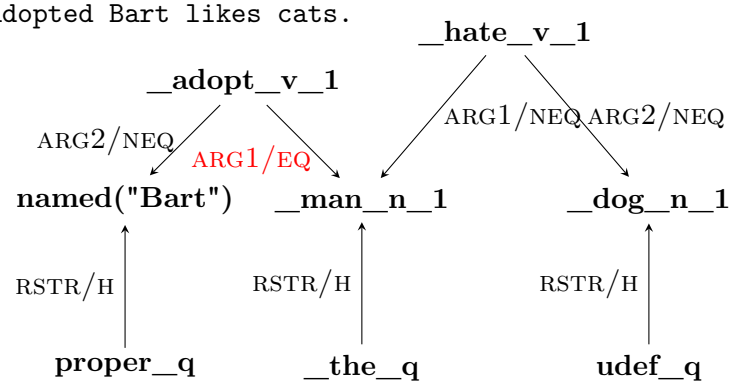
Figure 4.10.: Subclause decomposition for relative clauses with a preposition. The preposition EP _with_p connects the subclause with the main clause. After decomposition, the dependency relation between _with_p and _girl_n_1 is relaxed from ARG2/EQ in (a) to ARG2/NEQ in (c).



(a): `The man who adopted Bart likes cats but hates dogs.`



(b): `The man who adopted Bart likes cats.`



(c): `The man who adopted Bart hates dogs.`

Figure 4.11.: A revised coordination decomposer that preserves relative clauses. Taking (a) → (b) as an example, it travels from _but_c_rel's L-INDEX and collects all EPs that are related to _like_v_1_rel, directly or indirectly. The subject EP _man_n_1 is governed by two verb EPs: _have_v_1 and _adopt_v_1. Only _adopt_v_1 is preserved because it governs _man_n_1 by a post-slash EQ relation, indicating relative clause.
in (a) *decompose*({_like_v_1}, {_but_c}) → (b)
in (a) *decompose*({_hate_v_1}, {_but_c}) → (c) (c.f. Algorithm 4.1)

With or without a relative pronoun, the ERG gives identical MRS interpretation to both sentences. Without a preposition, the verb in a relative clause connects the clause to its main clause. However, mrs's with or without a preposition are different. With a preposition, it is the preposition 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 4.10.

Finally we revise the algorithm used in the coordination decomposer. We have just shown that relative pronouns are implicitly indicated by the post-slash EQ relation between a verb and its argument. Coming back to the example (4.7) involving a relative clause in coordination from subsection 4.3.3:

(4.12) (a) `The man who adopted Bart likes cats but hates dogs.`

Sentences extracted with the old coordination decomposition algorithm:

(b') `The man likes cats.`

(c') `The man hates dogs.`

Sentences that should be extracted after coordination decomposition (relative clauses should be kept):

(b) `The man who adopted Bart likes cats.`

(c) `The man who adopted Bart hates dogs.`

Sentences extracted after subclause decomposition on (b) and (c):

(d) `The man adopted Bart.`

(e) `The man likes cats.`

(f) `The man hates dogs.`

Thus we should revise the coordination decomposition algorithm as: "starting from the L/R-INDEX (name it as the INDEX-EP) of the coordinator, a coordination decomposer travels through all the governors and dependants (name them as the REF-EPS) of INDEX-EP, collects them and *excludes* any verb who governs REF-EPS by its argument **with a post-slash NEQ relation**, then rebuilds a new mrs.". Figure 4.11 shows the graph of decomposing (a) into (b) and (c). Then the decomposed (b) and (c) go to a subclause decomposer.

4.3.6. Why Decomposer

A why-question decomposer is a special subordinate decomposer for causal sentences. Causal conjunctions in ERG are taken as cue EPs of causal sentences, such as _because_x_rel, _therefore_x_rel, etc. A why-question decomposer tells which part of a sentence is the reason and which part is the result by walking through the ARG1/2 values of the cue EP. Figure 4.12(a) shows the reason and result part of a sentence. It is an MRS reading of the following sentences:

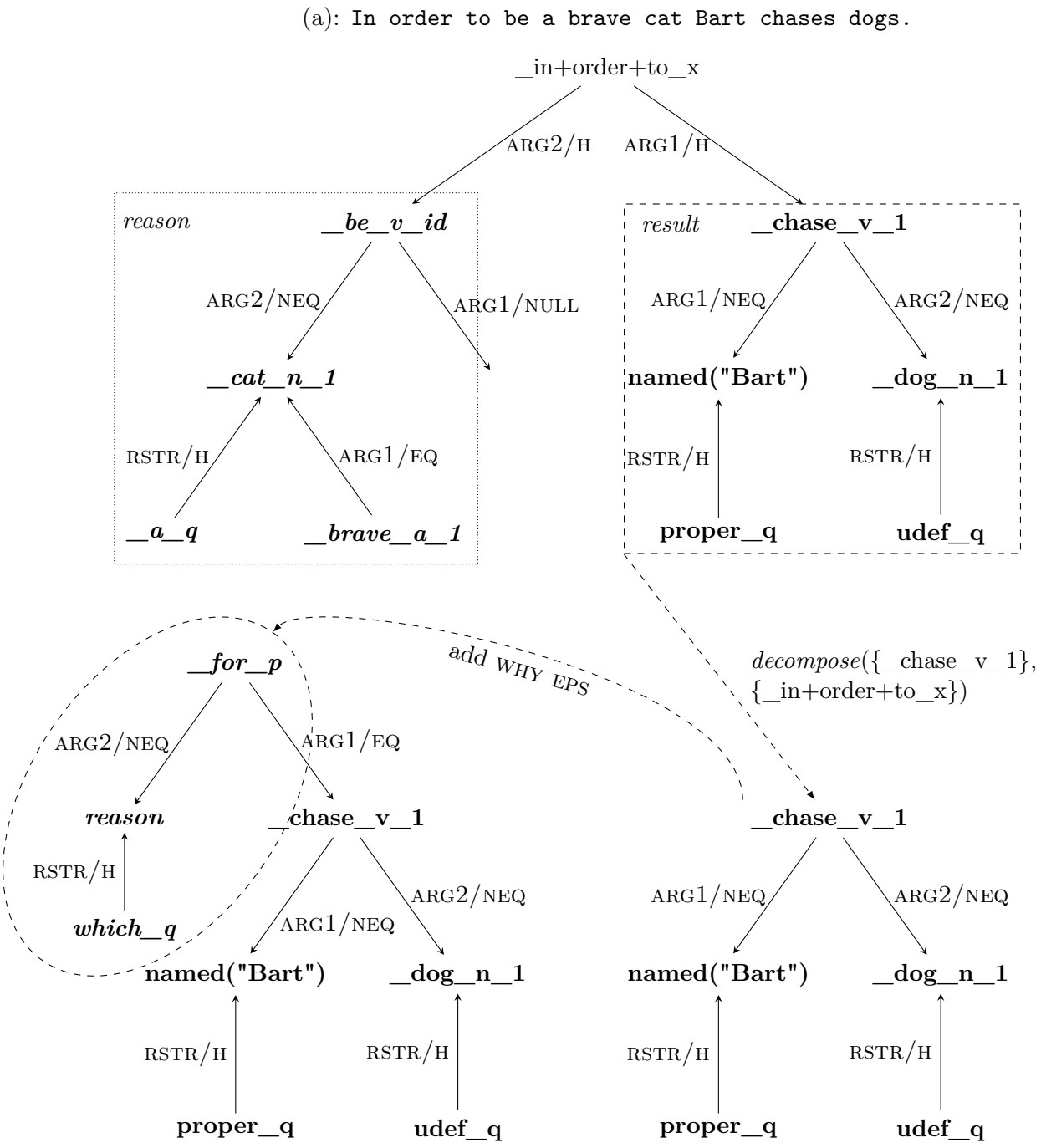


Figure 4.12.: A decomposer for causal sentences. The cue EP in (a) is _in+order+to_x. Its ARG1 is the *result* and ARG2 is the *reason*. The result part is extracted to form a sentence (b) and it can be transformed to a WHY question in (c) by adding the EPs for the question word. The reason part has an empty ARG1/NULL argument and does not stand alone as a sentence.

(4.13) (a) `In order to be a brave cat Bart chases dogs.` (in a world where dog-chasing cats are labeled as warriors.)

(b) `In order to be a brave cat, Bart chases dogs.`

(c) `Bart chases dogs in order to be a brave cat.`

It is clear-cut in Example (4.13b) and (4.13c) which part is the reason and which part is the result: in (b) the two parts are separated by the comma and in (c) by the cue words in the middle. However, in (a) it is hard to judge from the surface words. A DMRS graph shows the boundary distinctly in Figure 4.12(a).

Finding causal relations by cue words has its limitations. It does not have a good coverage and sometimes can be ambiguous. For instance, _since_x_subord_rel can either signal a causal relation or only a time frame. Thus a why-question decomposer functions as an add-on decomposer to increase the types of questions generated. It does not go deeply to discover implicit causal relations.

4.3.7. General Algorithm

The previous subsections have introduced various decomposers that work on different types of sentences. Examples of each type of sentence are also given. In this subsection we describe the general algorithm for sentence decomposition.

All of the decomposers employ a general algorithm that extracts related EPs of a certain EP through a dependency graph. This generic algorithm is the key step for decomposing complex MRS structures. Before describing it, we first sum up the dependencies and their intricacies. Note that the following is only a list of examples that are shown in this thesis and thus possibly 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`" in Figure 4.8.
  - Adverbs govern verbs, such as _very+much_a_1 → _like_v_1 from "`like ... very much`" in Figure 4.7.
  - Prepositions govern and attach to verbs, such as _with_p → _live_v_1 from "`live with ...`" in Figure 4.10.
  - Passive verbs govern and modify phrases, such as _chase_v_1 → _dog_n_1 from "`the chased dog`".
  - 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 4.9.
    - Prepositions govern phrases that are represented by relative pronouns, such as _with_p → _girl_n_1 from "`the girl John lives with`" in Figure 4.10(a).

