UNIVERSITY OF TWENTE.

**Faculty of Electrical Engineering,
Mathematics & Computer Science**

Follow-up Question Generation

Yani Mandasari

**M.Sc. Thesis
August 2019**

Thesis committee:
Dr. Mariët Theune
Prof. Dr. Dirk Heylen
Jelte van Waterschoot, M.Sc

Interaction Technology
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
Enschede, The Netherlands

Abstract

In this thesis, we address the challenge of automatically generating follow-up questions from the users' input for an open-domain dialogue agent. Specifically, we consider that follow-up questions associated with those that follow up on the topic mentioned in the previous turn. Questions are generated by utilizing the named entity, part of speech information, and the predicate-argument structures of the sentences. The generated questions were evaluated twice, and after that, a user study using an interactive one-turn dialogue was conducted. In the user study, the questions were ranked based on the average score from the question evaluation results. The user study results revealed that the follow-up questions were felt convincing and natural, especially when they were short and straightforward. However, there are still many rooms for improvements. The generated questions are very dependent to the correctness of the sentence structure and difficult to expand the conversation topics since the conversations are very related to the input from users.

keywords: open-domain dialogue agents, question generation, follow-up questions.

Acknowledgements

Alhamdulillaah. I want to express my gratitude to Mariët Theune for the patience, guidance, and feedback throughout my thesis work. Next to that, I want to thank Jelte van Waterschoot for giving me the idea for this thesis topic, discussions, and for sending helpful journals and websites so that I could understand something better. I would also like to thank Dirk Heylen for his feedback during the writing of the report.

Furthermore, I want to thank the Ministry of Communication and Informatics of Indonesia for granting a scholarship and allowing me to study Interaction Technology in the University of Twente. And finally, thanks to my family and friends for their continuous support throughout my study in the Netherlands.

Contents

List of Figures

List of Tables

Abbreviations

AI	**A**rtificial **I**ntelligence
NER	**N**amed **E**ntity **R**ecognition
NLG	**N**atural **L**anguage **G**eneration
NLP	**N**atural **L**anguage **P**rocessing
NLTK	**N**atural **L**anguage **T**ool**k**it
NLU	**N**atural **L**anguage **U**nderstanding
POS	**P**art **o**f **s**peech
QA	**Q**uestion **A**nswering
QG	**Q**uestion **G**eneration
SRL	**S**emantic **R**ole **L**abeling

Chapter 1

Introduction

1.1 Background

Humans are social beings. We talk to each other and engage in a conversation to establish social ties. Examples include dialogues when people meet, recurring talk between workers on topics not related to the job, or social networks through gossip. These types of dialogue not only important in social settings but also establishing and maintaining collaborative relationship [3].

Dialogue agents or conversational systems are computer programs designed to interact with users similarly as a human being would do. Gao et al. [10] give an example that depicts the kind of problems a dialogue agent is expected to handle, as shown in Table 1.1.

1. **question answering (QA):** the agents need to provide brief and direct answers to user queries based on knowledge drawn from various data sources, such as sales and marketing datasets or text collections from web documents (e.g. Turns 3 to 5 in Table 1.1)
2. **task-oriented:** the agents need to accomplish user commands, such as restaurant reservation or meeting scheduling (e.g. Turns 6 to 7 in Table 1.1)
3. **social chatbots:** the agents need to converse seamlessly and appropriately like a human (e.g. Turns 1 to 2 in Table 1.1)

Table 1.1: An illustration of a human-agent dialogue during the process of making a business decision (*usr:* user, *agt:* agent) from [10]

Turn	**Dialogue**
1	*usr:* Good morning!
	agt: Good morning, I hope you had a great weekend.
2	*usr:* not really.
	agt: You sound depressed. What can I do for you?
3	*usr:* I'd like to know where sales are lagging behind our forecast?
	agt: The worst region is [country], where sales are 15% below projections.
4	*usr:* Do you know why?
	agt: The forecast for [product] growth was overly optimistic.
5	*usr:* How can we turn this around?
	agt: Here are the 10 customers in [country] with the most growth potential, per our CRM model.
6	*usr:* Can you set up a meeting with the CTO of [company]?
	agt: Yes, I've set up a meeting with [person name] for next month when you are in [location].
7	*usr:* Thanks!

In general, dialogue agents can be grouped into two categories, task-oriented and non-task-oriented (also known as chatbots). Most of today's popular personal assistants are task-oriented agents. Examples include Apple Siri, Google Home, and Amazon Alexa. They are capable of answering questions and handling relatively simple tasks such as music playback, providing weather, traffic, sports, and other real-time information such as news.

Chatbots, on the other hand, usually have a primary aim to be AI companions to humans with an emotional connection instead of completing specific tasks [10]. Though human-computer dialogues may be different in some aspects from human-human conversations, the design of human-computer dialogue will benefit a lot from the study of human-human dialogues. It is necessary to build a dialogue agent that can respond to user utterances appropriately. When agents do not respond to the user's utterances properly, users will perceive that dialogue agents do not have an adequate capacity to maintain a conversation. For example, the participants of the study by [6] expressed no desire to build bonds with their voice-based assistant even though they found conversational features were useful. According to [6], the lack of enthusiasm for bonding may stem from the core belief that agents are poor dialogue partners that should be obedient to the users. Another reason lies in the perception that there is no support for social dialogue in the current infrastructure for conversation.

Dialogue agents can be inspired by human-human conversation even though they do not necessarily need to resemble it [6]. One of important social feature in human conversations suggest by [6] is active listenership. Paying attention, demonstrating engagement, and a willingness to participate in conversation was important in a two-way interactive dialogue. In line with this, a study by Huang [14] found that asking follow-up questions are perceived as higher in responsiveness. Follow-up questions are those that followed up on the topic the interlocutor had mentioned earlier in the conversation (almost always in the previous turn) [14]. Follow-up questions are considered as a sign of giving attention, which include listening, understanding, validation, and care. They are often asked to show that we are interested or surprised and intended as a topic continuation on the objects, properties, and relations that are salient [16]. It is an easy and effective way to keep the conversation going and show that the asker has paid attention to what their partner has said. People like the ones who asked follow-up questions more than those who did not. As a result, an increase in likability is spread across all types of interactions, be it professional, personal, or romantic [14].

In this research, we aim to develop a dialogue agent that is able to generate follow-up questions in the context of social dialogue, such as small talk to get to know each other. Small talk is usually thought of as what strangers do when they meet, but it can generally be considered as any talk that does not emphasize task goals [3]. This type of dialogue can be categorized as non-task-oriented or open domain conversations.

Creating a non-task-oriented dialogue agent is a challenging problem. Unlike task-oriented or closed-domain dialogue agents - in which it is possible to prepare knowledge for a domain and generate modules for that domain - open-domain dialogue agents have a wide variety of topics and actions, such as greetings, questions, and self-disclosure. Since it is difficult to handle all aspects of the user's open-domain utterances, to create workable systems, we focus on the generation of one-turn response from a short-text conversation. The one-turn response only examines one round of conversation, in which each round is considered by two short texts. The first text is input from the user and the later being a response given by the computer. This method has demonstrated by [24] and [26] to suppress the complexity of information that agents are required to deal with.

1.2 Research Questions

The main research question in this theses is formulated as follow: **How to generate follow-up questions for an open-domain conversational system?** This question is detailed into three sub-questions:

1. What is the system architecture to generate follow-up question?
2. How to formulate the follow-up questions?
3. How to evaluate the follow-up questions performance?

1.3 Outline

The rest of the report is structured as follows. Chapter 2 discusses the concept of dialogue systems and automatic question generation. Chapter 3 describes the methodology we use to generate follow-up questions. The evaluation of the generated questions is explained in Chapter 4. After that, Chapter 5 discusses about user study. Concluding remarks and future works follow in Chapter 6.

Chapter 2

Related Works

This chapter will start with a description about dialogue systems in section 2.1 which include two approaches to build non-task-oriented dialogue systems, i.e. rule-based systems in subsection 2.1.1 and corpus-based systems in subsection 2.1.2. After that, we will consider the common methods in question generation from texts in section 2.2. This include the discussion about syntax-based method in subsection 2.2.1, semantic-based method in subsection 2.2.2, and template-based method in subsection 2.2.3.

2.1 Dialogue Systems

Dialogue systems or conversational agents are programs that can communicate with users in natural language (text, speech, or even both) [15]. They can assist in human-computer interaction and might influence the behavior of the users by asking questions and responding to user's questions [1]. For example, participants of the study by [23] indicated a significant increase in positive attitudes following the persuasive dialogue towards regular exercise.

Generally, dialogue systems are distinguished into two classes: task-oriented and non-task-oriented dialogue agents. Task-oriented dialogue systems are aimed for a specific task and arranged to have short conversations to get information from the user to help complete the task [15]. Examples in our daily life include digital assistants that can be easily found in a cell phone or home controllers (e.g., Siri, Cortana, Alexa, Google Now/Home). These dialogue systems can help to send texts, make ticket reservations, order pizza, control home appliances, or give travel directions. Another example is virtual assistants in an online shopping website, where a task-oriented dialogue system is deployed to help customers answer questions or address problems.

Non-task-oriented dialogue agents or chatbots are systems designed for extended conversation [15]. They are also known as open-domain dialogue agents since they focus on conversing with a human on open domains. Chatbots are built to impersonate the unstructured chats characteristic of human-human interaction instead of intended for a specific task such as reservation of the restaurant. Generally, chatbots carry an entertainment aspect. Examples include Simsimi[1], which has the capability to chat with people on text-based platforms and Microsoft XiaoIce, a Chinese socialbot capable of making real spoken word phone calls.

Chatbot architectures are generally distinguished into two classes: rule-based systems and corpus-based systems. In the rule-based system, rules are composed of many pattern-response pairs that are built by hand. These systems find patterns that match the phrase contained in the user utterances and generate response sentences associated with the patterns [26]. Corpus-based systems mine vast amount of human-human conversations' dataset [15]. According to [15], some of the techniques used in corpus-based systems are information retrieval (IR-based system) [13, 26], machine translation paradigms such as the encoder-decoder model which can automatically generate a sentence associated with input sentence [27], or neural network sequence-to-sequence systems which map a user utterance to a system response [25]. In the rest of this section, we will discuss the two approaches to build chatbot systems since our focus is to develop non-task-oriented dialogue systems.

2.1.1 Rule-based systems

ELIZA is the very first chatbot system. It was designed to simulate a Rogerian psychologist, based on a branch of clinical psychology involving responding with non-directional questions to user inputs, reflecting the patient's statement at them. The following example is the most famous ELIZA conversation, excerpted from [15]:

[1] www.simsimi.com

```
Men are all alike.
IN WHAT WAY
They're always bugging us about something or other.
CAN YOU THINK OF A SPECIFIC EXAMPLE
Well my boyfriend made me come here.
YOUR BOYFRIEND MADE YOU COME HERE
He says I'm depressed much of the time.
I AM SORRY TO HEAR YOU ARE DEPRESSED
```

ELIZA was a rule-based system. ELIZA worked by pattern/transform rules. For example:

```
(0 YOU 0 ME) [pattern]
->
(WHAT MAKES YOU THINK I 3 YOU)[transform]
```

In the above example, 0 means Kleene* (asterisks). The numbers in the transform rules indicate the index of the constituent in the pattern, which implies that the number 3 points to the second 0 in the first pattern. Therefore, this rule will transform

`You like me`

into:

`WHAT MAKES YOU THINK I LOVE YOU`

Algorithm 1 is the simplified ELIZA algorithm (Algorithm 1) as mentioned in [15]. Each ELIZA pattern/rule is linked to a keyword that might occur in a user sentence.

Another example of a chatbot system a using rule-based method is the Alice Question Generation (AQG). AQG is a rule-based dialogue agent which uses a pattern matching technique to handle the user's text-based conversation [8]. AQG generates question and answers pairs about Alice in Wonderland. The generated QA pairs are stored in the QA database, wherein the questions from the users can be matched to the stored questions in the QA database when they are talking with virtual human Alice.

In addition, Higashinaka et al. [13] developed an open-domain conversational agent using a rule-based method that is able to respond to 90% of the sentences. Higashinaka et al. created the rules by referring to AIML (Artificial Intelligence Modelling Language) rules of ALICE (Artificial Linguistic Internet Computer Entity). They involved replacing

Algorithm 1 Excerpt of ELIZA algorithm [15]

```
function ELIZA GENERATOR(user sentence) return response
                        ▷ Find the word w in sentence that has the highest keyword rank
    if w exists then
        Choose the highest ranked rule r for w that matches sentence
        response ← Apply the transform in r to sentence
        if w = 'my' then
            future ← Apply a transformation to sentence
            Push future onto memory stack
        end if
    else                                                          ▷ (no keyword applies)
        either
        response ← Apply the transform for the NONE keyword to sentence
        or
        response ← Pop the top response from the memory stack
    end if
    return response
end function
```

certain words with asterisks (wildcard) to widen the coverage of patterns and modifying templates if necessary.

2.1.2 Corpus-based systems

Corpus-based systems mine conversations of human-human conversations or sometimes human-machine conversations instead of using hand-crafted rules [15]. For example, Sugiyama et al. [26] proposed a dialogue agents architecture that has three main parts: dialogue control or agent actions using preference-learning based inverse reinforcement learning (PIRL), utterance-generation, and question-answering.

To generate appropriate response utterances that have non-trivial information, Sugiyama et al. [26] synthesized a new sentence consisting of both a primary topic from a user utterance and a new topic relevant to the user utterance topic from a large corpus (Twitter). This way, they generate agent utterances containing new information relevant to user utterances in the hope of reducing the generation of parrot utterances (the same as the user sentences in the corpus).

To automatically define the relevance between topics, they extract both of the semantic units (phrase pair with a dependency relation that represents the topic utterances) from user utterances (3680 one-to-one text chats among people who talked without topic limitation in Japanese) and a large-scale corpus (150 M tweets in Japanese). For example, consider the illustration described in Figure 2.1. Given the user utterance "I want to go to Tokyo," they extract the semantic units from this sentence "to Tokyo" → "I want to go." After that, they search semantic units from the large corpus that have a topic related to user utterance such as "If I go to Tokyo, I want to visit Tokyo Tower." They then combine the retrieved semantic units and the input into a sentence like "If you go to Tokyo, are you going to visit Tokyo Tower?"

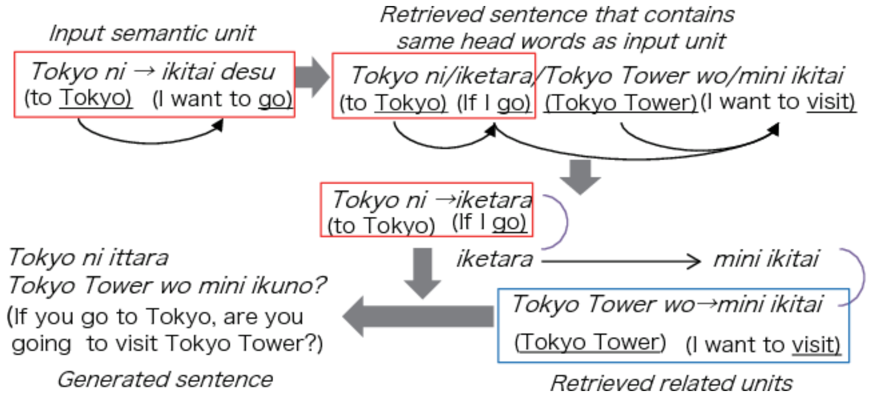
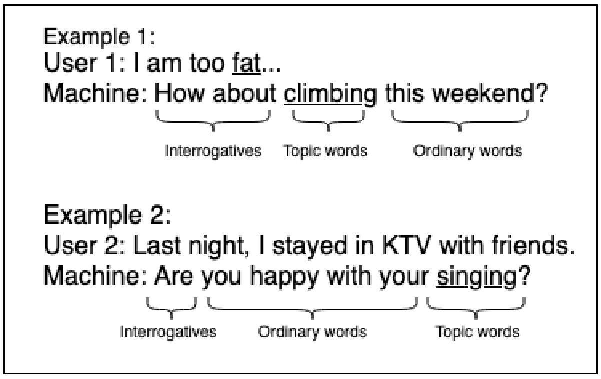

Input semantic unit
Tokyo ni → ikitai desu
(to Tokyo) (I want to go)
Retrieved sentence that contains same head words as input unit
Tokyo ni/iketara/Tokyo Tower wo/mini ikitai
(to Tokyo) (If I go) (Tokyo Tower) (I want to visit)
Tokyo ni →iketara
(to Tokyo) (If I go)
iketara → mini ikitai
Tokyo Tower wo→mini ikitai
(Tokyo Tower) (I want to visit)
Retrieved related units
Tokyo ni ittara
Tokyo Tower wo mini ikuno?
(If you go to Tokyo, are you going to visit Tokyo Tower?)
Generated sentence

Figure 2.1: The process of question creation in [26]

A study by Wang et al. [27] researched how to ask questions in an open domain conversational system with a Chinese text chat dataset. They collected about 9 million dialogue pairs from Weibo, a Chinese microblogging platform. They extracted the pairs whereby the response is in question form with the help of 20 hand-crafted templates. Wang et al. consider a task with one round conversation, in which each round is formed by a short text from a user and replied by a response from the computer. They suggested that a good question is composed of 3 elements: interrogatives, topic words, and ordinary words, as can be seen in Figure 2.2. Interrogatives lexicalize the pattern of questioning, topic words address the key information for topic transition in dialogue, and ordinary words play syntactical and grammatical roles in making sentences. Thus, they classify the words in a question into these three elements.

Work by [25] presents an approach to follow-up question generation for interview coaching in Chinese text using the integration of a CNTN (convolutional neural tensor network), seq2seq model, and an n-gram language model. Follow-up questions are divided into 16 types (verification, disjunctive, who, when, where, example, feature specification, quantification, comparison, interpretations, causal consequence, goal orientation, instrumental/procedural, enablement, expectation, judgmental). First, the authors adopt the



Example 1:
User 1: I am too fat...
Machine: How about climbing this weekend?

Example 2:
User 2: Last night, I stayed in KTV with friends.
Machine: Are you happy with your singing?

FIGURE 2.2: A good questions consists of interrogatives, topic words, and ordinary words [27].

word clustering method for automatic sentence pattern generation. Then the CNTN model is used to select a target sentence in an interviewee's answer turn. The selected target sentence pattern is fed to a seq2seq model to obtain the corresponding follow-up pattern. Then the generated follow-up question sentence pattern is filled with the words using a word-class table to obtain the candidate follow-up question. Finally, the n-gram language model is used to rank the candidate follow-up questions and choose the most suitable one as the response to the interviewee.

2.2 Question Generation

Question generation (QG) is the task to automatically generate questions given some input such as text, database, or semantic representation[2]. QG plays a significant role in both general-purpose chatbots (non-goal-oriented) systems and goal-oriented dialogue systems. QG has been utilized in many applications, such as generating questions for testing reading comprehension [12] and authentication question generation to verify user identity for online accounts [28]. In the context of dialogue, several studies have been conducted. For example, a question generation to ask reasonable questions for a variety of images [22], and a dialogue system to answer questions about Alice in Wonderland [8].

In order to generate questions, it is necessary to understand the input sentence or paragraph, even if that understanding is considerably shallow. QG utilizes both Natural

[2] http://www.questiongeneration.org/

Language Understanding (NLU) and Natural Language Generation (NLG). In conjunction with QG, there are three aspects to carry out in the task of QG, i.e., question transformation, sentence simplification, and question ranking as mentioned by Yao et al. in [29, 31]. Figure 2.3 illustrates these three challenges in an overview of a QG framework.

1. Sentence simplification. Sentence simplification is usually implemented in the preprocessing phase. It is necessary when long and complex sentences are transformed into short questions. It is better to keep the input sentences brief and concise to elude unnatural questions.
2. Question transformation. This task is to transform declarative sentences to interrogative sentences. There are generally three approaches to accomplish this task: syntax-based, semantics-based, and template-based, that will be the main discussion in this chapter.
3. Question ranking. Question ranking is needed in the case of over generation, that is, the system generates as many as questions as possible. A good ranking method is necessary to select relevant and appropriate questions.

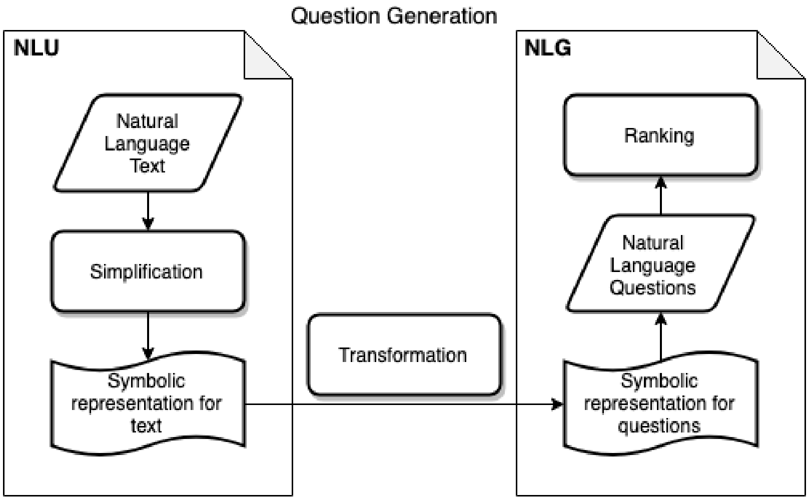

Question Generation
NLU
Natural Language Text
Simplification
Symbolic representation for text
Transformation
NLG
Ranking
Natural Language Questions
Symbolic representation for questions

FIGURE 2.3: Question generation framework and 3 major challenges in the process of question generation: sentence simplification, transformation, and question ranking, from [29].

There are generally three approaches to question transformation and generation: template-based, syntax-based, and semantics-based. We will discuss these approaches in the rest of this chapter.

2.2.1 Syntax-based Method

The word syntax derives from a Greek word *syntaxis*, which means *arrangement*. In linguistics, syntax refers to set of rules in which linguistic elements (such as words) are put together to form constituents (such as phrases or clauses). Greenbaum and Nelson in [11] refer to syntax as another term for grammar.

Work by Heilman and Smith in [12] exhibit question generation with a syntax-based approach. They follow the three-stage framework for factual QG question generation: (i) sentence simplification, (ii) question creation, and (iii) question ranking.

Sentence simplification. The aim of sentence simplification is to transform complex declarative input sentences into simpler factual statements that can be readily converted into questions. Sentence simplification involves two steps:

1. The extraction of simplified factual statements. This task aims to take complex sentences such as sentence 2.1 to become simpler statements such as sentence 2.2.

   `Prime Minister Vladimir V. Putin, the country's paramount leader, cut short a trip to Siberia.` (2.1)

   `Prime Minister Vladimir V. Putin cut short a trip to Siberia.` (2.2)

2. The replacement of pronouns with their antecedents (pronoun resolution). This task aims to eliminate vague questions. For example, consider the second sentence in example 2.3:

   `Abraham Lincoln was the 16th president. He was assassinated by John Wilkes Booth.` (2.3)

   From this input sentence, we would like to generate a proper question such as sentence 2.4. The other way around, with only basic syntactic transformation, the generated question is sentence 2.5:

   `Who was he assassinated by?` (2.4)

   `Who was Abraham Lincoln assassinated by?` (2.5)

Question creation. Stage 2 of the framework takes a declarative sentence as input and produces a set of possible questions as output. The process of transforming declarative sentences into questions is described in Figure 2.4.

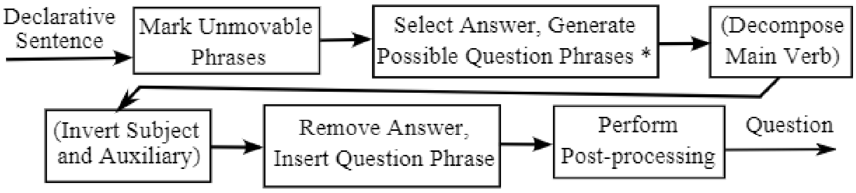

Declarative Sentence
Mark Unmovable Phrases
Select Answer, Generate Possible Question Phrases *
(Decompose Main Verb)
(Invert Subject and Auxiliary)
Remove Answer, Insert Question Phrase
Perform Post-processing
Question

FIGURE 2.4: The process of question creation of [12].

In *Mark Unmovable Pharses*, Heilman used a set of Tregex expressions. Tregex is a utility for identifying patterns in trees, like regular expressions for strings, based on tgrep syntax. For example, consider the expression 2.6.

$$\texttt{SBAR < /\^{}WH.*P\$/ << NP|ADJP|VP|ADVP|PP=unmv} \tag{2.6}$$

The expression 2.6 is used to mark phrases under a question phrase. From the sentence 'Darwin studied how species evolve,' the question 'What did Darwin study how evolve?' can be avoided because the system marks the noun phrase (NP) species as unmovable and avoids selecting it as an answer.

In the *Generate Possible Question Phrases*, the system iterates over the possible answer phrases. Answer phrases can be noun phrases (NP), prepositional phrases (PP), or subordinate clauses (SBAR). To decide the question type for NP and PP, the system uses the conditions listed in Table 2.1. For SBAR, the system only extracts the question phrase what.

In *Decomposition of Main Verb*, the purpose is to decompose the main verb into the appropriate form of *do* and the base form of the main verb. The system identifies main verbs which need to be decomposed using Tregex expressions.

The next step is called *Invert Subject and Auxiliary.* Consider the sentence 'Goku kicked Krillin'. This step is needed when the answer phrase is a non-subject noun phrase (example, 'Who kicked Krillin?') or when the generated question is a *yes-no* question (for example, 'Did Goku kick Krillin?'). But not when generating question 'Who did Goku kick?' After that, the system's task is to *remove the selected answer*

Table 2.1: Various WH questions from a given answer phrase in [12].