- 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`" in Figure 4.8.
  - Prepositions govern phrases after them, such as _with_p → _girl_n_1 from "`lives with the girl`" in Figure 4.10(c).
  - Some grammatical construction relations govern their arguments, such as appos_rel → named("John") and appos_rel → _boy_n_1 from "`the boy John`" in Figure 4.5.
- 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." in Figure 4.8.
  - 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.
  - E.g. there is a NULL/EQ relation between appos and _like_v_1 in Figure 4.5. NULL/EQ involves a complicated determination of equalities. More details in Copestake (2008).
- 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" in Figure 4.6.
- 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" in Figure 4.7.

The above are so far all relations that has been considered in this thesis. We first give a formal definition of a connected DMRS graph, then a generic algorithm for DMRS decomposition.

Definition. 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.

The definition of L and its two conditions need further elaboration. Normally a function $L(x, y)$ returns only one pair. But in some rare cases it returns two: such as the double edge from _but_c in Figure 4.11. For the simplicity of following description of algorithms, we assume $L(x, y)$ only returns one pair from now on.

Regarding L's two conditions, recall that in Figure 4.5 there is a NULL/EQ relation between appos and _like_v_1: they do not have any dependencies in between so the edge has no direction. This explains that if $pre =$ NULL, then the edge between (x, y) is not directed. In Figure 4.12(a) the ARG1 of _be_v_id is empty: _be_v_id has an ARG1/NULL relation with a null node. This explains that if $post =$ NULL, then $y =$ NULL, x has no dependant by a pre relation.

Algorithm 4.1 shows how to find all related EPs for some target EPs. It is a graph travel 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. In the most simple case, given a compound sentence:

(4.14) `John likes cats very much and Mary loves dogs a lot.` (Figure 4.6)

The set of target EPs is {_like_v_1}, the L-INDEX of _and_c, or {_love_v_1} and the R-INDEX of _and_c. Taking the L-INDEX as an example, if we start from _like_v_1 and travel through the graph, we will pass by the subject (`John`), the object (`Mary`) and the adverb (`very much`). We put _and_c to the exception set of EPs so we do not go further beyond it to mistakenly retrieve its R-INDEX. By this way we can collect all related EPs to the initial target set {_like_v_1}:

(4.15) *decompose*({_like_v_1}, {_and_c}) → `John likes cats very much.`
decompose({_love_v_1}, {_and_c}) → `Mary loves dogs a lot.`

Algorithm 4.1 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 verbsand 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

There are two optional parameters for the algorithm that define different behaviors of graph traversal. *relaxEQ* controls whether we want to relax the scope constraint 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 4.9(c) and 4.10(c). *keepEQ* controls whether we want to keep a subclause introduced by a verb or preposition. It is set to true by default. If set to false, the relative clause will be removed from the main clause. Examples can be found in Figure 4.9(b) and 4.10(b).

All decomposers except the apposition decomposer employs this generic algorithm. Figures 4.6 to 4.12 are marked with corresponding function calls of this algorithm that decompose one complex MRS structure to two simpler ones.

MRS decomposition is the main idea behind the task of sentence simplification. There are quite some other research issues we have not touched in this section, such as anaphora resolution and text cohesion. But the skeleton is already presented. In this section we first gave multiple examples of English sentences to evoke the introduction of different decomposers. These examples also motivate and elaborate the inner algorithm for connected DMRS graphs discussed at the end. By breaking complex sentences into simpler ones the task of question generation can be better performed.

4.4. Automatic Generation with Rankings

4.4.1. Overview

MrsQG is written in Java. It calls PET for parsing and LKB for generation, which are written in C++ and Common Lisp respectively. All of them need to load a considerable amount of model or grammar files to work properly while a practical working system can only afford loading only once for any quick batch processing or good user interactions. How to connect these three systems together and run them without human intervention is an engineering challenge. MrsQG has integrated both PET and LKB. They run in server mode and communicate with MrsQG in XML. Thus the pipeline of parsing-transformation-generation is totally automatic.

However, this automatic process usually produces too much output. The reasons are twofold: Firstly, PET and ERG "over-parse". Sometimes even a simple sentence has tens of parsed MRS structures due to substantial unification rules. Secondly, LKB and ERG overgenerate. Sometimes even a simple MRS structure has tens of realizations also due to substantial unification rules. Thus, even if the 5 best MRS structures are taken from a parsed sentence, and the 5 best realizations are taken from each MRS structure, a single sentence can still have 25 realizations (including duplicates).

The over-parsing problem has been alleviated by statistical parse selection proposed by Zhang et al. (2007a). The most confident parses are among the top selections. The over-generation problem is more complicated and is the focus of this section. The following gives a simple example.

Recall that an mrs does not bear any EPs for relative pronouns. It is reasonable that an MRS structure does not make a difference between "`This is the proposal John`

made.” and “This is the proposal that John made.”. But during realization, the possible choices of relative pronouns are enumerated:

(4.16) This is the proposal John made.

This is the proposal that John made.

This is the proposal which John made.

This is the proposal who John made.

This is the proposal, that John made.

This is the proposal, which John made.

This is the proposal, who John made.

ERG does not give a preference to the types of relative pronoun and the restrictiveness of the relative clause. Heuristic rules can be defined on top of LKB generation to select the best one but this requires an experienced knowledge of how ERG overgenerates and even some world knowledge, such as that “the proposal” is not a “who”. A simple and effective way instead of heuristics is to statistically rank the output.

Section 2.4 has already presented various ranking techniques. An LKB derivative from the LOGON infrastructure[3] has included the MaxEnt model shown in Table 2.1, which has an exact match accuracy of 64.28% and 5-best accuracy of 83.60% on a balanced cross-domain dataset. Note that the MaxEnt model only works best for declarative sentences since its training corpus does not include questions. Thus question ranking is needed on top of this MaxEnt model. A simple way is to use language models trained only on questions.

4.4.2. N-gram Language Model Essentials

An n-gram language model is used to approximate the probability of m consecutive words:

$$P(w_1, \ldots, w_m) = \prod_{i=1}^{m} P(w_i|w_1, \ldots, w_{i-1}) \approx \prod_{i=1}^{m} P(w_i|w_{i-(n-1)}, \ldots, w_{i-1}) \qquad (4.17)$$

The above equation interprets the probability of m consecutive words as the product of the conditional probability of the i-th word given the history of all the $n-1$ words before it. The conditional probability is thus estimated by a context window of only n words, rather than the whole history. Various methods can be applied to calculate this conditional probability, all involving a process of counting and smoothing. Even though, these methods still do not enumerate all the combinations of the words in an n-word context window. Thus the idea of *back-off* is introduced to n-gram language models: the probability of n words backs off to the probability of $n-1$ words if these n words are not observed from the training corpus.

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

An n-gram language model contains two types of numbers: log-likelihood of smoothed probability P_{lm} for all observed word sequences from unigram to n-gram and back-off weight α_{bo} for all observed word sequences from unigram to $n-1$-gram. Thus if an n-gram does not exist, it backs off to $n-1$-gram and this process applies recursively if $n-1$-gram does not exist. An example of trigram is listed in Figure 4.13.

In practice, with an n-gram language model, the conditional probability of the n-th word given the previous $n-1$ words can be calculated as (Jurafsky and Martin 2008):

$$P(w_n|w_1,\ldots,w_{n-1}) = \begin{cases} P_{lm}(w_n|w_1,\ldots,w_{n-1}) & if\ (w_1,\ldots,w_n)\ exists \\ \alpha_{bo}(w_1,...,w_{n-1})P(w_n|w_2,\ldots,w_{n-1}) & otherwise \end{cases} \tag{4.18}$$

Substituting Equation 4.18 in Equation 4.17 gives the probability of a sequence of m words. A typical value of m is the length of a sentence.

In terms of a trigram model for the word sequence (x, y, z), the probability is:

$$P(z|x,y) = \begin{cases} P_{lm}(z|x,y) & if\ (x,y,z)\ exists \\ \alpha_{bo}(x,y)P(z|y) & else\ if\ x,y\ exists \\ P_{lm}(z) & otherwise \end{cases} \tag{4.19}$$

$$P(z|y) = \begin{cases} P_{lm}(z|y) & if\ (y,z)\ exists \\ \alpha_{bo}(x)P_{lm}(z) & otherwise \end{cases} \tag{4.20}$$

Perplexity (PP) is a measure of the quality of a language model. It is usually calculated on a test set. For a test set $W = w_1 w_2 \ldots w_N$, perplexity is defined as:

$$PP(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \tag{4.21}$$

$P(w_1 w_2 \ldots w_N)$ is already defined in Equation 4.17. Note that N introduces a normalization factor to make sure that perplexities between different length of words are comparable. Perplexity is a better indicator than probability in the task of question generation because both long questions and short questions can be generated from a sentence and it is not always shorter questions (which usually come with a higher probability) are better. Perplexity is balanced over the length of sentences thus it is actually used to rank questions in this work.

4.4.3. Building N-gram Language Models

There is no standard corpus with mostly questions. Thus we need to collect questions and construct a corpus. However, the currently collected questions are still too few thus it needs to be adapted with some other wider-coverage corpora. Thus the whole procedure follows a standard protocol of "*build-adapt-prune-test*". Firstly we *build* a small language model for only questions. Then this language model is *adapt*ed with

```
\data\
ngram  1=          9
ngram  2=          8
ngram  3=          8


\1-grams:
-1.09691    <s> -0.401401
-1.09691    marry   -0.264818
-1.09691    is  -0.264818
-1.09691    married -0.264818
-1.09691    to  -0.264818
-1.09691    john    -0.264818
-1.09691    .   -0.264818
-1.09691    </s>    -0.264818
-0.443698   <unk>

\2-grams:
-0.30103    <s> <s> 0.0791812
-0.778151   <s> marry   0
-0.30103    marry is    0
-0.30103    is married  0
-0.30103    married to  0
-0.30103    to john 0
-0.30103    john .  0
-0.30103    . </s>  0

\3-grams:
-0.39794    <s> <s> <s>
-0.69897    <s> <s> marry
-0.30103    <s> marry is
-0.30103    marry is married
-0.30103    is married to
-0.30103    married to john
-0.30103    to john .
-0.30103    john . </s>
\end\
```

Figure 4.13.: A trigram language mode for the sentence "`<s> marry is married to john . </s>`". `<s>` and `</s>` mark the start and end of a sentence. The number in the left-most column of 1/2/3-grams shows log-likelihood of P_{lm} based on log_{10}. The right-most column of 1/2-grams represents the back-off weight α_{bo}. Note there is also an estimated entry for unknown words (`<unk>`) in 1-grams.