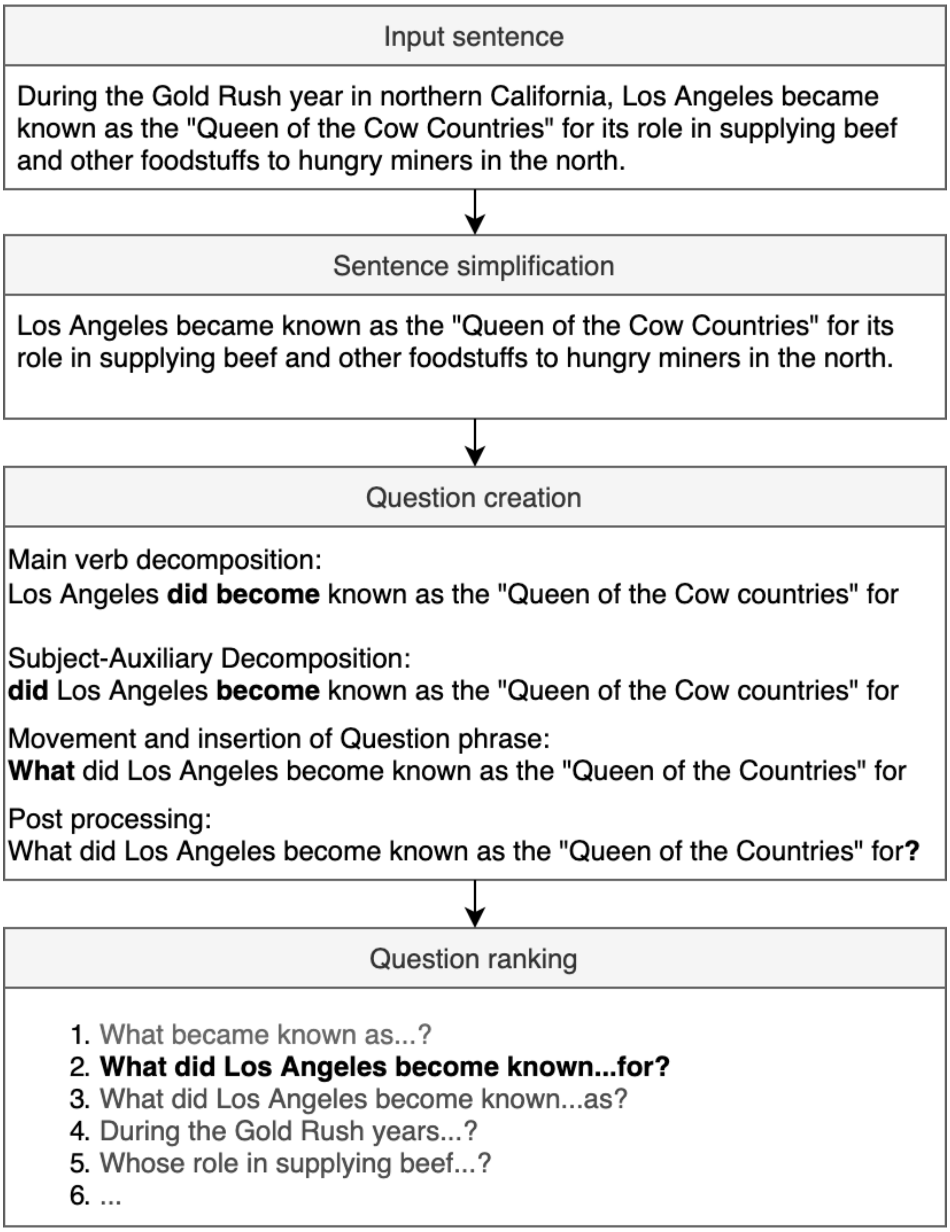
Wh word	**Condition**	**Examples**
who	The answer phrase's head word is tagged `noun.person` or is a personal noun (I, he, herself, them, etc)	Barack Obama, him, the 44th president
what	The answer phrase's head word is not tagged `noun.time` or `noun.person`	The pantheon, the building
where	The answer phrase is a prepositional phrase whose object is tagged `noun.location` and whose preposition is one of the following: *on, in, at, over, to*	in the Netherlands, to the city
when	The answer phrase's head word is tagged `noun.time` or matches the following regular expression (to identify years after 1900, which are common but not tagged as `noun.time`): `[1|2]\d\d\d`	Sunday, next week, 2019
whose *NP*	The answer phrase's head word is tagged `noun.person`, and the answer phrase is modified by a noun phrase with a possessive ('s or ')	Karen's book, the foundation's report
how many *NP*	The answer phrase is modified by a cardinal number or quantifier phrase (`CD` or `QP`, respectively)	eleven hundred kilometres, 9 decades

phrase and produce a new candidate question by inserting the question phrase into a separate tree.

Lastly, performing *post-processing* is necessary to put proper formatting and punctuation. For example, transforming sentence-final periods into question marks and removing spaces before punctuation symbols.

Question ranking. Stage 1 and two may generate many question candidates in which many of them are unlikely acceptable. Therefore, the stage 3 task is to rank these question candidates. Heilman uses a statistical model, i.e., least-square linear regression, to model the quality of questions. This method assigns acceptability scores to questions and then eliminates the unacceptable ones.

To illustrate how the system works, Figure 2.5 gives an example of the proposed approach by Heilman [12].



During the Gold Rush year in northern California, Los Angeles became known as the "Queen of the Cow Countries" for its role in supplying beef and other foodstuffs to hungry miners in the north.

Los Angeles became known as the "Queen of the Cow Countries" for its role in supplying beef and other foodstuffs to hungry miners in the north.

Main verb decomposition:
Los Angeles **did become** known as the "Queen of the Cow countries" for

Subject-Auxiliary Decomposition:
did Los Angeles **become** known as the "Queen of the Cow countries" for

Movement and insertion of Question phrase:
What did Los Angeles become known as the "Queen of the Countries" for

Post processing:
What did Los Angeles become known as the "Queen of the Countries" for**?**

1. What became known as...?
2. **What did Los Angeles become known...for?**
3. What did Los Angeles become known...as?
4. During the Gold Rush years...?
5. Whose role in supplying beef...?
6. ...

FIGURE 2.5: Example of question generation process from [12].

2.2.2 Semantics-based Method

Semantic analysis is the process to analyze the meaning contained within the text. It looks for relationships among the words, how they are combined, and how often certain words appear together. Usually the methods employed in semantic analysis include part of speech (POS) tagging, named entity recognition (NER) - finding parts of speech POS that refers to an entity and linking them to pronouns appearing later in the text (for example, distinguish between `Apple` the company and `apple` the fruit), or lemmatisation

- a method to reduce many forms of words to their base forms (for example, `tracking, tracked, tracks,` might all be reduced to the base form `track`).

The QG system developed at UPenn for QGSTEC, 2010, by Mannem et al. [18] represents the semantics approach. Their system combines semantic role labeling (SRL) with syntactic transformations. Similar to [12], they follow the three stages of QG systems: (i) content selection, (ii) question formation, and (iii) ranking.

Content selection. In this phase, ASSERT (Automatic Statistical SEmantic Role Tagger)[3] is employed to parse the SRL of the input sentences to obtain the predicates, semantic arguments, and semantic roles for the arguments. An example of an SRL parse resulting from ASSERT is given in sentence 2.7.

`[ She (ARG1)] [jumped (PRED)] [out (AM-DIR)] [to the pool (ARG4)] [with great confidence (ARGM-MNR)] [because she is a good swimmer (ARGM-CAU)]` (2.7)

This information is used to identify potential *target* content for a question. The criteria to select the targets are [18]:

1. Mandatory arguments. Any of the predicate-specific semantic arguments (*ARG0*. . . *ARG5*) are categorized as mandatory argument. From sentence 2.7, given *ARG1* of `jumped`, the question `'Who jumped to the pool with great confidence?'` (Ans: `She`) could be formed.

2. Optional arguments. Table 2.2 lists the optional arguments that are considered informative and good candidates for being a target. From sentence 2.7, the generated questions from *ARGM-CAU* would be `'Why did she jump out to the pool with great confidence?'`

3. Copular verbs. Copular verbs are special kind of verbs used to join an adjective or noun complement to a subject. Common examples are: be (is, am, are, was, were), appear, seem, look, sound, smell, taste, feel, become, and get. Mannem et al [18] limit their copular verbs to only the `be` verb and they use the dependency parse of the sentence to determine the arguments of this verb. They proposed

[3] http://cemantix.org/software/assert.html

TABLE 2.2: Roles and their associated question types

Semantic Role	**Question Type**
ArgM-MNR	How
ArgM-CAU	Why
ArgM-PNC	Why
ArgM-TMP	When
ArgM-LOC	Where
ArgM-DIS	How

to use the *right* argument of the verb as the *target* for a question unless the sentence is existential (e.g. `there is a...`). Consider the sentence `'Motion blur is a technique in photography.'` Using the *right* argument of the verb `'a technique in photography,'` we can create question `'What is motion blur?'` instead of using the *left* argument since it is too complex.

Question formation. In this phase, the first step is to identify the *verb complex* (main verb adjacent to auxiliaries or modals, for example, `may be achieved, is removed`) for each target in the first stage. The identification is using the dependency parse of the sentence. After that, transform the declarative sentence into an interrogative. The examples are shown in sentence 2.9 to 2.11, each generated from one of the target semantic roles from sentence 2.8.

`[Bob (ARG0)] [ate (PRED)] [a pizza (ARG1)] [on Sunday (ARGM-TMP)]` (2.8)

`Who ate a pizza on Sunday?` (2.9)

`What did Bob eat on Sunday?` (2.10)

`When did Bob eat a pizza?` (2.11)

Ranking. In this stage, generated questions from stage 2 are ranked to select the top 6 questions. There are two steps to rank to the questions:

1. The questions from main clauses are ranked higher than the questions from subordinate clauses.

2. The questions with the same rank are sorted by the number of pronouns occurring in the questions. A lower score is given to the questions that have pronouns.

2.2.3 Template-based Method

A question template is any predefined text with placeholder variables to be replaced with content from the source text. In order to consider a sentence pattern as a template, Mazidi and Tarau [20] specify 3 criteria: (i) the sentence pattern should be working on different domains, (ii) it should extract important points in the source sentence and create an unambiguous question, and (iii) semantic information that is transferred by the sentence pattern should be consistent across different instances.

Lindberg et al. [17] presented a template-based framework to generate questions that are not entirely syntactic transformations. They take advantage of the semantics-based approach by using SRL to identify patterns in the source text. The source text consists of 25 documents (565 sentences and approximately 9000 words) exhibiting a high-school science curriculum on climate change and global warming. Questions are then generated from the source text. The SRL parse gives an advantage for the sentences with the same semantic structure since they will map to the same SRL parse even though they have different syntactic structures. Figure 2.6 illustrates this condition.

Input 1: Because of automated robots (AM-CAU), the need for labor (A1) decreases (V).
Input 2: The need for labor (A1) decreases (V) due to automated robots (AM-CAU).
Generated question: Describe the factor(s) that affect the need for labor.

FIGURE 2.6: Example generated question from two sentences with the same semantic structure.

Lindberg et al. manually formulated the templates by observing patterns in the corpus. Their QG templates have three components: plain text, slots, and slot options. *Plain text* acts as the question frame in which semantically-meaningful words from a source sentence are inserted to create a question. *Slots* receive semantic arguments and can occur inside or outside the *plain text*. A slot inside the *plain text* acts as a variable to be replaced by the appropriate semantic role text, and a slot outside the *plain text* provides additional matching criteria. The *slot options* task is to modify the source sentence text. To illustrate this, expression 2.12 gives an example of a template. This template has A0 and A1 slots. A0 and A1 determine the template's semantic pattern, which will match any clause containing an A0 and an A1. The symbols ## express the end of the question string.

`What is one key purpose of [A0]?  ## [A1]` (2.12)

The template approach by Lindberg et al. enables the generation of questions that do not include any predicates from the source sentence. Therefore, it allows them to ask more general questions. For example, look at sentence 2.13, instead of generating questions such as sentence 2.14, we could expect a question that is not merely factoid (question which requires the reader to memorize facts clearly stated in the source text) such as sentence 2.15.

`Expanding urbanization is competing with farmland for growth and putting pressure on available water stores.` (2.13)

`What is expanding urbanization?` (2.14)

`What are some of the consequences of expanding urbanization?` (2.15)

Another representation of the template-based approach is QG from sentences by Mazidi and Tarau [20]. To generate questions from sentences, their work consists of 4 major steps:

1. Create the MAR (Meaning Analysis Representation) for each sentence
2. Match sentence patterns to templates
3. Generate questions
4. Evaluate questions

Creating MAR (Meaning Analysis Representation). Mazidi developed the DeconStructure algorithm to create MAR. This task involved two major phases: deconstruction and structure formation. In the deconstruction phase, the input sentence is parsed with both a dependency parse and an SRL parse using SPLAT[4] from Microsoft Research. In the structure formation phase, the input sentence is divided into one or more independent clauses, and then clause components are identified using information

[4] http://research.microsoft.com/en-us/projects/msrsplat/

from SPLAT. Given sentence 2.16, Table 2.3 illustrates the output of SRL and dependency parses. Table 2.4 gives an example of MAR.

```
The DeconStructure algorithm creates a functional-semantic
representation of a sentence by leveraging multiple parses.
```
(2.16)

Table 2.3: Output from SRL and dependency parser from [20]

	Token	**SRL**	**Dependency**
1	The	B-A0	det (algorithm-3,the-1)
2	DeconStructure	I-A0	compmod(algorithm-3,DeconStructure-2)
3	algorithm	E-A0	nsubj(creates-4,algorithm-3)
4	creates	S-V	ROOT(root-0,creates-4)
5	a	B-A1	det(representation-7,a-5)
6	functional-semantic	I-A1	amod(representation-7,functional-semantic-6)
7	representation	I-AI	dobj(creates-4,representation-7)
8	of	I-AI	adpmod(representation-7,of-8)
9	a	I-AI	det(sentence-10,a-9)
10	sentence	E-AI	adpobj(of-8,sentence-10)
11	by	B-AM-MNR	adpmod(creates-4,by-11)
12	leveraging	I-AM-MNR	adpcomp(by-11,leveraging-12)
13	multiple	I-AM-MNR	amod(parses-14,multiple-13)
14	parses	E-AM-MNR	dobj(leveraging-12,parses-14)

Table 2.4: MAR for sentence 2.16 from [20]

Constituent	**Text**
predicate	creates
subject	the DeconStructure algorithm
dobj	a functional-semantic representation of a sentence
MNR	by leveraging multiple parses

Matching sentence patterns to templates. A sentence pattern is a sequence that consists of the root predicate, its complement, and adjuncts. The sentence pattern is key to determine the type of questions. Table 2.5 gives examples of sentence patterns and their corresponding source sentences commonly found in the repository text from [20].

Generating questions. Before generating a question, each sentence is classified according to its sentence pattern. After that, the sentence pattern is compared against

Table 2.5: Example of sentence patterns from [20]

Sentence Pattern and Sample
Pattern: `S-V-acomp` Meaning: Adjectival complement that describes the subject. Sample: Brain waves during REM sleep appear similar to brain waves during wakefulness.
Pattern: `S-V-attr` Meaning: Nominal predicative defining the subject Sample: The entire eastern portion of the Aral sea has become a sand desert, complete with the deteriorating hulls of abandoned fishing vessels.
Pattern: `S-V-ccomp` Meaning: clausal complement indicating a proposition of subject Sample: Monetary policy should be countercyclical to counterbalance the business cycles of economic downturns and upswings.
Pattern: `S-V-dobj` Meaning: indicates the relation between two entities Sample: The early portion of stage 1 sleep produces alpha waves.
Pattern: `S-V-iobj-dobj` Meaning: indicates the relation between three entities Sample: The Bill of Rights gave the new federal government greater legitimacy.
Pattern: `S-V-parg` Meaning: phrase describing the how/what/where of the action Sample: REM sleep is characterized by darting movement of closed eyes.
Pattern: `S-V-xcomp` Meaning: non-finite clause-like complement Sample: Irrigation systems have been updated to reduce the loss of water.
Pattern: `S-V` Meaning: May contain phrases that are not considered arguments such as ArgMs. Sample: The 1828 campaign was unique because of the party organization that promoted Jackson.

roughly 70 templates. Each template contains filters to check the input sentence, for example, whether the sentence is in an active or passive voice. A question can be generated if a template matches a pattern. The templates used by [20] has six fields. Sentence 2.16 together with Table 2.6 give an example of a template and its description.

Evaluating questions. Instead of ranking the output questions to identify which questions are more likely to be acceptable, [20] opted to evaluate the question importance. They utilized the TextRank algorithm [21] to extract 25 nouns as keywords from the input passage. After that, they gave a score to each generated question based on the percentage of top TextRank words. Sentences with a very short question such as `‘What is a keyword?’` were excluded.

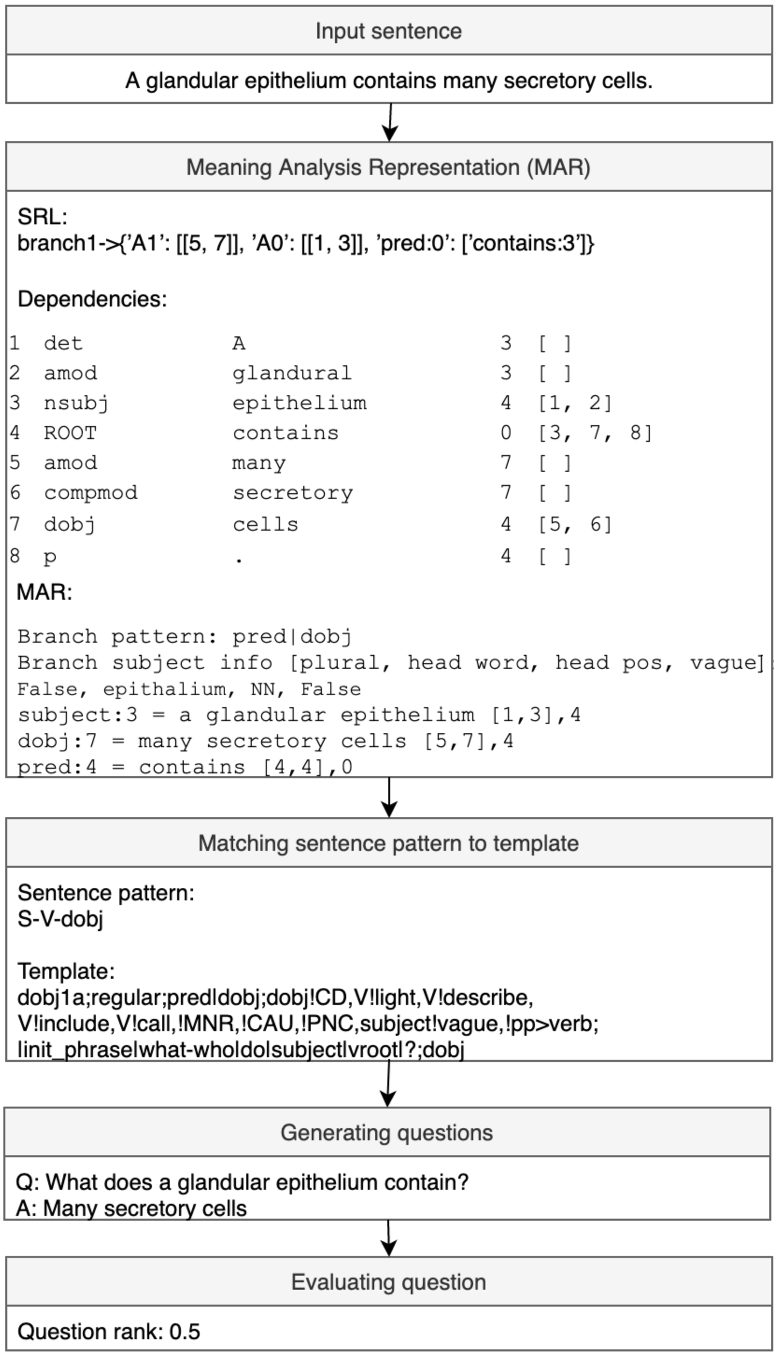
Table 2.6: Example of template from [20]

Field	**Content**
label	`dobj`
sentence type	`regular`
pattern	`pred|dobj`
requirements and filters	`dobj!CD, V!light, V!describe, V!include, V!call, !MNR, !CAU, !PNC, subject!vague, !pp>verb`
surface form	`|init_phrase|what-who|do|subject|vroot|`
answer	dobj

Figure 2.7 shows the overall process by [20] given the sentence `A glandural epithelium contains many secretory cells`.

2.3 Discussion

Creating rules is still the standard way of creating a conversational system. Even though it was argued that the rule-based method could not deal with a wide range of topics, Higashinaka et al. [13] overcame this drawback with many predicate-driven rules. This procedure involved substituting certain words with asterisks (wild card) to improve the coverage of the topic sentence and adjusting template if necessary. For example, *I like * → What do you like about it?*. Moreover, the winners in the Loebner Prize are still dominated by the rule-based system chatbots. The Loebner Prize is the oldest Turing Test contest, started in 1991 by Hugh Loebner and the Cambridge Center for Behavioral Studies. As of 2018, none of the chatbots competing in the finals managed to fool the judges believing it was human, but there is a winning bot every year. The judges ranked the chatbots according to how human-like they were. In 2018, Mitsuku, build based on rules written in AIML, developed by Steve Worswick, scores 33% out of 100%, the highest among all participants. Other than that, rule-based systems are easy to understand, to maintain, and to trace and fix the cause of errors [5]. Based on these considerations, the rule-based approach was finally chosen because the developing time was reasonable compared to the corpus-based approach.



Input sentence

A glandular epithelium contains many secretory cells.

Meaning Analysis Representation (MAR)

SRL:
branch1->{'A1': [[5, 7]], 'A0': [[1, 3]], 'pred:0': ['contains:3']}

Dependencies:

```
1  det           A                   3  [ ]
2  amod          glandural           3  [ ]
3  nsubj         epithelium          4  [1, 2]
4  ROOT          contains            0  [3, 7, 8]
5  amod          many                7  [ ]
6  compmod       secretory           7  [ ]
7  dobj          cells               4  [5, 6]
8  p             .                   4  [ ]
```

MAR:

```
Branch pattern: pred|dobj
Branch subject info [plural, head word, head pos, vague]
False, epithalium, NN, False
subject:3 = a glandular epithelium [1,3],4
dobj:7 = many secretory cells [5,7],4
pred:4 = contains [4,4],0
```

Matching sentence pattern to template

Sentence pattern:
S-V-dobj

Template:
dobj1a;regular;predldobj;dobj!CD,V!light,V!describe,
V!include,V!call,!MNR,!CAU,!PNC,subject!vague,!pp>verb;
linit_phraselwhat-wholdolsubjectlvrootl?;dobj

Generating questions

Q: What does a glandular epithelium contain?
A: Many secretory cells

Evaluating question

Question rank: 0.5

Figure 2.7: An example of a generated question and answer pair from the QG system of Mazidi and Tarau.

Furthermore, previous works indicate that automatically-generated questions are a dynamic, ongoing research area. The generated questions are generally the result of transformations from declarative into interrogative (question) sentences. This makes these approaches applicable across source text in different domains. Many approaches use the source text to provide answers to the generated questions. For example, given the sentence `‘Saskia went to Japan yesterday’`, the generated question might be `‘Where did Saskia go yesterday?’` but not `‘Why did Saskia go to Japan?’` This behavior of asking for stated information in the input sources makes question generation applicable in areas such as educational teaching, intelligent tutoring system to help learners check their understanding, and closed-domain question answering system to assemble question-answers pairs automatically.

We believe that there is still value in generating questions in an open-domain area. The novel idea we wish to explore is semantic-based templates that use SRL as well as POS and NER tags in conjunction with open-domain scope for a dialogue agent. Research by [30] and [8] shows that QG can be applied for a conversational character. In addition, Lindberg et al. [17] have demonstrated the question generation for both general and domain-specific questions. General questions were intended to generate questions that are not merely factoids (questions that have facts explicitly stated in the source text). General questions can benefit us to generate follow-up questions in which the answer is not mentioned in the source text.

Lindberg et al. [17] used SRL to identify patterns in the input text from which questions are generated. This work is most closely parallel with our work with some distinctions: our system only asks questions that do not have answers in the input text, our approach is domain-independent, and we observe not only the source sentence but also how to create the follow-up question in a conversation, and exploit the use of NER and POS tagging to create the question templates.

Chapter 3

Methodology

We propose a template-based framework to generate follow-up questions from input texts, which consists of 3 major parts: pre-processing, deconstruction, and construction, as shown in Figure 3.1. The system does not generate answers. A design decision was made only to generate follow-up questions in response to the input sentence. In this chapter, we will describe the component of the systems. Started with the description of the dataset in section 3.1, followed by the explanation of the pre-processing in section 3.2, the deconstruction in section 3.3, and finally the construction of the follow-up questions in section 3.4.

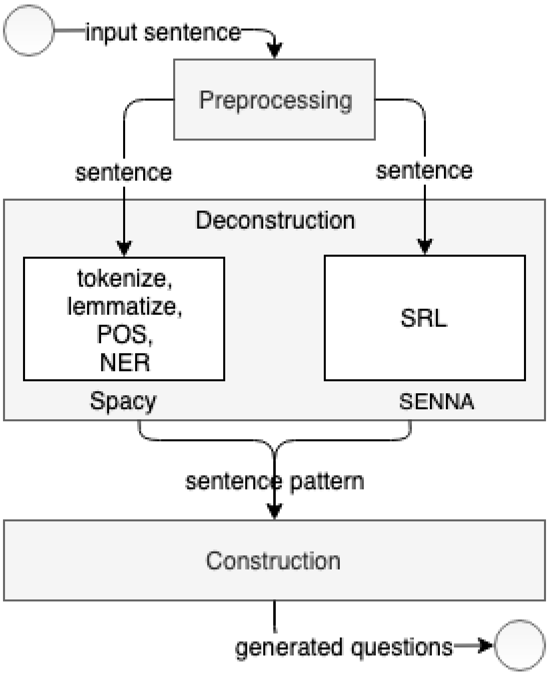


FIGURE 3.1: System architecture and data flow.

3.1 Dataset

We use a dataset[1] from research by Huang et al. [14] to analyze the sample of follow-up questions and their preceding statements. The dataset is about live online conversations with the topic 'getting to know you.' It contains 11867 lines of text, and 4545 of them are classified as questions. The questions are labeled with the following tags: followup, full (full switch), intro (introductory), mirror, partial (partial switch), or rhet (rhetorical). In this dataset, there are 1841 questions with label 'followup' and we focus our observation only on this label. The following example illustrates the kind of follow-up question that we found in [14].

```
User 1:  I enjoy listening to music, spending time with my children, and
vacationing
User 2:  Where do you like to go on vacation?
```

According to [14], follow-up questions comprise of appreciation to the previous statement ("nice," "cool," "wow"), or question phrases that stimulate elaborations ("which," "why. . . ," "what kind. . . ," "is it. . . ," "where do. . . , " "how do. . . "). These are the most prominent distinctive features of follow-up questions when [14] classified the question types. For practical reason, we analyze follow-up questions that start with the question words that encourage elaborations. With these criteria, there are 295 pairs of statement and follow-up questions used for observation. The distribution of the follow-up question types our dataset can be seen in Table 3.1.

TABLE 3.1: Distribution of the selected follow-up question types from the dataset

Follow-up Question Types	**Number**
Which	30
Why	22
What kind	59
Is it	50
Where do	70
How do	64

[1] train_chats.csv available online at https://osf.io/8k7rf/

3.2 Pre-processing

The system first pre-processed the input sentences. In the pre-processing stage, extra white spaces are removed, and contractions are expanded. Extra white spaces and contractions can be problematic for parsers, as they may parse sentences incorrectly and generate unexpected results. For example, sentence `I'm from France` is tagged differently from `I am from France` as illustrated in 3.1 and 3.2. This may affect the template matching process as we combine POS tagging, NER, and SRL to create a template.

```
I   '   m  from France
PRP VBZ NN IN   NNP[2]                                  (3.1)
```

```
I   am  from France
PRP VBP IN   NNP[3]                                     (3.2)
```

To handle the contractions, we use the Python Contractions library[4], which is able to perform contraction by simple replacement rules of the commonly used English contractions. For example, "don't" is expanded into "do not". It also handles some slang in contractions such as "ima" is expanded to "I am going to" and "gimme" is expanded to "give me".

Similar to [17], we do not perform sentence simplification since the common method of sentence simplification can discard useful semantic content. Discarding semantic content may cause to generate questions that have an answer in the input sentence, something that we want to prevent as we aim to generate follow-up questions. Sentence 3.3 and 3.4 show how a prepositional phrase can contain important semantic information. In this example, removing the propositional phrase in sentence 3.3 discards temporal information (AM-TMP modifier) as can be seen in sentence 3.4. Thus, the question `When do you run?` for example, is not fit to be a follow-up question for sentence 3.3,

[2] See Appendix A

[3] See Appendix A

[4] https://pypi.org/project/contractions/

because the answer `During the weekend` is already mentioned.

`During the weekend (AM-TMP), I (A0) ran (V).` (3.3)

`I (A0) ran (V).` (3.4)

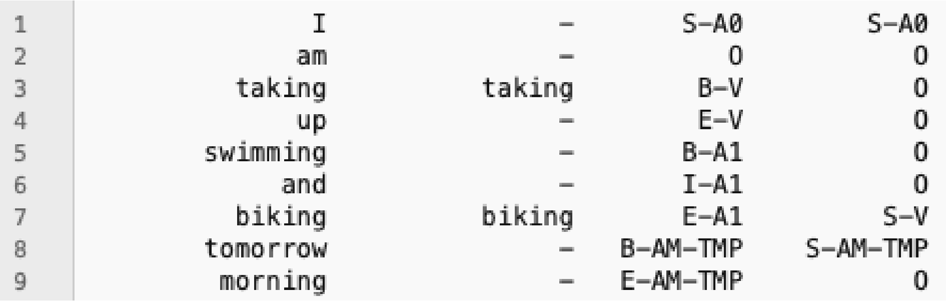
3.3 Deconstruction

After pre-processing, the next step is called deconstruction, which aims to determine the sentence pattern. Each input sentence is tokenized and annotated with POS, NER, and its SRL parse. By using SRL, the input sentence is deconstructed into its predicates and arguments. SENNA [7] is used to define the SRL of the text input. SENNA was selected since it is easy to use and able to assign labels to many sentences quickly. Semantic role labels in SENNA are based on the specification in Propbank 1.0. Verbs (V) in a sentence are recognized as predicates. Semantic roles include mandatory arguments (labeled A0, A1, etc.) and a set of optional arguments (adjunct modifiers, started with AM). Table 3.2 provides an overview.