	Interrogatives					Declaratives
	TREC	CLEF	QC	YahooAns	All	Wikipedia
Sentence count	4950	13682	5452	581348	605432	30716301
Word count (K)	36.6	107.3	61.1	5773.8	5978.9	722845.6
Words/Sentence	7.41	7.94	11.20	9.93	9.88	23.53

Table 4.1.: 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.

the whole English Wikipedia to increase lexicon coverage and it produces a much larger language model. Finally we *prune* it for rapid access and *test* it for effectiveness.

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]

To increase n-gram coverage, the language model based on only questions is adapted with a plain-text English Wikipedia[8] corpus. The Wikipedia corpus is remarkably larger and more complex than the question corpus (see comparison in Table 4.1). But we do not want it to take too much probability footprint in the language model otherwise the model will not be discriminate on questions. Thus bigger weights were assigned to the question corpus during adaptation.

The following is the procedure of how the language model is built. The IRST Language Modeling Toolkit (Federico and Cettolo 2007) was used. Both the question and Wikipedia corpora are randomly divided: 90% for training and 10% for testing.

[4]Collected from `http://trec.nist.gov/data/qamain.html`
[5]Collected from `http://celct.isti.cnr.it/ResPubliQA/index.php?page=Pages/pastCampaigns.php`
[6]Downloaded from `http://l2r.cs.uiuc.edu/~cogcomp/Data/QA/QC/`
[7]Downloaded from `http://ir.mathcs.emory.edu/shared/`
[8]`http://meta.wikimedia.org/wiki/Wikipedia_Machine_Translation_Project`

LM	Training Set	Test Set	PP	PPwp	OOV	Size
questions	q.train	q.test	180.91	81.73	3.73%	36MB
adapted	(q+wiki).train	(q+wiki).test	219.23	18.03	0.57%	2.9GB
pruned	(q+wiki).train	(q+wiki).test	623.44	51.26	0.57%	148MB

Table 4.2.: Summary of the language models (all trigrams). "q.train" stands for the training set of the question corpus. OOV is Out-Of-Vocabulary rate. PPwp is the perplexity contributed by OOV. Size is the file size based on plain-text, rather than binary format.

1. Build a language model on the training set of the question corpus. The modified shift-beta (Federico and Bertoldi 2004) smoothing method was used for all language models.

2. Test the the question language model by perplexity on the test question set.

3. Adapt the question language model with the Wikipedia corpus. Minimum Discriminative Information Adaptation (Federico 1999) was used. This method is effective when domain related data is very little but enough to estimate a unigram. For the adaptation rate from 0 (no adaptation) to 1 (strong adaptation), a value of 0.8 was used.

4. Test the adapted language model by perplexity on the mixed test corpus of both questions and Wikipedia articles.

5. Prune the language model and test it again by perplexity. All occurrences with a probability of less than 10^{-6} were removed.

The summary of different language models produced above is shown in Table 4.2. The Out-Of-Vocabulary (OOV) rate is much larger for the question model than the adapted model. It is because the question corpus contains mostly random questions asked by community users. There are no correlations among questions and the vocabulary can be very different. Meanwhile, the Wikipedia corpus contains individual articles and the vocabulary in each article should be distributed averagely. Thus a random selection of 10% sentences (rather than articles) does not lead to high OOV rate.

Since the mixed corpus is much larger than the question corpus, the perplexity is also higher. By pruning, the size of the language model is much reduced without causing too much increase in perplexity: the size is reduced by 20 times and the perplexity goes up by 3 times.

4.4.4. Question Ranking

Since the LKB generator only employs a MaxEnt-based model for declarative sentences, the language model described previously is combined with the ranking from LKB to perform question ranking. In this subsection 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 $R = [r_1, r_2, \ldots r_N]$. $LP(r_i|\text{ME})$ is the MaxEnt score for the i-th realization under a Maximum Entropy model ME. Then we can normalize this score to a linear range of $[0, 1]$:

$$P(r_i|\text{ME}) = \frac{10^{LP(r_i|\text{ME})}}{Z} \tag{4.22}$$

Z is a normalization factor. It can be derived by:

$$\sum_{i=1}^{N} P(r_i|\text{ME}) = \sum_{i=1}^{N} \frac{10^{LP(r_i|\text{ME})}}{Z} = 1 \tag{4.23}$$

$$Z = \sum_{i=1}^{N} 10^{LP(r_i|\text{ME})} \tag{4.24}$$

Also, a single language model LM calculates the probability of each of the N different realizations by $P(r_i|\text{LM}) \in [0, 1]$, where $\sum_{i=1}^{N} P(r_i|\text{LM}) = 1$. With two rankings $P(r_i|\text{ME})$ and $P(r_i|\text{LM})$ we can borrow the idea of *F-measure* from information retrieval and combine them:

$$R(r_i) = F_\beta = (1 + \beta^2)\frac{P(r_i|\text{ME})P(r_i|\text{LM})}{\beta^2 P(r_i|\text{ME}) + P(r_i|\text{LM})} \tag{4.25}$$

With $\beta = 1$ the ranking is unbiased. With $\beta = 2$ the ranking weights $P(r_i|\text{LM})$ twice as much as $P(r_i|\text{ME})$.

Figure 4.14 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. In some cases the language model is not confident enough to discriminate between good and bad sentences. This can be shown from the score ranges. The MaxEnt scores range from 4.31 to 0.29. It has a stronger preference on the good sentences among different realizations than the language model, whose scores only range from 1.97 to 0.75. Thus we need to combine and take advantages of both of them. This issue is further discussed in Section 6.1.

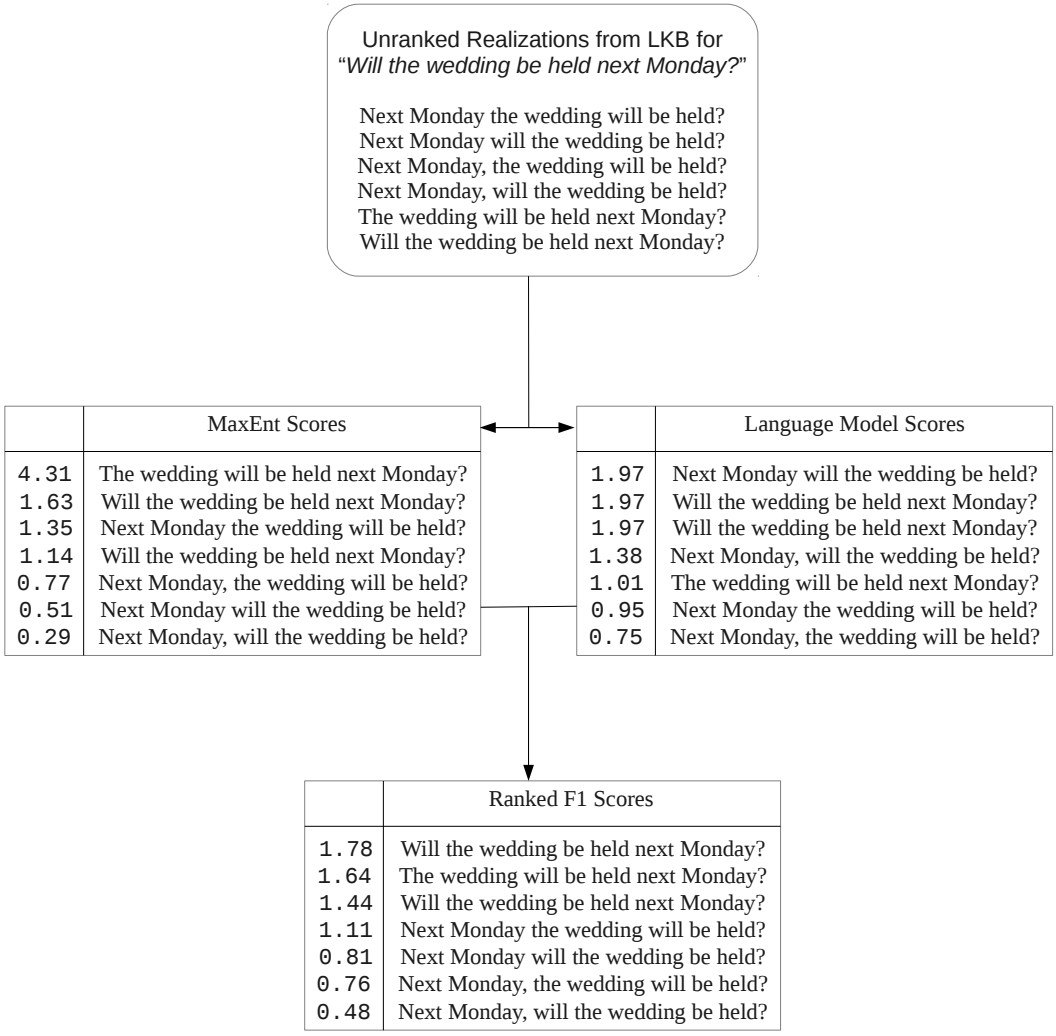


Figure 4.14.: Combined ranking using scores from a MaxEnt model and a language model. The final combined scores come from Equation 4.25 with $\beta = 1$. All scores are multiplied by 10 for better reading. Note that the question "`Will the wedding be held next Monday?`" appears twice and has different scores with the MaxEnt model. This is because the internal generation chart is different and thus it is regarded as different realizations even though the lexical wording is the same.

4.5. Robust Generation with Fallbacks

The primary reason that compromises robustness in both parsing and generation is the presence of unknown words. With the technology of *chart mapping* (Adolphs et al. 2008), PET is able to construct a parsing chart using the POS tags of unknown words, which are given by external POS taggers. As of the 1004 release of ERG, LKB is also able to generate from unknown words.

The secondary reason involves the grammar coverage of ERG and is more implicit. First of all, ERG does not tolerate ungrammatical sentences, which is not helpful in not well edited texts and spoken languages. Also, some rare cases of language uses are not covered by ERG, even though they are grammatical. In this case, PET simply does not parse. So a fallback strategy is needed.

The tertiary reason is due to errors introduced by MRS transformation. The modification of MRS structures must abide by ERG rules. However, errors are inevitable. In this case, LKB does not generate. So a fallback strategy is needed as well.

When deep processing does not work, it falls back on shallow processing. MrsQG employs a simple fallback strategy of replacing terms with question words. For instance, given a sentence "`John gave Mary a ring.`", assuming the `who` question on `Mary` cannot be generated, we apply the following fallback procedure:

1. Replace `Mary` with `who` to produce "`John gave who a ring`?".
2. Send the sentence to the parse-generation (PET-LKB) pipeline, hoping the wording can be re-ordered.
   a) In this case, LKB generates the following questions:
      - `To who did John give a ring?`
      - `To whom did John give a ring?`
      - `Who did John give a ring to?`
      - `Whom did John give a ring to?`
      - `Who did John give a ring?`
      - `Whom did John give a ring?`
   b) If LKB does not generate, take the original question as output.

In the worst case, if term extraction fails to recognize, chunks of phrases are replaced by the `what` question word.

The fallback strategy is only an effort to increase the quantity of questions by sacrificing quality. It is an optional choice depending on the need. But it does take effect as shown later in Subsection 5.3.2.

5. Evaluation

The evaluation of question generation with semantics was conducted along with the Question Generation Shared Task and Evaluation Challenge (QGSTEC2010[1]). It consists of three tasks:

- Task A: QG from Paragraphs. Questions are to be generated from a complete paragraph of text.
- Task B: QG from Sentences. Participants are given one complete sentence from which their system must generate questions.
- Task C: Open Task. Participants submit significant work on independent evaluations of Question Generation approaches.

Task B expresses the most basic requirement and functionality of question generation. MrsQG was also designed to fit this task.

In task B, participants are given a set of inputs consisting of an input sentence + question type and their system should generate two questions. Question types include yes/no, which, what, when, how many, where, why and who. Input sources are Wikipedia, OpenLearn[2] and Yahoo! Answers. Each source contributes 30 input sentences. With 3 sources and 2 questions per sentence, there will be 180 questions generated in total.

Below is an example of generating *when* and *which* questions.

(5.1) `The Council of Europe was set up in 1949.`

Target question type: WHEN, WHICH

Possible generated questions:

(a) (WHEN) `When was the Council of Europe set up?`

(b) (WHEN) `In what year was the Council of Europe set up?`

(c) (WHICH) `Which council was set up in 1949?`

(d) (WHICH) `In 1949, which council was set up?`

The following sections introduce evaluation criteria, participants' systems and results.

[1] `http://www.questiongeneration.org/QGSTEC2010`
[2] `http://openlearn.open.ac.uk`

5.1. Evaluation Criteria

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.

Rank	Description
1	The question is completely relevant to the input sentence.
2	The question relates mostly to the input sentence.
3	The question is only slightly related to the input sentence.
4	The question is totally unrelated to the input sentence.

2. Question type. Questions should be of the specified target question type.

Rank	Description
1	The question is of the target question type.
2	The type of the generated question and the target type are different.

3. Syntactic correctness and fluency. The syntactic correctness is rated to ensure systems can generate sensible output.

Rank	Description
1	The question is grammatically correct and idiomatic/natural.
2	The question is grammatically correct but does not read fluently.
3	There are some grammatical errors in the question.
4	The question is grammatically unacceptable.

4. Ambiguity. The question should make sense when asked more or less out of the blue.

Rank	Description
1	The question is unambiguous.
2	The question could provide more information.
3	The question is clearly ambiguous when asked out of the blue.

5. Variety. Pairs of questions in answer to a single input are evaluated on how different they are from each other.

Rank	Description
1	The two questions are different in content.
2	Both ask the same question, but with grammatical and/or lexical differences.
3	The two questions are identical.

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.

5.2. Participants Description

Four systems participated in Task B of QGSTEC2010:

- Lethbridge, University of Lethbridge, Canada
- MrsQG, Saarland University, Germany
- JUQGG, Jadavpur University, India
- WLV, University of Wolverhampton, UK

Lethbridge (Ali et al. 2010) employs a three-stage generation system. It first performs syntactic analysis of complex sentences and extracts simpler elementary ones. Then the named entities in elementary sentences are classified into five coarse types: human, entity, location, time and count. Finally based on 90 predefined rules, these types interact with each other and generate questions. For instance, given the following sentence:

(5.2) $\texttt{Tom}_{\text{human}}$ `ate an orange`$_{\text{entity}}$ `at 7 pm`$_{\text{time}}$`.`

With a rule "subject$_{\text{human}}$ object$_{\text{entity}}$ preposition$_{\text{time}}$", questions can be generated as:

(5.3) `Who ate an orange?`

`Who ate an orange at 7 pm?`

`What did Tom eat?`

`When did Tom eat an orange?`

JUQGG (Pal et al. 2010) employs a rule-based approach. The output from a Semantic Role Labeler (SRL) and a dependency parser jointly identifies question cue phrases. For instance, if a noun phrase is labeled as the first argument from SRL and tagged as a subject from the dependency parser, then it is a potential cue phrase for WHO questions. Then a question generator module replaces the cue phrase with question words and possibly reorder some chunks by some pre-defined rules. However, the details of rules used in this system are not very clear.

WLV (Varga and Ha 2010) has its origin from a multiple-choice question generation system (Mitkov et al. 2006). It mainly includes two steps: term extraction and syntactic rule transfer. Term extraction considers terms as all noun phrases matching a regular expression pattern of [AN]+N or [AN]*NP[AN]*N, which capture a large range of structural patterns according to Justeson and Katz (1995). If the head of term is recognized as a named entity, then a question of corresponding type can be asked. Depending on the grammatical function of the term, i.e. whether it is a subject or an object, corresponding syntactic transfer rules take effect. For instance, for WHO questions, the rule is simply "Who VO?" or "Who do/does/did SV?"

	Wikipedia	OpenLearn	YahooAnswers	All
sentence count	27	28	35	90
average length	22.11	20.43	15.97	19.20
standard deviation	6.46	5.26	7.16	6.93
question count	120	120	120	360

Table 5.1.: Average length and standard deviation of sentences in test set.

The rule settings for WLV seem to be too strict: in a test on the development data, it generates 115 questions out of the required 180 questions. The generation failures were due to two reasons: YES/NO questions were not generated because WLV was not modified for that and WLV also was not able to generate from complicated sentences. These generation failures, reported from its original system description, also appeared in the final evaluation, which will be shown in the next section.

5.3. Evaluation Results

5.3.1. Test Set

The test set contains 90 sentences from Wikipedia, OpenLearn and YahooAnswers. The genres of domains are quite diverse. 360 questions in all were asked on these 90 sentences. Table 5.1 gives this statistics.

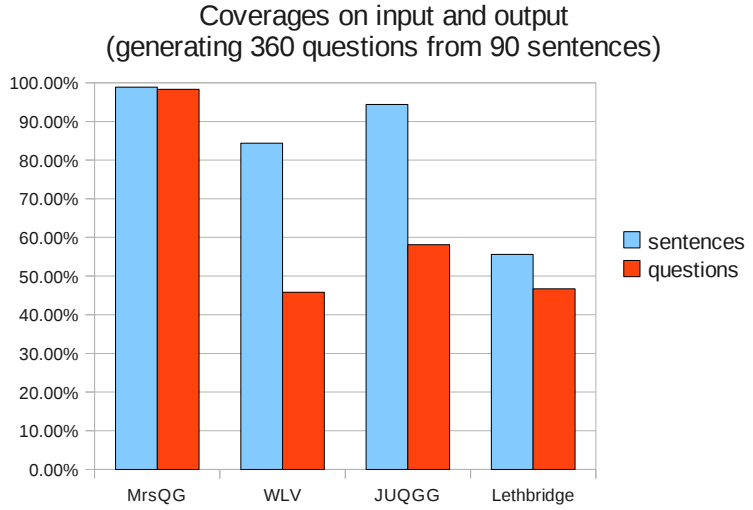
5.3.2. Generation Coverage

The low coverage number in most cases of Table 5.2 shows that generation from all input sentences and from all required question types was not an easy task. None of the systems reaches 100%. Among them, MrsQG, WLV and JUQGG generate 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 5.2b in Table 5.2 illustrates this.

Good coverage on required questions usually depends on whether the employed named entity recognizer is able to identity corresponding terms for a question type and whether the reproduction 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 the coverage. The next subsection shows this point.

	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 5.2.: 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.