TABLE 3.2: Semantic role label according to PropBank 1.0 specification from [9]

Label	**Role**
A0	proto-agent (often grammatical subject)
A1	proto-patient (often grammatical object)
A2	instrument, attribute, benefactive, amount, etc.
A3	start point or state
A4	end point or state
AM-LOC	location
AM-DIR	direction
AM-TMP	time
AM-CAU	cause
AM-PNC	purpose
AM-MNR	manner
AM-EXT	extent
AM-DIS	discourse markers
AM-ADV	adverbial
AM-MOD	modal verb
AM-NEG	negation

Given a sentence input, SENNA divides input sentence into one or more clauses. For instance, in Figure 3.2, we can see that SENNA divides the sentence '`I am taking up` `swimming and biking tomorrow morning'` into two clauses. The first clause is `'I'm (A0) taking up (V) swimming and biking (A1) tomorrow morning (AM-TMP)'` and the second clause is `'I'm (A0) biking (V) tomorrow (AM-TMP).'`



```
        I          -       S-A0       S-A0
       am          -          O          O
   taking     taking        B-V          O
       up          -        E-V          O
 swimming          -       B-A1          O
      and          -       I-A1          O
   biking     biking       E-A1        S-V
 tomorrow          -   B-AM-TMP   S-AM-TMP
  morning          -   E-AM-TMP          O
```

FIGURE 3.2: Sample of SRL representation produced by SENNA.

The Python library spaCy[5] was employed to tokenize and gather POS tagging and NER from the sentences. SpaCy was selected because, based on personal experience, it is easy to use, and according to the research by [2], spaCy provides the best overall performance compared to Stanford CoreNLP Suite, Google's SyntaxNet, and NLTK Python library.

Figure 3.3 illustrates a sentence and its corresponding pattern. Named entity and part of speech annotations are shown in the left-hand side of the figure, and one predicate and their semantic arguments are shown in the right-hand side of the figure. This sentence only has one clause, belonging to predicate *go*, and a semantic pattern described by an A1 and an AM-DIR containing two entity of type ORGANIZATION and LOCATION. The description of the POS tags is provided in Appendix A.

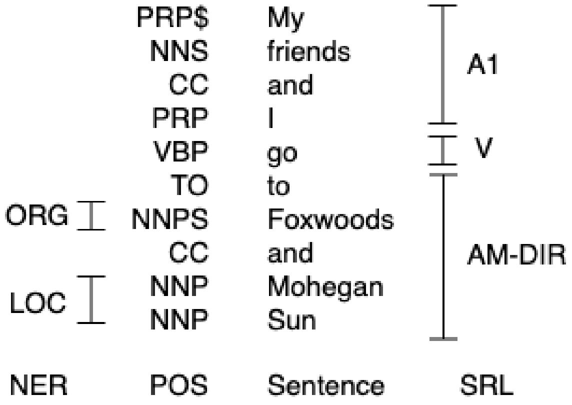


FIGURE 3.3: An example of a sentence and its corresponding pattern.

[5] https://spacy.io/

3.4 Construction

The purpose of the construction stage is to construct follow-up questions by matching the sentence pattern obtained in the Deconstruction phase to several rules. A follow-up question is generated according to the corresponding question's template every time there is a matching rule.

To develop the follow-up question's templates, first, we analyzed a set of follow-up questions from the dataset described in Chapter 3.1. We examine samples of the follow-up questions that contain the topics mentioned in the source sentence. A topic is a portion from input text containing useful information [4]. We do not handle follow-up questions which topics are not in the body text. For example, consider the source sentence 3.5 we take from the dataset. A possible follow-up question generated from the system is sentence 3.6 but not sentence 3.7. Because the word 'music' in sentence 3.7 is not a portion of the source sentence 3.5.

`My friend and I are actually in a band together on campus.` (3.5)

`What kind of band are you in?` (3.6)

`What kind of music does your band play?` (3.7)

In addition to follow-up questions that repeat parts of the input sentence, we also use general-purpose rules to create question's templates, enabling us to ask questions even though the answer is not present in the body of source sentences as demonstrated by [4]. Templates are defined mainly by examining the SRL parse, as well as NER and POS tagging. SENNA is run over all the sentences in the dataset to obtain the predicates, their semantic arguments, and the semantic roles for the arguments. Along with this, we use spaCy to tag plain text with NER and POS tagging. This information then used to identify the possible content for the question template. The selection of contents in question templates is grouped based on three major categories: (i) POS and NER tags, (ii) semantic arguments, and (iii) default questions. Initially, there are 48 rules to create question templates which consist of 26 rules in category POS and NER tags, 12 rules in category semantic arguments, and 10 rules for default questions as can be seen in Table B.2 Appendix B. We will describe the specification of each category in the rest of this section.

3.4.1 POS and NER tags

The choice of question words based on the Named Entity Relation (NER) and Part of Speech (POS) tags are applied to mandatory arguments (A0…A4), optional arguments (start with the prefix ArgM), and copula verbs (refer to section 2.2.2).

We follow the work of Chali et al. [4] that utilized NER tags (people, organizations, location, miscellaneous) to generate some basic questions in which the answers are not present in the input source. Questions 'Who', 'Where', 'Which', and 'What' are generated using NER tag. Table 3.3 shows how different NER tags are employed to generate different possible questions.

Table 3.3: Question templates that utilize NER tag

Tag	**Question templates**	**Example**
person	Who is *person*?	Who is *Alice*?
org	Where is *org* located?	Where is *Wheelock College* located?
	What is *org*?	What is *Wheelock College*?
location	Where in *location*?	Where in *India*?
	Which part of *location*?	Which part of *India*?
misc	What do you know about *misc*?	What do you know about *Atlantic Fish*?

Often one sentence has multiple items with the same NER tag. For example, consider the following:

```
We went everywhere!  Started in Barcelona, then Sevilla, Toledo, Madrid...
                                    LOC            LOC     LOC     LOC
```

In this sentence, there are four words with the named entity 'LOC'. In order to minimize the repeated question about 'LOC', we only select one item to be asked. For practical purposes, we select the first 'LOC' item mentioned in the sentence. Thus, one example follow-up question about location from this sentence is `Which part of Barcelona?` The other locations are ignored.

We employ POS tags to generate 'Which' and 'What kind' questions to ask for specific information. Based on our observation of the dataset, the required elements to create 'Which' and 'What kind' questions are noun plural (NNS) or noun singular (NN). We also notice that the Proper Noun (NNP or NNPS) can be used to explore the opinion of the interlocutors. Thus we formulate the general-purpose questions '`What do you think about...`' and '`What do you know about...`' to ask for further information. Table 3.4 shows how we use POS tag to generate question templates.

Table 3.4: Question templates that utilize POS tag

Tag	**Question templates**	**Example**
nns	Which *nns* are your favorite?	Which *museums* are your favorite?
	What kind of *nns*?	What kind of *museums*?
nn	What kind of *nn*?	What kind of *museum*?
nnp	What do you think about *nnp*?	What do you think about *HBS*?
nnps	What do you know about *nnps*?	What do you know about *Vikings*?

If there are more multiple nouns in one sentence, similar to what we did to NER tag results, for practical purpose, we only select one noun to be asked, i.e. the first noun. For example:

```
I (A0) play (V) disc golf, frisbees, and volleyball (A1)

PRP    VBP      NN   NN     NNS       CC  NN
```

There are three nouns in this sentence: disc golf, frisbees, and volleyball. Since disc golf is the first noun in the sentence, then the possible follow-up question is `What kind of disc golf?` The other nouns that are mentioned in this sentence are ignored.

3.4.2 Semantic arguments

Mannem et al. and Chali et al. mentioned in their work [4, 18] that optional arguments starting with prefix AM (AM-MNR, AM-PNC, AM-CAU, AM-TMP, AM-LOC, AM-DIS) are good candidates for being a target in question templates. These roles are used to create questions that cannot be generated using only mandatory arguments (A0...A4). For example, AM-CAU can be used to generate a *Why* question, and AM-LOC can be used to generate a *Where* question. See Table 2.2 for all possibilities of optional arguments and their associated question types. However, this method is intended to create questions that have answers in their source sentences. To generate questions that do not have the answer in the body text of the source sentence, we examine whether the sentence pattern does not comprise one of these arguments. We also consider that any predicates having semantic role A0 and A1 are viable to formulate questions. Table 3.5 provides examples of generated questions that utilize semantic role.

Table 3.5: Example of generated questions with source sentences

Question 1: Where do you like to ride your bike? **Source:** I (A0) like (V) to ride my bike (A1) **Condition:** Source sentence does not have AM-LOC
Question 2: Why do you walk the dogs? **Source:** I (A0) then (AM-TMP) probably (AM-ADV) walk (V) the dogs (A1) later this afternoon (AM-TMP) **Condition:** Source sentence does not have AM-CAU and AM-PNC
Question 3: How do you enjoy biking around the city? **Source:** I (A0) enjoy (V) biking around the city (A1) **Condition:** Source sentence does not have AM-DIS and AM-MNR
Question 4: When did you visit LA and Portland? **Source:** I (A0) have also (AM-DIS) visited (V) LA and Portland (A1). **Condition:** Source sentence does not have AM-TMP

The template that generated Question 1 asks information about a place. It requires a verb, argument A0, and A1 that is started with an infinitive 'to' (POS tag = 'TO'). The template filter outs sentences with AM-LOC in order to prevent questioning statements that already provide information about a place. The template also filters out argument A1 that do not begin with *TO* to minimize questions that are not suitable when we ask 'Where' questions. Examples are shown in 3.8 and 3.9.

```
Source:   I (A0) have (V) a few hobbies (A1)
                                                        (3.8)
Question:   Where do you have a few hobbies?
```

```
Source:   I (A0) also (AM-DIS) love (V) food (A1)
                                                        (3.9)
Question:   Where do you love food?
```

In any case, this comes at a cost, as we lost the opportunity to create appropriate *Where* questions from Argument A1 that do not precede with an infinitive to a verb. For example:

```
Source:   Last week (AM-TMP) I (A0) saw (V) this crazy guy
drink and bike (A1)
                                                        (3.10)
Question:   Where did you see this crazy guy drink and
bike?
```

However, this shortcoming can be covered by asking other questions such as `‘Why do you have a few hobbies?’` for source sentence 3.8.

The template that generates Question 2 asks about reasons and explanations. Hence, it does not require optional arguments AM-CAU and AM-PNC in its source sentence. It requires a verb, argument A0, and A1. We do not apply any filter in argument A1.

The template for Question 3 requires a verb, argument A0 and A1, but does not include AM-DIS and AM-MNR. This template asks about *How* questions. Similar to question *Why,* we do not apply any filter in Argument 1.

Question 4 asks for information about what time something happens. Although this type of question was not in the dataset observation, we consider creating *When* question templates when argument AM-TMP is not in the source sentence. Aside from the absence of AM-TMP, this template requires a verb, A0, and A1.

3.4.3 Default questions

We provide default responses in the event of the system cannot match the sentence patterns and the rules. Some of these default responses were inspired by the sample questions in the dataset, and some were our creation. Since it can get a little boring getting the same old questions over and over, we prepare seven default questions as listed below. The detail conditions (rules) of these default are explained in Appendix B.

1. What do you mean?
2. How do you like that so far?
3. How was it?
4. Is that a good or a bad thing?
5. How is that for you?
6. When was that?
7. Can you elaborate?

Chapter 4

Question Evaluation

This chapter describes evaluations performed on the generated questions. In the following sections, two evaluations of follow-up questions are presented. In Evaluation 1, an initial evaluation is conducted by the author to ensure the quality of the templates. In Evaluation 2, external annotators carried out the evaluation.

4.1 Evaluation 1 Setup

The first step to evaluate question templates was an assessment by the author. Using 48 different rules to create templates, 514 questions generated from 295 source sentences in the dataset. See Table B.2 in Appendix B for a complete listing of the templates used in Evaluation 1.

We used a methodology derived from [4] to evaluate the performance of our QG systems. Each follow-up question is rated using two criteria: grammatical correctness and topic relatedness. For grammatical correctness, the given score is an integer between 1 (very poor) and 5 (very good). These criteria are intended to provide us a way to measure whether a question is grammatically correct or not. For topic relatedness, the given score is also an integer between 1 (very poor) and 5 (very good). We looked at whether the follow-up question is meaningful and related to the source sentence. Both criteria are guided by the consideration of the following aspects (Table 4.1).

Table 4.1: 5-Scales rating score adapted from [8]

Score	**Explanation**
Very Good (5)	The question is as good as the one that you typically find in a conversation
Good (4)	The question does not have any problem
Borderline (3)	The question might have a problem, but I'm not sure
Poor (2)	The question has minor problems
Very poor (1)	The question has major problems

4.2 Evaluation 1 Results

The results of Evaluation 1 are presented in Table 4.2. The overall both grammatical score and relation between follow-up question and source sentences are above the borderline score (3). However, the average Grammar and Relation score for question type 'Who' and 'How do' are below borderline. We will discuss the error analysis and templates improvement in the following subsection.

Table 4.2: Evaluation 1 results

Category	**Type**	**# Rules**	**# Question**	**Grammar**	**Relation**
POS and NER	What	8	26	4.0	4.0
	What kind	6	53	3.9	3.9
	Where	6	42	4.2	3.8
	Which	4	29	4.0	4.0
	Who	2	2	4.0	2.0
Semantic Arg	How do	4	123	2.7	2.9
	When	3	8	4.2	3.6
	Where do	1	18	3.9	3.6
	Why	4	144	3.9	3.8
Default	Default	10	69	4.0	3.8
Total		48	514	3.9	3.5

4.3 Error Analysis and Template Improvement

Based on the results of the first evaluation, we investigate the errors from each category. After that, the rules and question templates are improved. We provide the improved question templates in Table B.3 Appendix B. The error analysis and the template improvements are described in the following subsections.