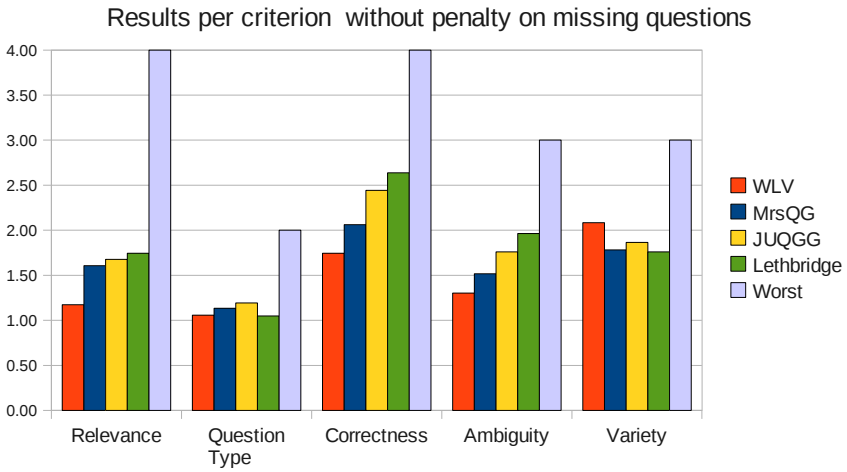
5.3.3. Overall Evaluation Grades

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 5.2), the final grades are calculated with and without penalty on missing questions.

Table 5.3 present the results without penalty on missing questions. Grades were calculated based only on *generated* questions from each system. Out of all grading criteria, all systems did the worst in the SYNTACTIC CORRECTNESS AND FLUENCY of generated questions, largely due to the fact that rule-based syntactic transformation does not guarantee grammaticality. Except for question type, all systems did the best job in RELEVANCE. This result is not surprising since the task is defined as generating questions from input sentences, the questions generated from the sentence should be relevant with the original one.

	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 5.3.: Results per participant without penalty for missing questions. Lower grades are better. Agreement closer to 100% indicates better inter-rater reliability.

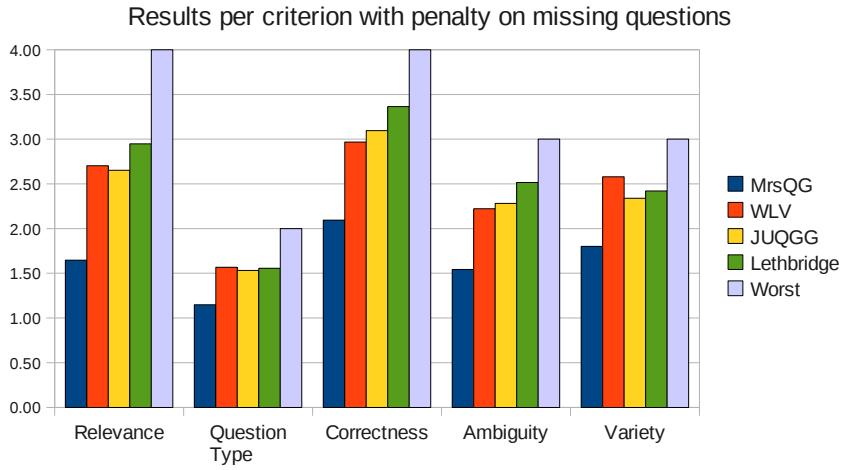
Note that the best ranking system WLV from Table 5.3b did the worst in VARIETY. This result corresponds to WLV's coverage in Table 5.2b. WLV seems to be designed as a high-precision but low-recall system.

If taking all questions that should be generated into account, then Table 5.4 show the other aspect of the evaluation. 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 a relatively lower coverage on generated questions (between 40% ~ 60%), their scores are much decreased by a large margin as compared to MrsQG, which has nearly 99% coverage on required questions.

The inter-rater agreement shown by Table 5.3 is not satisfactory. A number of over 80% usually indicates good agreement but only QUESTION TYPE has achieved this standard. Landis and Koch (1977) has 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.

	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 5.4.: Results per participant with penalty for missing questions. Lower grades are better.

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.

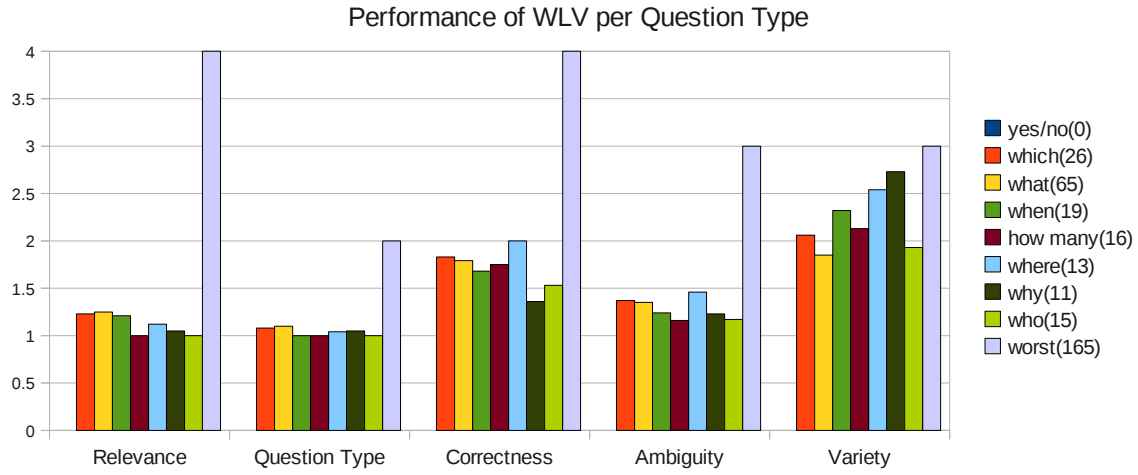
5.3.4. Evaluation Grades per Question Type

QGSTEC2010 requires eight types of questions: yes/no, which, what, when, how many, where, why and who. Among them, why questions involve more reasoning than others. However, it is not the case that why questions receive the worst scores for most of the systems, as shown in Figure 5.1. Also, yes/no questions could have been the easiest ones since it is only a matter of auxiliary fronting in the most simple case. But it is not the case either that yes/no questions receive the best scores for most of the systems.

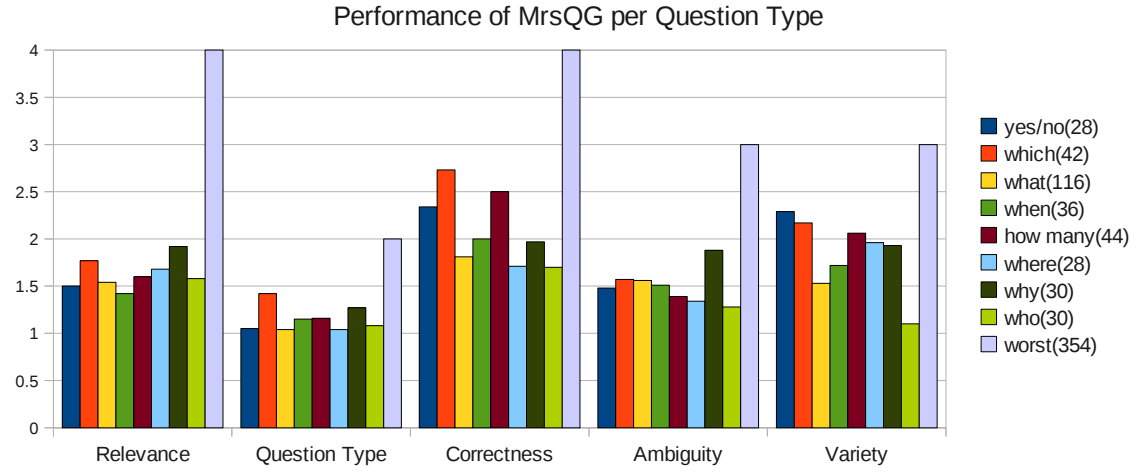
Generally speaking, WLV and JUQGG did the worst for where questions, MrsQG and Lethbridge did the worst for which questions. It is hard to tell the reasons for other systems. But for MrsQG the quality of which questions depends on the term extraction component. If a named entity recognizer fails or the ontology cannot provide a hypernym

for a term, a which question cannot be properly generated.

Analysis for evaluation grades per dataset shows no obvious difference of system performances on different data sources. Thus it is not shown in this thesis.

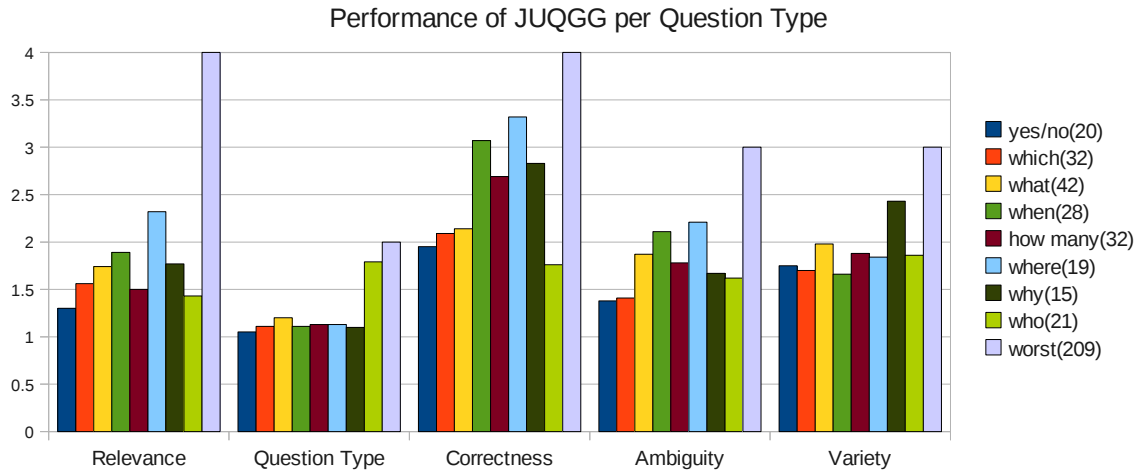


(a) WLV

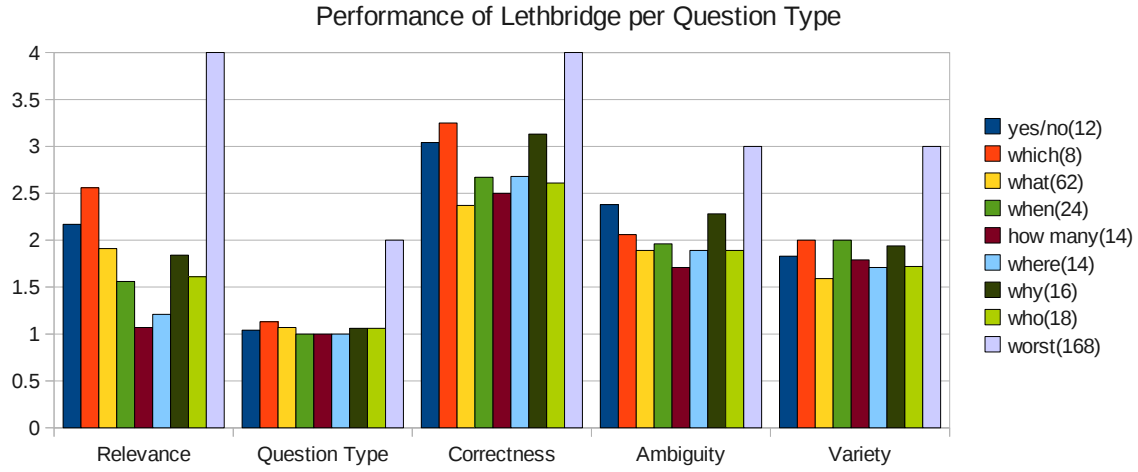


(b) MrsQG

Figure 5.1.: (continued in the next page)



(c) JUQGG



(d) Lethbridge

Figure 5.1.: Performances of 4 systems 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.

6. Discussion

This chapter discusses various aspects in question generation with semantics. Some of the discussion is based from 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. Deep Parsing can be Precise

As the backbone of MrsQG, the English Resource Grammar encodes more information than other shallow processing techniques, such as a dependency grammar or a PCFG grammar. It provides a more precise reading of a sentence, which can be effectively used by following components. For instance, the *_subord_rel pattern identifies subordinate clause conjunctions. To illustrate this, we re-use Example 4.8 here:

(6.1) `Given that Bart chases dogs, Bart is a brave cat.`

The subordinate conjunction ("`given that`") is marked by the elementary predication _given+that_x_subord_rel (see Figure 4.8 for details). This triggers a subordinate clause decomposer to function. However, a PCFG grammar gives a weak reading for the same sentence (Figure 6.1). Firstly, it breaks the semantically integrated conjunction "`given that`" into two constituents. Each one is assigned a different POS tag (`given`←VBN, `that`←IN) and does not give a meaningful and standalone reading. Secondly, the subordinate clause "`Bart chases dogs`" is only marked as an S but not explicitly specified as a subordinate clause. Thus this POS tag S can either indicate a relative clause or a subordinate clause but shallow processing does not tell the distinction.

Another example that shows the advantages of deep processing involves the precise description of noun-noun compounds:

(6.2) (a) (`Queen Elizabeth`)$_{\text{compound\_name}}$ `is wise.`

(b) (Mineral water)$_{\text{compound}}$ `is healthy.`

Desired questions:

(c) `Which Queen is wise?`

(d) `What water is healthy?`

In (a) the hypernym is the first word so as that `Queen Elizabeth` is a `Queen`, based on which a question (c) can be asked. However, in (b) the hypernym changes to be the second word so as that Mineral water is `water`, based on which a question (d) can be asked. With a CFG grammar, this distinction of hypernyms with different lexical

positions can not be shown since noun-noun compounds only follow a NP → NN(P) NN(P) rule without more fine-grained information, as in Figure 6.2(a)(b).

The ERG, on the other hand, clearly tells this difference by using different types of EPs. For noun-noun compounds containing proper names, compound_name is used. It has the first word, or its ARG2, as its hypernym. Thus Hurricane Katrina is a hurricane and Agent Smith is an agent. For noun-noun compounds not containing proper names, a different EP, compound, is used. It has the second word, or its ARG1, as its hypernym. Thus a wedding ring is a ring and chocolate milk is milk. Figure 6.2(c) illustrates this.

Note that however ERG also has its limitations: an apposition relation given by ERG cannot tell whether ARG1 or ARG2 is the hypernym. For instance, in "$(\texttt{Elizabeth})_{\text{ARG1}}$ $(\texttt{the Queen})_{\text{ARG2}}$" ARG2 is the hypernym but in "$(\texttt{the Queen})_{\text{ARG1}}$ $(\texttt{Elizabeth})_{\text{ARG2}}$" ARG1 is the hypernym. Apposition resolution itself is a research question and cannot be simply solved by a deep precision grammar.

By precisely identifying the grammatical function of different constituents in a sentence, deep processing makes some tasks of question generation that require more fine-grained structural information possible. However, this powerful technique must be used with proper preprocessing. For instance, ERG mistakenly recognizes a person name such as "`John Smith`" as apposition. Then it would sound bizarre to ask a question "`Which John likes Mary?`" or even "`Which Smith likes Mary?`" from "`John Smith likes Mary.`". By employing a named entity recognizer and tagging the whole "`John Smith`" as a proper noun (NNP), it prevents apposition in pure person names. Thus a more natural question "`Who likes Mary?`" can be asked. This hybrid processing technique of combining both external tagging and deep processing is fully addressed in Adolphs et al. (2008).

6.2. Generation with Semantics can Produce Better Sentences

One semantic representation can correspond to multiple syntactic variations. With a good ranking mechanism the best realization can be selected, which makes the generation even better than the original one in some cases. Take the English active and passive voice as an example:

(6.3) (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 can only generate questions as best as (b). A semantics-based system, generating from chase(who, the dog), can produce both the passive form

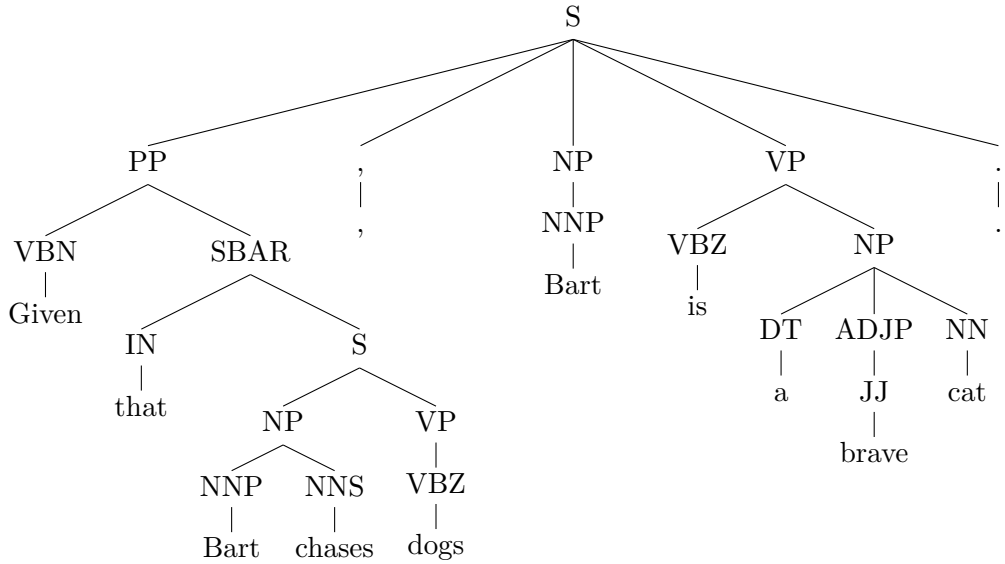


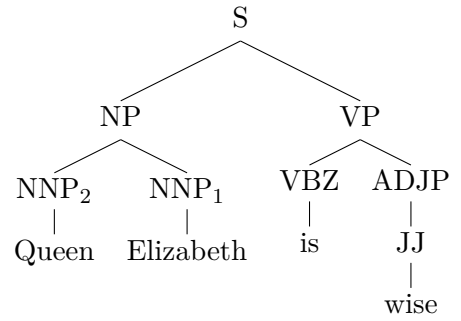
Figure 6.1.: A Context Free Grammar's reading of the sentence "`Given that Bart chases dogs, Bart is a brave cat`". The subordinate conjunction "`given that`" cannot be properly marked. The subordinate clause "`Bart chases dogs`" also cannot be indicated explicitly.

in (b) and the active form in (c). Usually (c) is preferred by most English speakers since the English language prefers active verbs in most cases, (c) is also shorter than (b) or it just simply sounds more natural. However, the case shown in this example might not be common and thus does not expose the benefits of a semantics-based system easily.

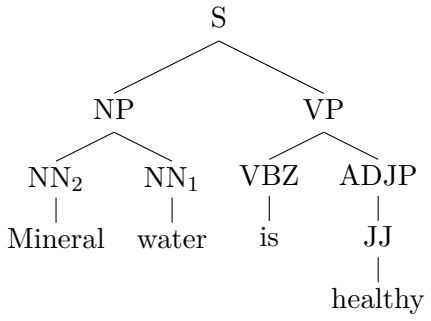
6.3. Interface to Lexical Semantics Resources

Although one semantic representation can produce multiple syntactic variations, this procedure is handled internally through chart generation with ERG. Also, different syntactic variations 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 able to focus on the semantics level 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.