4.3.1 POS and NER tag

In this section, we provide the error analysis and the improvements on question templates which were created based on POS and NER tags.

What. The average scores of *What* question type are above the borderline. Since we did not find items scores lower than the borderline, we leave the templates as they were.

What kind. The average scores of *What kind* question type are above the borderline. However, after observing the lower scores, several improvements were applied to the templates. The examples of errors found in this question type are shown in Table 4.3.

Table 4.3: Examples of errors found on *What kind* question type

No	Template	Clause	FU Question
1	WKD1	I enjoy playing video games, fitness, and exploration	What kind of video?
2	WKD6	I go to festivals and such	What kind of festival?

The SRL parse and POS tagging for the first clause are shown in 4.1:

```
I (A0) enjoy playing (V) video games, fitness, and exploration (A1)
PRP    VBP   VBG          NN   NNS  ,  NN    ,  CC  NN
```
(4.1)

There are three nouns in the first clause: `video games`, `fitness`, and `exploration`. However, the system failed to recognize compound noun `video games` because it distinguish the POS tags NN and NNS. `Video` and `games` are recognized as two nouns, not as a compound noun `video games`.

Our solution is to utilize SpaCy universal POS tags instead of the original POS tags (Penn Treebank tagset) as listed in Table A.1 Appendix A. The SpaCy universal POS tag set consists of 16 universal part-of-speech categories: NOUN (nouns), VERB (verbs), ADJ (adjectives), ADV (adverbs), PRON (pronouns), DET (determiners and articles), ADP (prepositions and postpositions), NUM (numerals), CONJ (conjunctions), PART (particles), PUNCT (punctuation marks), SYM (symbols), SPACE (space), PROPN (noun, proper), INTJ (interjection), and X (a catch-all for other categories such as abbreviations or foreign words)[1]. This way, the system is able to recognize compound

[1] https://spacy.io/api/annotation#pos-tagging

nouns as a whole as shown in 4.2.

```
I (A0) enjoy playing (V) video games,    fitness,   and exploration (A1)
PRP    VBP   VBG         NN    NNS  ,     NN    ,    CC  NN    (Orig)
PRON   VERB  VERB        NOUN  NOUN PUNCT NOUN PUNCT CONJ NOUN  (Univ)
```

(4.2)

From 4.2, we can see that using the universal POS tagging, the POS tag for the `video` and `games` are now a `NOUN`. The other nouns, `fitness` and `exploration`, are also labeled as a `NOUN`. To recognize `video games` as the first noun found in this sentence, we check whether POS tagging in the left and the right side of the first `NOUN` found is also a `NOUN` or the other tag. If it is also a `NOUN` then we consider it as a compound noun, otherwise it is a different entity. In this case, we can see that the word `video (NOUN)` is followed by `games (NOUN)`, therefore they are a compound noun. `Video games` are positioned between the word `playing (VERB)` and a comma `(PUNCT)` which are acknowledged as a different entity from `video games` as they have different POS tags.

Another improvement is conducted for the problem in the second clause of Table 4.3. In second clause, we transform the word `festivals` (plural, NNS) to `festival` (singular, NN). Thus, the generated question from the second clause (example 4.3) is `‘What kind of festival?’`. However, it is more natural if the system asks about various kinds of festivals. The improvement for this question template is to let the plural noun (NNS) stays as plural (NNS).

```
I (A0) go (V) to festivals and such (A1)
PRP    VBP    IN NNS       CC  JJ
```

(4.3)

Where. The average scores of *Where* questions are also above the borderline. However, we noticed some errors caused by errors in the NER tagging by SpaCy. `‘Nova’` mentioned in the first sentence in Table 4.4 is a name of a dog, yet its NER tag is ORG (organization). The word `‘Marvel’` in the second sentence refers to a movie franchise was tagged as LOC (location). Possible improvements are using another parser besides SpaCy or re-training the model from SpaCy's side which is out of our scope.

Which. The average scores of *Which* questions are also above the borderline. We also leave the templates as they were since we did not find items scores lower than the borderline.

TABLE 4.4: Examples of errors found on *Where* question type

No	Template	Sentence	FU Question
1	WHR2	Nova is a Shiba Inu	Where is *Nova* located?
2	WHR3	Hopefully Avengers live up to the hype, saw Marvel movies kinda fall flat for me	Where in *Marvel*?

Who. There are only two questions that resulted from this category as can be seen in Table 4.5, but both are incorrectly tagged by SpaCy. In the first sentence, `‘Sunset Cantina’` refers to the name of a Mexican Restaurant, but the NER tag is PERSON. In the second sentence, `‘Herbed Chicken’` is recognized as a PERSON. However, we notice that the first letter of `‘Herbed’` and `‘Chicken’` was written in capital, that is why this is tagged as PERSON. When we corrected the writing into `‘Herbed chicken’` (with *c* in the lowercase) then it is no longer labeled as PERSON. We consider that both of these errors are error parsings. In spite of this, we have to pay attention to the writing of the input sentence.

TABLE 4.5: Examples of errors found on *Who* question type

No	Template	Sentence	FU Question
1	WHO1	I like Machine, and Sunset Cantina.	Who is *Sunset Cantina*?
2	WHO1	Just a TV dinner. Herbed Chicken from Lean Cuisine.	Who is *Herbed Chicken*?

4.3.2 Semantic arguments

In this section, we provide the error analysis and the improvements on question templates which were created based on the semantic arguments.

How do. The average scores for question type *How do* are above the borderline. But, question templates HOW1 and HOW2 give a very low score. As we can see from the first and second sentence of Table 4.6, both templates do not manage the possessive pronoun, and more importantly they only have one mandatory argument. Since at least two mandatory arguments are needed to formulate questions [17], we exclude HOW1 and HOW2 in the improved version of the templates.

Table 4.6: Examples of errors found on *How do* question type

No	Template	Clause	FU Question
1	HOW1	I (A0) started (V) several months ago myself (AM-TMP).	How did you start several months ago myself?
2	HOW2	I'm (A0) studying (V) part time and in the process of starting my business (AM-TMP).	How do you study part time and in the process of starting my business?
3	HOW3	I (A0) enjoy (V) fitness like activities, professional sports, and photography (A1).	How do you enjoy fitness like activities, professional sports, and photography?

The follow-up question in third sentence `'How do you enjoy fitness like activities, professional sports, and photography?'` does not feel natural. From our observation of the dataset, people tend to ask a short and straightforward question rather than one long question at a time. To improve the template, we only include the first element before the comma in Arg1. Therefore, the new follow-up question is `'How do you enjoy fitness like activities?'`

When. Some problems that are found in this category can be explained using the clauses that are displayed in Table 4.7.

Table 4.7: Examples of errors found on *When* question type

No	Template	Clause	FU Question
1	WHN3	I really love to run, play soccer, hike, e-outdoors whenever possible, explore new places and go on adventures.	When do you love to run, play soccer, hike, e-outdoors whenever possible, explore new places and go on adventures?
2	WHN2	I (A0) actually (AM-ADV) took (V) a month (A1) off from life (AM-DIR) to travel after spring semester (AM-PNC)	When do you take a month?

The case of the first clause from Table 4.7 is similar to question template HOW3. Therefore, we simplify the question by selecting the first element before the comma in Arg1. The improved question is `'When do you love to run?'`

In the second clause, we noticed that the predicate is in past form, but the follow-up question is in the present form. Differentiating the type of verb helps to address this issue. We also found that the phrase `'month off'` was not labeled as one entity by SENNA. We consider this as error parsing.

Where do. The generated questions that are displayed in Table 4.8 show incorrect questions according to the given clause. The sentence for the first example is `‘I would love to work here for sometime’`, and SENNA parses this sentence into two clauses as can be seen in the Table 4.8. The generated question is based on the first clause (I (A0) would (AM-MOD) love (V) to work here (A1)) that does not contain AM-LOC. This has implications for the generated question `‘Where do you love to work here?’` This question is somewhat not suitable for a follow-up question since the source sentence already mentioned `‘here’` as the answer. However, the follow-up question from the dataset is kind of similar to the question generated by the system: `‘Where do you want to work?’`. Hence, we consider that this question might be asked in real life. However, it is interesting to analyze the SRL not only based on parsing results in one clause but also the whole sentence for future work.

Table 4.8: Examples of errors found on *Where do* question type

No	Template	Clause	FU Question
1	WHR4	I (A0) would (AM-MOD) love (V) to work here (A1) I (A0) work (V) here (LOC) for sometime (TMP)	Where do you love to work here?
2	WHR4	I like to cook, if you can call that a hobby and I like all types of craftwork.	Where do you like to cook, if you can call that a hobby and you like all types of craftwork?

The second example from Table 4.8 is similar to question template HOW3 and WHN3. We apply the same solution to the improved template by selecting the first element before the comma in Arg1.

Why. Some problems that are found in this category are explained using the clauses that are displayed in Table 4.9. For example, the first clause is a negation, but the follow-up question is asking the opposite. To overcome this situation, we are adding Negation question templates as described in section 4.3.3.

Another example is the follow-up question for the second clause: `‘Why do you do some work?’`. This question is not suitable for a follow-up question because the reason `‘so I can go have fun this weekend’` is already given in the input sentence. The source sentence for this question is `‘Doing some work today so I can go have fun this weekend’`, and SENNA parses this sentence into three clauses as can be seen in Table

4.9. However, none of these three clauses has AM-CAU to indicate the reason for an action. One improvement that can be done is to recognize the word `'so'` as a cause clause. Nonetheless, this template is still kept as it is. Because in this situation the word `'so'` does not belong to any argument so we can't generalize it to another sentence patterns.

Table 4.9: Examples of errors found on *Why* question type

No	Template	Clause	FU Question
1	WHY4	I (A0) haven't (NEG) done (V) it (A1) myself (A2) in a while (AM-TMP).	Why did you do it?
2	WHY4	Doing (V) some work (A1) today so I (A0) can go have fun this weekend. I (A0) can (AM-MOD) go (V). I (A0) can (AM-MOD) go have (V) fun (A1) this weekend (AM-TMP).	Why do you do some work?

4.3.3 Negation

We add some question templates to manage negative sentences. The Cambridge dictionary[2] mentions that one way to form a follow-up question is to use the auxiliary verb or modal verb contained in the statement that the question is responding to (see sentence 4.4 for example). Table 4.10 lists the questions templates that have been created for Negation category.

```
S: I can't swim.

Q: Can't you?
```

(4.4)

4.4 Evaluation 2 Setup

After error analysis and template improvement, another evaluation was conducted. An evaluation with external annotators was held to rate the generated follow-up questions from the improved templates. The 5-scales rating system displayed in Table 4.1 was again used for the evaluation. There were 418 questions and from 60 templates in this evaluation.

[2] https://dictionary.cambridge.org/grammar/british-grammar/speaking/question-follow-up-questions

Table 4.10: Templates for the negation

No	Template structure	Example
1	Why + aux + n't + A0 + V + A1?	S: I (A0) do not (NEG) have (V) a car (A1) Q: Why don't you have a car?
2	Why + aux + n't + A1 + V + A2?	S: I (A1) don't (NEG) get (V) into PRISE (A2) Q: Why don't you get into PRISE?
3	Why + aux + n't + A0?	S: I (A0) do not (NEG) know (V) that Q: Why don't you?
4	Why + AM-MOD + n't + A0 + + V + A1?	S: I (A0) could (MOD) not (NEG) get (V) the visa (A1) Q: Why couldn't you get the visa?
5	Why + AM-MOD + n't + A1 + V + A2?	S: I (A1) could (MOD) not (NEG) get (V) into PRISE(A2) Q: Why couldn't you get into PRISE?
6	Why + AM-MOD + n't + A0?	S: I (A0) will (MOD) not (NEG) know (V) that Q: Why won't you?
7	Why + AM-MOD + n't + A1 + V?	S: I (A1) could (MOD) not (NEG) get up (V) Q: Why couldn't you get up?

Two external annotators were assigned to judge the quality of the generated questions. One annotator is a master student of English taught program at the University of Twente, and another one is a university graduate who works as an editor in a publishing company. Both annotators are non-native English speakers, but they understand and speak English in daily life. Both annotators were briefed and explained about the 5-scale scoring system. The annotators were told to focus on the grammar of the follow-up questions as well as its relation to the source sentence. We used the same dataset as Evaluation 1. There is no dataset division, and each annotator judged the whole dataset.

4.5 Evaluation 2 Results

Table 4.11 presents the evaluation results after improvement. The first thing that we noticed in these results was the average scores for question type *How do* are improved above the borderline after we remove the templates that use only one semantic argument. Another thing is that almost of the errors found in Evaluation 1 are improved, except for the ones that cannot be fixed because of parser errors, such as NER tag errors in *Who* question type. Overall, total average scores are above the borderline and not so different compared to the results in Evaluation 1. The use of the same dataset as Evaluation 1 could result in this. Therefore we conduct a user study as an extension of the evaluation of generated questions.