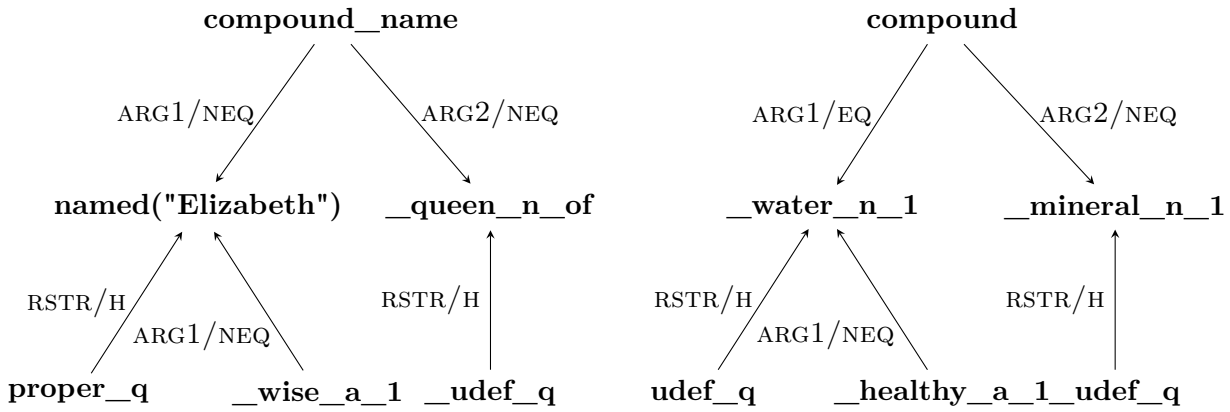
The potential of question generation to employ semantic relations is not restricted



(a) NNP$_2$ is a hypernym of NP



(b) NN$_1$ is a hypernym of NP



(c) DMRS graph of `Queen Elizabeth is wise.`"(left) and "`Mineral water is healthy.`" (right).

Figure 6.2.: Comparison of treatment of compounding in a Context Free Grammar and the English Resource Grammar. In (a) and (b) there is no way to tell which one is the hypernym of the noun phrase. However, in (c) they are clearly distinguished: on the left side, compound_name has its ARG2 as the hypernym (corresponding to that NNP$_2$ is a hypernym of NP in (a)); on the right side, compound has its ARG1 as the hypernym (corresponding to that NN$_1$ is a hypernym of NP in (b)).

to employing hypernym-hyponym[1] relations to produce which questions. Holonym-meronym[2] 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 per-

[1] 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.
[2] 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.

formed 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 totally nonsense questions. So expanding the range of questions with lexical semantics resources should be used with caution.

6.4. Ranking *vs.* Reranking

Section 4.4 has proposed a method of performing question ranking to select the best questions. Figure 4.14 shows that this selection runs in *parallel*: the two scores from the MaxEnt model and language models are combined together to give an overall score. An alternative way is to do this in *serial*: first select the top N output from the MaxEnt model and then use the language model to rerank the questions. Thus the final result is directly decided by the language model. The parallel method can be called hybrid ranking and the serial method can be called reranking.

MrsQG has chosen the hybrid ranking method instead of reranking largely based on that the MaxEnt model is more discriminate on different realizations and quite stable. Taking a close look at the scores in Figure 4.14, MaxEnt scores have a wider range of $[0.29, 4.31]$ (already been converted into linear scope) while language models scores range only between $[0.75, 1.97]$. With the best score as 4.31 while the worst score as 0.29 the MaxEnt model shows a strong preference towards better sentences. Also, the features used by the MaxEnt model involve syntactic structures from derivation trees as well as language model lexicons. It is more robust against uncommon or even unknown words than the language model. For instance, the language model gives the same top score to three questions in Figure 4.14. A serial reranking method fails to select the best one in this case.

6.5. Language Independence and Domain Adaptability

MrsQG aims to stay language-neutral based on a semantics transformation of sentences. In principle, it needs little modification to adapt to other languages, as long as there is a grammar[3] conforming with the HPSG structure and lkb. However, the experience on multi-lingual grammar engineering has shown that although MRS offers a higher level of abstraction from the 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 needs to be adapted for the specific grammars, similar to the situations in MRS-based machine translation (Oepen et al. (2004)).

The domain adaptability is confined in the following parts:

1. Named entity recognizers. For a different domain, the recognizers must be retrained. MrsQG also uses an ontology named entity recognizer. Thus collections

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

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.

6.6. Limitations of Proposed Method

A semantics-based question generation system is theoretically sound and intuitive. But the implementation is limited to currently available tools, such as the grammar, the parser, the generator and the preprocessors. The central theme of MrsQG is MRS, which is an abstract layer from the ERG. The parser and generator cannot work without the ERG as well. 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.

From 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 on 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.

From the generation side, theoretically all parsed MRS structures should generate. But there exists the problem of overgeneration. As shown in Table 2.1, the MaxEnt model has 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.

From 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.

From the theoretical point of view, the theory underlying MrsQG is Dependency Minimal Recursion Semantics, a derivation of MRS. DMRS provides a connecting interface between a semantic language and a dependency structure. Although still under development, it has shown a success in the application of question generation via MrsQG. However, there are still redundancies in DMRS (such as the non-directional NULL/EQ label in Figure 4.5). Some of the redundancies are even crucial: a mis-manipulation of the MRS structure inevitably leads to generation failure thus makes the MRS transformation process fragile. Fixing this issue is not trivial however, which requires the joint effort of the MRS theory, the ERG grammar and the LKB generator.

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

7. Conclusion and Future Work

This thesis introduces the task of question generation and proposes a semantics-based method to perform this task. Generating questions from the semantic level of languages is intuitive but also has its difficulties, namely sentence simplification, question transformation and question ranking. In correspondence to these difficulties, this thesis proposes three methods: 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 theory 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 thesis is MRS decomposition and transfer. A generic decomposition algorithm is developed to perform sentence simplification while this algorithm boils down to solving a graph traversal problem following the labels of edges which encode linguistic properties.

Evaluation results reveal some of the fine points of this semantics-based method but 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 targets 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, most development effort and result are still in a preliminary stage. Combining question generation with specific application requirement has been put into the long-term schedule.

To sum up, the method proposed by this thesis produces so far the first known and 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 depen-

dency 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.

A. QGSTEC2010 Test Set

The following 3 tables lists the sentences from OpenLearn, Wikipedia and Yahoo Answers. The types of questions asked on each sentence are also listed. Detailed statistics of these sentences can be found in Table 5.1.

Table A.1.: Sentences from Wikipedia

#	**Sentences**	**Question Types**
1	In 1980, the son of Vincent J. McMahon, Vincent Kennedy McMahon, founded Titan Sports, Inc. and in 1982 purchased Capitol Wrestling Corporation from his father.	when who which
2	The first CCTV system was installed by Siemens AG at Test Stand VII in Peenemünde, Germany in 1942, for observing the launch of V-2 rockets.	when where what why
3	The first plan to harness the Shannon's power between Lough Derg and Limerick was published in 1844 by Sir Robert Kane.	when who where which
4	The IEEE consists of 38 societies, organized around specialized technical fields, with more than 300 local organizations that hold regular meetings.	how many
5	The motto of Stanford University, selected by President Jordan, is "Die Luft der Freiheit weht."	what who which
6	Because, by definition, there are no written records from human prehistory, dating of prehistoric materials is particularly crucial to the enterprise.	why what
7	The major agricultural products can be broadly grouped into foods, fibers, fuels, and raw materials.	which
8	The line was derived from cervical cancer cells taken from a patient named Henrietta Lacks, who eventually died of her cancer on October 4, 1951.	when who what
9	Although 99.9% of human DNA sequences are the same in every person, enough of the DNA is different to distinguish one individual from another.	how many yes/no
10	To ensure that there would never be any more confusion over who would handle homicides at the federal level, Congress passed a law that put investigations of deaths of federal officials within FBI jurisdiction.	why what

11	Lemurs arrived in Madagascar approximately 62 to 65 mya by rafting on mats of vegetation at a time when ocean currents favored oceanic dispersal to the island.	what when where
12	The main cause of variation is mutation, which changes the sequence of a gene.	what
13	The sovereign state of Brunei, located on the north coast, comprises about 1% of Borneo's land mass.	where how many
14	The language, initially called Oak after an oak tree that stood outside Gosling's office, also went by the name Green and ended up later renamed as Java, from a list of random words.	which yes/no
15	A buffer overflow occurs when data written to a buffer, due to insufficient bounds checking, corrupts data values in memory addresses adjacent to the allocated buffer.	when which what why
16	The strength of the depletion zone electric field increases as the reverse-bias voltage increases.	yes/no what
17	Soon after Locke's birth, the family moved to the market town of Pensford, about seven miles south of Bristol, where Locke grew up in a rural Tudor house in Belluton.	when where
18	Many Somerset soldiers died during the First World War, with the Somerset Light Infantry suffering nearly 5,000 casualties.	how many which
19	The Tigris unites with the Euphrates near Basra, and from this junction to the Persian Gulf the mass of moving water is known as the Shatt-al-Arab.	which what
20	According to the Biblical book of Daniel, at a young age Daniel was carried off to Babylon where he was trained in the service of the court under the authority of Ashpenaz.	which where who
21	Rescue teams located wreckage near the community of Grammatiko 40 km (25 miles) from Athens.	where how many
22	Similarly, as elevation increases there is less overlying atmospheric mass, so that pressure decreases with increasing elevation.	why
23	Glacier ice is the largest reservoir of fresh water on Earth, and is second only to oceans as the largest reservoir of total water.	yes/no what
24	James Maury "Jim" Henson (September 24, 1936 - May 16, 1990) was one of the most widely known puppeteers in American history and was the creator of The Muppets.	who what how many
25	Over 80% of the city was destroyed by the earthquake and fire.	how many what

26	Since light is an oscillation it is not affected by travelling through static electric or magnetic fields in a linear medium such as a vacuum.	yes/no
27	Stone tools were used by proto-humans at least 2.5 million years ago.	when