Table 4.11: Evaluation 2 results

Category	**Question Type**	**Quantity**	**Grammatical**	**Relation**
POS and NER	How is	17	4.4	4.0
	What	29	4.2	3.8
	What kind	28	4.3	3.9
	Where	42	4.4	3.8
	Which	26	4.5	4.3
	Who	2	3.0	2.0
Optional Arg	How do	74	3.8	3.3
	When	7	3.9	3.4
	Where do	15	4.1	3.9
	Why	96	3.9	3.6
Negation	Negation	9	3.7	3.3
Default	Default	73	4.1	3.6
Total		418	4.0	3.6

Chapter 5

User Study

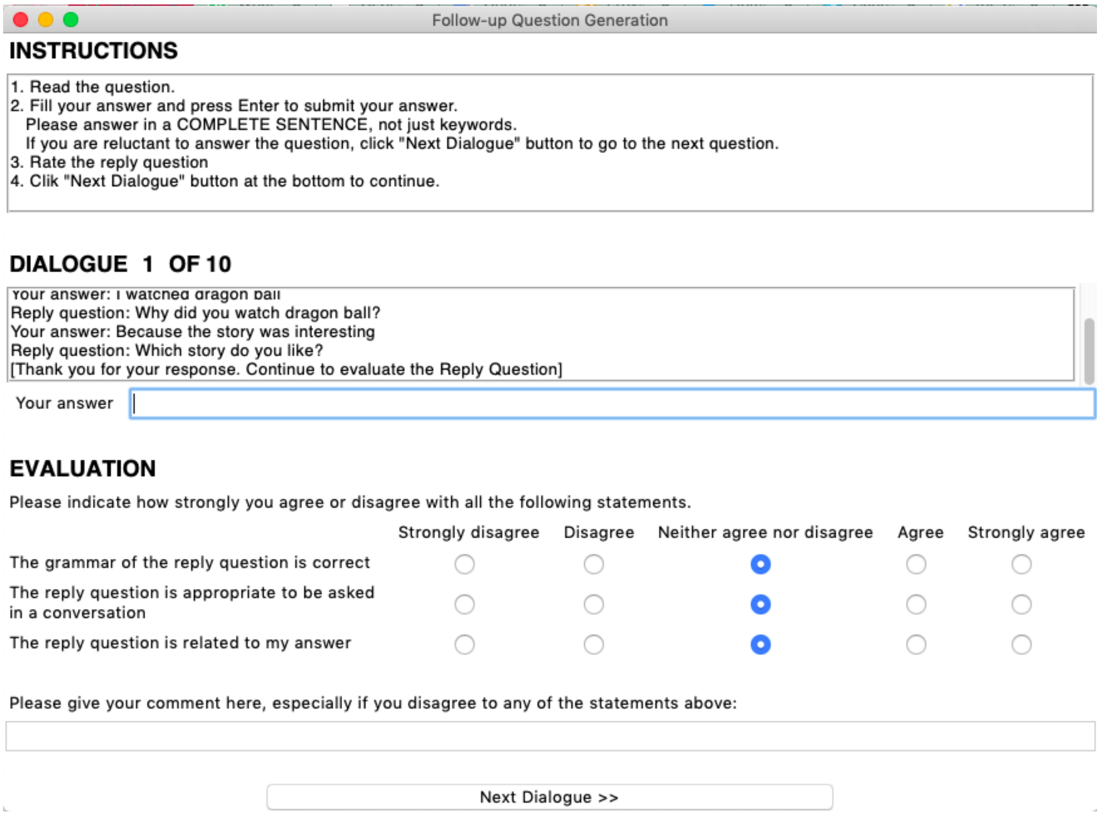
An evaluation with an interaction between the system and a user was conducted after evaluating question templates. Before executing the interactive user study, a pilot evaluation and improvements were implemented. Section 5.1 provides an explanation about pilot evaluation. Followed by the description about user study setup in section 5.2. After that, the results and discussion of the user study are explained in section 5.3.

5.1 Pilot Evaluation

The pilot evaluation was conducted to identify design issues before the main research is done. The procedure of the pilot study is described in subsection 5.1.1, results of the pilot study in subsection 5.1.2, and improvement during the pilot study in subsection 5.1.3.

5.1.1 Procedures

The two evaluators from the question evaluation phase (see section 4.4) were involved again in the pilot evaluation. The evaluators were first asked for their informed consent, that they were 18 years of age or older, fluent in English, their participation in the study was voluntary, and they were told that they might choose to terminate their participation in the study at any time for any reason. After that, the evaluators were asked to answer a series of casual conversation questions. They could choose to skip a question, but in the end, they needed to answer ten questions in total. The questions were randomly selected from 80 questions about ‘getting to know each other’, gathered from an English learning website[1]. After the evaluators answered each question, a follow-up question was displayed. In the system, the follow-up questions are referred to as the reply questions. Evaluators were informed that they may or may not answer the reply question, in the hope that we can make this a measure to find out if the reply question is interesting to answer. Following this, evaluators were obliged to rate the reply question, which consists of 3 statements on a 1 to 5 scale. Score 1 is “strongly disagree”, score 2 is “disagree”, score 3 is “neither agree nor disagree”, score 4 is “agree”, and score 5 is “strongly agree”. Next, they could give their opinion, especially when low scores are given. Finally, evaluators were asked to give their overall comments about the reply questions. Figure 5.1 shows the interaction page between the system and evaluators.



Follow-up Question Generation

INSTRUCTIONS

1. Read the question.
2. Fill your answer and press Enter to submit your answer.
   Please answer in a COMPLETE SENTENCE, not just keywords.
   If you are reluctant to answer the question, click "Next Dialogue" button to go to the next question.
3. Rate the reply question
4. Clik "Next Dialogue" button at the bottom to continue.

DIALOGUE 1 OF 10

Your answer: I watched dragon ball
Reply question: Why did you watch dragon ball?
Your answer: Because the story was interesting
Reply question: Which story do you like?
[Thank you for your response. Continue to evaluate the Reply Question]

Your answer

EVALUATION

Please indicate how strongly you agree or disagree with all the following statements.

	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
The grammar of the reply question is correct	○	○	●	○	○
The reply question is appropriate to be asked in a conversation	○	○	●	○	○
The reply question is related to my answer	○	○	●	○	○

Please give your comment here, especially if you disagree to any of the statements above:

FIGURE 5.1: User interface for pilot evaluation.

[1] https://www.eslconversationquestions.com/icebreakers-speaking-activities/

5.1.2 Results

The average rating results that the pilot evaluators gave are shown in Table 5.1. Overall, the interaction between the system and the evaluators was satisfactory. Both evaluators always respond back to the reply questions, although they were told that it was not mandatory. One evaluator gave a general impression that it was fun to answer the questions related to his interest. In spite of this, another evaluator felt that the follow-up questions were stiff. She expected that the system asked as a friend did.

Table 5.1: The rating results in pilot evaluation

No	**Statements**	**Rating**
1	The grammar of the reply question is correct	3.7
2	The reply question is appropriate to be asked in a conversation	3.8
3	The reply question is related to my answer	3.25

From our observation on the results of the pilot study, we found out that sometimes evaluators wrote in a complete sentence and sometimes just keywords, although we instructed them to write a complete sentence. We also learned that the evaluators tended to write the sentences in lower case. This had implications for the generated questions. For example, consider the following dialogue:

```
Q: Where are you from?
A: im from indonesia
Q: What do you mean?
```

We expected that the system would generate a follow-up question about the location mentioned in the answer, such as `‘Which part of Indonesia?’`. However, the answer `‘im from indonesia’` was all written in lower case, and the system did not recognize `‘indonesia’` as a location (LOC) since the first letter of `‘indonesia’` was not in the capital. The system also did not generate SRL pattern for this sentence because SENNA does not give predicate-argument structure for copular verbs. Hence, the system generated a default question `‘What do you mean?’`.

Another thing we observe is that there is a possibility that a default question is selected since we randomly select a follow-up question from a set of generated questions in the construction stage. This may happen when the evaluator inputs a long answer and

SENNA parses it into some clauses. If too many default questions arise, evaluators consider that the system does not understand them as expressed in a comment form by an evaluator, `‘The machine does not understand me.’`

5.1.3 Improvement

To overcome the issue that a user types in all lower case letters, we implemented auto-capitalization with a Python library named TrueCase[2]. TrueCase is a language modeling-based tool that restores case information for text. For example:

```
Original:  hey, what is the weather in los angeles?
TrueCase:  Hey, what is the weather in Los Angeles?
```

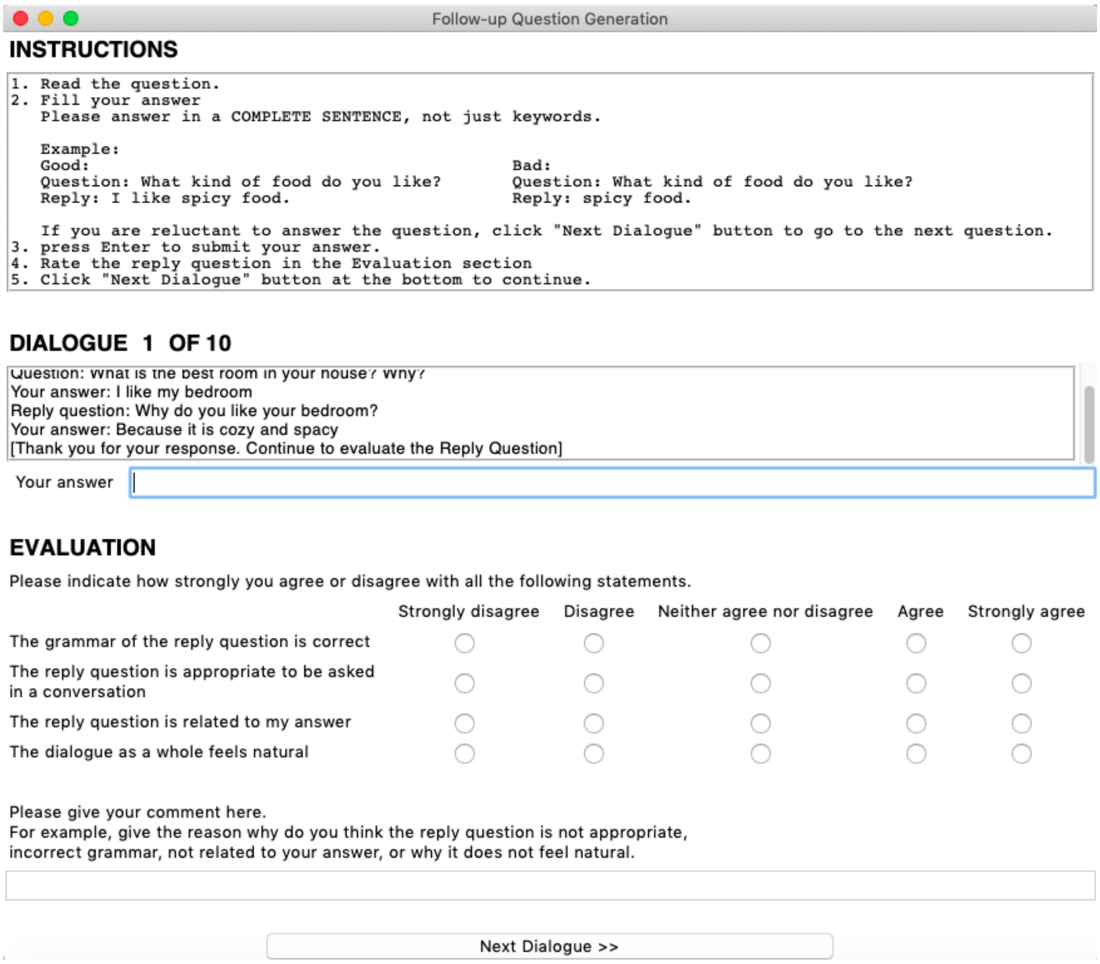
We also implemented a simple ranking mechanism as explained in section 5.1.4, in the hope that the selected question is a question that has correct grammar, related to the user’s input, and fewer default questions are displayed. Additionally, the default question `‘What do you mean?’` was changed to `‘Could you tell me more?’` to reduce the impression that the system is unintelligent.

Along with that, we improved the instruction displayed in the user interface by adding examples and more explanations. Moreover, we removed the second reply question from the display as this may confuse the evaluator when giving a score to the reply question. Furthermore, we added one more evaluation criterion: “The dialogue as a whole feels natural.” Figure 5.2 presents the user interface after improvement.

5.1.4 Ranking

The system accumulates a list of questions after the construction stage. This set of questions is ranked to select the best questions unless there’s only one question generated. We implemented a simple ranking mechanism using Python pandas library[3]. All question templates except default questions are ranked based on the average score from the evaluation results in section 4.5. The question template with the highest average score is selected as the follow-up question. If there are more than one question templates

[2] https://pypi.org/project/truecase/
[3] https://pandas.pydata.org/



INSTRUCTIONS

```
1. Read the question.
2. Fill your answer
   Please answer in a COMPLETE SENTENCE, not just keywords.

   Example:
   Good:                                     Bad:
   Question: What kind of food do you like?  Question: What kind of food do you like?
   Reply: I like spicy food.                 Reply: spicy food.

   If you are reluctant to answer the question, click "Next Dialogue" button to go to the next question.
3. press Enter to submit your answer.
4. Rate the reply question in the Evaluation section
5. Click "Next Dialogue" button at the bottom to continue.
```

DIALOGUE 1 OF 10

Question: What is the best room in your house? Why?
Your answer: I like my bedroom
Reply question: Why do you like your bedroom?
Your answer: Because it is cozy and spacy
[Thank you for your response. Continue to evaluate the Reply Question]

Your answer

EVALUATION

Please indicate how strongly you agree or disagree with all the following statements.

	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
The grammar of the reply question is correct	○	○	○	○	○
The reply question is appropriate to be asked in a conversation	○	○	○	○	○
The reply question is related to my answer	○	○	○	○	○
The dialogue as a whole feels natural	○	○	○	○	○

Please give your comment here.
For example, give the reason why do you think the reply question is not appropriate, incorrect grammar, not related to your answer, or why it does not feel natural.

FIGURE 5.2: User interface after improvement.

that have the highest score, one question is chosen randomly. The system generates a default question when there's only default question in the set of questions.

5.2 User Study Setup

We conducted the user evaluation with eight evaluators. None of them had been involved in in the question evaluation and pilot stage. Seven of them were students of the University of Twente: 5 masters and 2 PhD students. One of the evaluators was a university graduate. None of the evaluators were native English speakers, but they were able to communicate in English really well.

The same procedures in the pilot study (see subsection 5.1.1) were applied to all evaluators with some additions. All of the evaluators were explained what a complete sentence is along with an example (see Figure 5.2). But, we did not tell them the aim of the study. We emphasized that they were free not to fill the question in the dialogue to see whether there was less interest in answering some of the questions.

5.3 User Study Results and Discussion

There were supposed to be 80 follow-up questions (reply questions) in total, but there were only 79 questions evaluated by eight evaluators during the study. One question was not generated because the system was not responding to the last question. Table 5.3 presents the average rating result of the user study by each category. The total number of questions and their templates are given in Table 5.2.