Table A.2.: Sentences from OpenLearn

#	**Sentences**	**Question Types**
1	The view that prevailed was that there should be a fitting public memorial to the 49,076 gunners who died.	how many
2	At some very basic level, there are three ways in which businesses can respond to the green imperative.	what how many
3	Two sources of fuel are particularly important to ensure a healthy and efficiently functioning brain - oxygen and water.	what how many
4	In 1996, the trust employed over 7,000 staff and managed another six sites in Leeds and the surrounding area.	where when how many
5	Being able to link computers into networks has enormously boosted their capabilities.	what
6	The codes of practice apply to a range of social care professionals (including, for example, residential care workers) and are not specific to social work.	yes/no what
7	The decimal representation of rational numbers has the advantage that it enables us to decide immediately which of two distinct positive rationals is the greater.	why
8	As you saw earlier, phrasal verbs tend to be more common in speech than in writing as they are less formal.	why what
9	The number of obese children has doubled since 1982, 10% of six year olds and 17% of fifteen year olds are now classified as obese.	when what how many
10	Early in the twentieth century, Thorstein Veblen, an American institutional economist, analysed cultural influences on consumption.	where who when what
11	Kate was put off by the word 'paradox' and Erin did not know what 'marginal tax' meant.	who which yes/no
12	Alexander Graham Bell, who had risen to prominence through his invention of the telephone, took a great interest in recording sounds, even suggesting to Edison that they might collaborate.	who what why
13	Nash began work on the designs in 1815, and the Pavilion was completed in 1823.	who what when
14	Rashid (1998) points out that in markets where entry barriers are low, quality has the greatest tendency to vary.	who which what
15	Repetitive play can also be a dilemma, in that adults are uncertain about when, or indeed whether, they should intervene to move the child on.	why
16	If BAC levels increase to around 200 mg/100 ml then speech becomes slurred and coordination of movement is impaired.	what how many

17	The passing of the First World War soon saw the establishment of many national radio broadcasting organisations, the BBC being formed in 1922.	which when
18	There are seven wooden pieces, which can be assembled to form a solid cube.	what how many
19	Air contains several different gases but about twenty per cent of it is oxygen.	what how many
20	Some of Britain's most dramatic scenery is to be found in the Scottish Highlands.	where
21	Among many others, the UK, Spain, Italy and France are unitary states, while Germany, Canada, the United States of America, Switzerland and India are federal states.	yes/no which
22	Designer Ian Callum, originally from Dumfries in Scotland, studied at the Glasgow School of Art and at the Royal College of Art in London.	where who what which
23	It is agreed that the Enlightenment was at the height of its influence in the 1760s and early 1770s.	when what
24	Motivation theories seek to explain why we do the things we do either by examining how a behaviour satisfies our 'needs' or the processes we go through as we decide how to achieve our goals.	what
25	Furthermore, Softbank BB's president, Masayoshi Son, announced that he and other senior executives would take a 50 per cent pay cut for the next six months.	who what which how many
26	One of the earliest popular applications of the Web was an online auction system called eBay.	what yes/no
27	The number system which we all use in everyday life is called the denary representation, or sometimes the decimal representation, of numbers.	what
28	It is also possible to buy circular protractors that measure angles from 0° to 360°.	what

Table A.3.: Sentences from Yahoo Answers

#	**Sentences**	**Question Types**
1	Having a teddy bear at eight years old is completely normal.	yes/no what how many
2	There are four 10's in each standard deck of 52 cards.	how many
3	Each player takes 6 dominoes and places them so that the other players can't see their value.	what how many
4	Librations have allowed us to see about 60% of the surface.	what how many
5	The axiom in Euclidian Geometry is that parallel lines never intersect.	what
6	The reason Polaris is important to us is that it is very close to the north celestial pole, so it gives us an easy way to find north by looking at the stars.	why which
7	The whole bogus 2012 nonsense is based on the fact that the ancient Mayan ceremonial calendar will run out in 2012.	what when
8	Some research was done on non-migratory birds in the UK and Germany and it was found that the same species had different songs in the two countries.	where what
9	The giant tortoise lives the longest, about 177 years in captivity.	which how many
10	Egypt is part of the Continent of Africa, and is situated in the north eastern part of Africa.	what where
11	lol means laugh out loud.	what
12	People use this site to learn English.	why what
13	Boxing Day - December 26th - is the feastday of Saint Stephen, and is called St Stephen's Day.	which when what
14	Friday was long considered an unlucky day, especially by sailors, so when 13 and Friday coincide it was thought to be a double dose of bad luck.	which what
15	A UFO is an Unidentified Flying Object, and therefore by definition... Unidentified what	
16	In Philosopher's/Sorcerers Stone, information on Dean Thomas was left in the US version, but not the British.	yes/no who what
17	For the sake of historical accracy, let me state that it is useful to talk of slavery everywhere, rather than black slavery in the United States.	why
18	BMW does mean Bavarian Motor Works.	what
19	The "Rising Sun" owned by Larry Ellison is up for sale.	who what
20	You shouldn't really pour warm or hot water on cold glass, cos it'll crack.	why what

21	In the USA, you get your permit when you are 15, meaning you have to be with a licenced driver in the car over the age of 25 I think.	where how many
22	Women tend to cover shorter urban journeys and therefore their driving is slower and accidents tend to be relatively minor.	why what
23	Plug it in most computers will do an automatic set up.	yes/no what
24	Yes, PS2 plays regular DVD-Video disc.	what yes/no
25	Take it to the place you purchased it an let them recycle it.	where
26	I'm pretty sure it will depend on your service provider, but i know that T Mobile charges for 0800 calls.	which what
27	I won't be voting in the May local elections because in our constituency there are no elections.	when
28	Vinegar or bicarbonate of soda are the most environmentally friendly.	what
29	If as you say your dog will eat anything train it to eat the snails.	what
30	As with everything, the right kind of person can be influenced by almost anything.	what
31	A lot of very productive people get sick and/or die because of smoking related diseases.	why which
32	Another 20,000 will only bring troop levels back to where they were in 2005.	how many when
33	If scroll lock is enabled on the keyboard when you press any of the arrow keys the screen will move in that direction but the selected cell will not change.	what
34	One of the best sites for computer parts is Ebuyer.	which what
35	Therefore, if you piled your ton of bricks onto a weighing scale, the pile would be smaller than the feathers.	yes/no

Bibliography

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.

James Allen. *Natural Language Understanding (2nd ed.)*. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1995.

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.

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 and B. Srinivas. Automatic induction of rules for text simplification. *Knowledge Based Systems*, 10(3):183–190, 1997.

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.

M. Collins and T. Koo. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70, 2005.

A. Copestake. Robust Minimal Recursion Semantics. `http://www.cl.cam.ac.uk/~aac10/papers/rmrsdraft.pdf`, 2004.

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.

M. Federico and N. Bertoldi. Broadcast news LM adaptation over time. *Computer Speech & Language*, 18(4):417–435, 2004.

Marcello Federico. Efficient Language Model Adaptation Through MDI Estimation. In *In Proc. of EUROSPEECH*, pages 1583–1586, 1999.

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.

D. Flickinger. The English Resource Grammar. Technical report, LOGON Technical Report # 2007-7, 2007.

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

Dan Flickinger, Jan Tore Lønning, Helge Dyvik, and Stephan Oepen. SEM-I rational MT. Enriching deep grammars with a semantic interface for scalable machine translation. In *In Proceedings of the 10th Machine Translation Summit*, pages 165–172, 2005.

G. E Heidorn. Intelligence Writing Assistance. In R. Dale, H. Moisl, and H. Somers, editors, *A Handbook of Natural Language Processing: Techniques and Applications for the Processing of Language as Text*, pages 181–207. Marcel Dekker, New York, 1998 (published in August 2000), 2000.

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.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2nd Edition)*. Prentice Hall, 2 edition, May 2008.

J. Justeson and S. Katz. Technical terminology: some linguistic properties and an algorithm for identification in text. *Natural Language Engineering*, pages 9–27, 1995.

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

Dan Klein and Christopher D. Manning. Fast Exact Inference with a Factored Model for Natural Language Parsing. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 3–10, Cambridge, MA, 2003. MIT Press.

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

Irene Langkilde and Kevin Knight. Generation that exploits corpus-based statistical knowledge. In *ACL-36: Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 704–710, Morristown, NJ, USA, 1998.

R. Levy and G. Andrew. Tregex and Tsurgeon: tools for querying and manipulating tree data structures. In *5th International Conference on Language Resources and Evaluation (LREC 2006)*, 2006.

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.

Ruslan Mitkov, Le An Ha, and Nikiforos Karamanis. A computer-aided environment for generating multiple-choice test items. *Nat. Lang. Eng.*, 12(2):177–194, 2006.

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.

Hiroko Nakanishi, Yusuke Miyao, and Jun'ichi Tsujii. Probabilistic models for disambiguation of an HPSG-based chart generator. In *Parsing '05: Proceedings of the Ninth International Workshop on Parsing Technology*, pages 93–102, Morristown, NJ, USA, 2005.

Gertjan Noord and Robert Malouf. Wide coverage parsing with stochastic attribute value grammars. In *In Proceedings of the IJCNLP workshop Beyond Shallow Analysis*, 2004.

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. QGSTEC 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.

Ehud Reiter and Robert Dale. Building applied natural language generation systems. *Nat. Lang. Eng.*, 3(1):57–87, 1997.

V. Rus and A. Graesser, editors. *The Question Generation Shared Task and Evaluation Challenge*. 2009.

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.

Yves Schabes. Stochastic lexicalized tree-adjoining grammars. In *Proceedings of the 14th conference on Computational linguistics*, pages 425–432, Morristown, NJ, USA, 1992.

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

Advaith Siddharthan. *Syntactic Simplification and Text Cohesion*. PhD thesis, University of Cambridge, 2003.

Natalia Silveira. Towards a Framework for Question Generation. In *Proceedings of the first Workshop on the Question Generation Shared Task and Evaluation Challenge*, Arlington, VA: NSF, 2008.

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.

Erik Velldal and Stephan Oepen. Maximum Entropy Models for Realization Ranking. In *Proceedings of MT-Summit X*, Phuket, Thailand, 2005.

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

M.A. Walker, O. Rambow, and M. Rogati. Spot: A trainable sentence planner. In *Second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies 2001*, page 8, 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.

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, 2007a.

Yi Zhang, Stephan Oepen, and John Carroll. Efficiency in unification-based N-best parsing. In *IWPT '07: Proceedings of the 10th International Conference on Parsing Technologies*, pages 48–59, Morristown, NJ, USA, 2007b.