Table 5.2: The total number of questions and their templates

Category	Type	Number of templates	Number of questions
POS and NER	What	4	4
	What kind	2	5
	Where	5	6
	Which	3	11
	Who	1	2
Semantic Arg	Where do	1	2
	Why	1	17
Default	Default	5	27
Negation	Negation	2	5
Total		24	79

Table 5.3: The average rating result of the user study

Category	Type	Grammar	Appropriateness	Relation	Naturalness
POS and NER	What	4.0	4.0	4.0	4.0
	What kind	4.0	2.5	2.8	2.9
	Where	3.8	3.3	3.1	3.1
	Which	4.6	4.3	4.4	4.1
	Who	4.5	1.5	2.5	1.5
Semantic Arg	Where do	4.0	4.0	4.0	4.0
	Why	3.9	3.5	3.7	3.2
Default	Default	4.1	3.7	3.3	3.2
Negation	Negation	3.9	4.0	4.0	3.7
Average		4.1	3.4	3.5	3.3

The overall interaction between the evaluators and the system is good considering, 72.2% of the reply questions get an answer. Evaluators used complete sentences even though the majority of them still wrote them in lower case. Several evaluators expressed that some of the dialogues were convincing and felt natural, especially when the reply questions are not too long. Take a look at examples 5.1 and 5.2 where follow-up questions are short and straightforward. However, even though the conversation felt natural, one evaluator expected to answer in short sentences: "I feel like I speak with someone and it would be better if we also can answer in short sentences."

`Q: Which sports do you like?`
`A: I love cycling`
`Q: What kind of cycling?` (5.1)
`A: I like cross country`

`Q: Where are you from?`
`A: I am from China`
(5.2)
`Q: Which part of China?`
`A: The south of China`

Another evaluator mentioned that he liked it when the system asked a question using the words from his answers, like examples 5.3 and 5.4. This is in line with the characteristic of follow-up questions which followed up on the topic in the previous turn. Moreover, he also added: "It gave me the sense of connectivity with the machine."

`Q: Do you like reality TV shows? Why or why not? If so, which ones?`
`A: I don't like reality TV shows because I believe most of them only fake programs.`
`Q: Why do you believe most of them only fake programs?`
`A: Because the shows are too good, or contrary too bad to happen in the real world.`

(5.3)

`Q: Where is the most beautiful place you have been?`
`A: In my opinion, Kaliurang is the most beautiful place I ever went.`
`Q: What do you think about Kaliurang?`
`A: It is beautiful, and I love it because I can see the vast scene and enjoy the fresh mountain air.`

(5.4)

Next, we found out that default questions can be good follow-up questions and acceptable for a natural conversation. For instance, one evaluator stated that the follow up question in example 5.5 was natural and appropriate to be asked.

Q: What will be the future of TV shows?

A: The future of TV shows will be closely related to the internet technology

Q: Can you elaborate?

(5.5)

But when the same follow-up question appeared for the third time (see example 5.6), the evaluator revealed that the follow-up question is repetitive.

Q: What were you really into when you were a kid?

A: When I was a kind, I was really into reading

Q: Can you elaborate?

(5.6)

Furthermore, we found out 22 follow-up questions (27.8%) were not answered. We discovered that the evaluators did not respond to the reply question when they felt it was strange or unnatural (example 5.7), asked for an elaboration (example 5.8), or the evaluators did not understand the question (example 5.9). The rest of this section provides error analysis based on their category.

Q: Which sports do you like?

A: I like jogging

Q: Which part of Jogging??

(5.7)

Q: What do you think will be the best time period of your life?

A: The best time period of my life is when I was a child

Q: Could you tell me more?

(5.8)

Q: What would be your perfect weekend?

.A: The perfect weekend could watching TV all day at home.

Q: Why does The perfect weekend watch TV?

(5.9)

5.3.1 POS and NER tag

The majority of errors encountered in this category are due to errors in NER tagging, either because of incorrect input by the user or the auto-capitalization by TrueCase. An example of incorrect input by the user is displayed in 5.10. In this sentence, evaluator incorrectly type `the Netherlands` as `the Netherland` (missing an s). The resulting NER tag is PERSON instead of LOC (location).

`Q: What is the best memory of the Netherlands that you will take back home with you?`

`A: I will remember mostly about the Netherland's weather`

`Q: Who is Netherland?`

(5.10)

Example of error by the TrueCase is shown in 5.11. TrueCase capitalized the first letter of `'infotainment'`, making it tagged as ORG (organization). This is not correct and had an impact on the generated question.

`Q: What's the most underrated or overrated TV show? Why?`

`A: The infotainment is the most overrated TV show`

`Q: Where is Infotainment located?`

(5.11)

Another example is shown in 5.12. In this example, JJ was tagged as 'PRODUCT', which we categorized as 'MISC', and NNP (proper noun) by SpaCy. We could expect that the systems generated the kind of follow-up question `'What do you know about JJ?'`. But JJ was auto-corrected by TrueCase to Jj. It made Jj was no longer recognized as 'MISC' and it was also incorrectly tagged as NNS (plural noun).

`Q: Who is someone who is popular now that you really like?`

`.A: I like JJ because he is a famous singer`

`Q: What kind of Jj?`

(5.12)

These errors caused the evaluators to give low scores to the follow-up questions because they felt inappropriate. This is reflected in the average scores for *What kind* and *Who* questions that are below the borderline (see Table 5.3).

5.3.2 Semantic Arguments

The average scores for the questions in the Semantic Arguments category are above the borderline. Questions that are generated with semantic arguments tend to get a lower score when they felt not natural. An example is illustrated in 5.13.

`Q: What makes you nervous?`

`A: public speaking makes me so nervous`

`Q: Why does Public speaking make you?`

(5.13)

The SRL result for the answer in example 5.13 is '`public speaking (A0) makes (V) me (A1) so nervous (A2)`'. The question template for this sentence does not include A2, making it loses an adverb. It will be more natural if we include A2 into the template, so the follow-up question becomes '`Why does public speaking make you so nervous?`'.

Incorrect input sentences can also cause errors in this category. Example 5.14 illustrates this issue. The evaluator intended to say that his perfect weekend is watching TV all day at home, but his intention was not conveyed correctly because '`The perfect weekend`' in his sentence acted as an agent/causer (A0) as we can see from 5.14 and it could mean that '`The perfect weekend`' watches the TV. It will be better if the agent for this sentence is changed to '`I`'.

`Q: What would be your perfect weekend?`

`A: The perfect weekend (A0) could (AM-MOD) watching (V) TV (A1) all day (AM-TMP) at home (AM-LOC).`

`Q: Why does the perfect weekend watch TV?`

(5.14)

5.3.3 Default Questions

One reason the default questions were given a lower score was that they asked for an elaboration when the evaluators already explained in their answer as shown in example 5.15. The question became boring and felt like it does not interested to the answer, as stated by an evaluator: "The reply is boring because it does not incorporate the detailed answer given and does not show interest to the answer."

`Q: How often do you stay up past 3 a.m.?`

`A: I stay up really late approximately three days a week every weekend.`

`Q: Can you elaborate?`

(5.15)

Another reason was that the evaluator felt that the system was disrespectful because it doubted his answer. An example is given in 5.16.

```
Q: When was the last time you worked incredibly hard?
A: the day I worked hard was yesterday.
Q: Did you?
A: Yes, I did
```

(5.16)

5.3.4 Negation

The average scores for the negation category are above the borderline. However, we found out that we did not manage all kind of negation words. We only handle ‘no’ and ‘not’, but leave out other negation words[4] like ‘rarely’ or ‘never.” Example 5.17 gives an illustration of this. Unfortunately, SENNA also does not label these kinds of words as AM-NEG as we can see in example 5.18. We leave this as future work.

```
Q: How often do you stay up past 3 a.m.?
A: I never do it.
Q: Why do you do it?
```

(5.17)

```
I (A0) never (AM-TMP) watch (V) TV series (A1).
I (A0) rarely (AM-TMP) watch (V) TV series (A1).
```

(5.18)

5.4 Summary

The overall follow-up questions generated by the systems was good considering the average user evaluation scores are above the borderline. Also, 72.2% of the evaluators answered the follow-up questions, indicating that the questions are interesting to answer. Some of the questions are felt convincing and natural especially when the follow-up questions are short and straightforward. The strategy to use the topic from the previous turn as follow-up questions show an indication of a sense of connectivity between the user and the systems.

[4] https://dictionary.cambridge.org/grammar/british-grammar/questions-and-negative-sentences/negation

However, some of the questions generated by the systems are strange and unnatural due to incorrect input, parsing error, or imperfect question templates. One of the causes of incorrect input is because the user writes in lowercase letters. This causes many locations, person, or organization names are not tagged properly. We overcame this situation by using TrueCase to auto-capitalize the source texts. The consequence is many words get auto-capitalized too, causing an error parsing. Another reason is due to grammatical errors. We must emphasize users to use the correct sentence structure.

Question generations in open-domain conversation is a very challenging area. It relies on the correctness of the sentence structure. The sentence structure of the source texts is very influential in generating questions.

Chapter 6

Conclusion and Future Work

In this chapter, the conclusion of this research is discussed in section 6.1 and future work is presented in section 6.2.

6.1 Conclusion

This thesis has presented an approach to generate questions from a text that leverages semantic role labeling (SRL), POS, and NER tagging for text analysis. SENNA is used as a tool to retrieve the SRL and a Python library called SpaCy is used to retrieve POS and NER tags. Although SRL and templates have been used in many question generation approaches, they generally provide answers to the generated questions. We have demonstrated that a template-based method can be used to generate follow-up questions in an open-domain conversational system in which the answers are not available in the source sentences. We formulated the rules to create follow-up questions and group them into three categories: the questions that utilize POS and NER tag, the questions that use semantic arguments, and default questions.

Our evaluation was divided into two parts: question evaluation and user evaluation. In question evaluation, the generated questions were assessed for their grammatical correctness and their relation with the source sentence. The average score of grammar is 4 out of 5 and for the relation 3.6 out of 5. An improvement on the templates was conducted before the interactive evaluation with users.

In the user evaluation, the system acted as a chatbot and made dialogue with the users interactively with a one-turn response (the system only asks one follow-up question in one dialogue). The system asked one opening question, and after the user put their answers, the system replied with a follow-up question. We implemented a simple ranking procedure based on the question templates' final average rating score obtained from the question evaluations to select a question from a set of the generated questions. The overall interaction between the system and the evaluators was satisfactory considering the evaluators answered 72.2% of the follow-up questions. The evaluation results show that some evaluators feel connected to the system while answering the questions. This is a good indication that question generation with follow-up questions is an interesting research area.

Nonetheless, our question generation from text is far from a solved problem. When we conducted the user evaluation, we instructed the evaluators to answer in a complete sentence to prevent the system from generating too many default questions caused by the mismatch between rules and sentence patterns. Naturally, users prefer to answer with short answers as in the casual conversation, which may cause problems for the question generation system. For example, consider the question `‘What TV series do you watch?’`. If users input `‘mister robot’` as the answer, the follow-up question is a default question `‘What do you mean?’`, which could be perceived that the system does not understand the users' answer. But if users input `‘I like to watch mister robot’`, the question could be more interesting such as `‘When do you usually watch mister robot?’`. In question generation system, more complete sentences are more beneficial to create a meaningful question.

The generated questions also have difficulty expanding the conversation topics since the questions are very related to the input from users. Another shortcoming is that each question includes a portion of the source sentence, which sometimes does not match the dialogue context. An example is shown in 6.1. The follow-up question in this example ignores the explanation of the particular sport, which is the main point of the answer. As pointed out by [8], the majority of the semantic tools only take one sentence as the input, and it can eliminate the context of the conversation.

```
Q: Which sports do you like?
A: I like a lot of sports, particularly table tennis.
Q: Why do you like a lot of sports?
A: I like a lot of sports because they are fun
```
(6.1)

6.2 Future Work

Further work might need to handle negation words such as `‘rarely’` or `‘never’` properly, since SENNA does not label these words as AM-NEG (modifier negation). The alternative is to opt for another semantic role labeler tool, or a better algorithm to take a more comprehensive approach.

The next necessary step in term of the research presented in this thesis would be improving the ranking mechanism. A more robust mechanism is needed to select a question from a set of the generated questions. For example, to calculate the topic similarity between the generated question and the source sentence using probabilistic topic modeling, and to automatically judge the syntactic correctness of each generated question as demonstrated by Chali et al. [4].

Finally, more work can be done to improve the quality of questions generated from a sentence by leveraging multiple clauses for question generation, such as checking discourse connectives between clauses like “so”. SENNA divides a sentence into one or more clauses. One question is generated from one clause based on the SRL parsing results by SENNA. This has implications for the generated questions. There is a chance that the answer to the follow-up question is in another clause even though optional arguments have been taken into account. For example, consider the sentence `‘I used to live in Guatemala, so I speak Spanish.’` One of the generated questions to this sentence is `‘Why do you speak Spanish?’` This is not a good question since the reason has been mentioned in the sentence. To create the question `‘Why’`, we check the absence of AM-CAU (cause) and AM-PNC (purpose). AM-PNC is used to label purpose clauses, to show the motivation for some actions. Similar to AM-PNC, AM-CAU indicates the reason for an action. However, this is not the case for the sentence `‘I used to live` `in Guatemala, so I speak Spanish,'` since the SRL parse result does not contain AM-CAU and AM-PNC as shown below:

```
I (A0) used (AM-MOD) to live (V) in Guatemala (AM-LOC)
I (A0) speak (V) Spanish (A1)
```

Bibliography

[1] Abdul-Kader, S. A., and Woods, J. Survey on chatbot design techniques in speech conversation systems. *International Journal of Advanced Computer Science and Applications 6*, 7 (2015).

[2] Al Omran, F. N. A., and Treude, C. Choosing an nlp library for analyzing software documentation: a systematic literature review and a series of experiments. In *Proceedings of the 14th International Conference on Mining Software Repositories* (2017), IEEE Press, pp. 187–197.

[3] Bickmore, T., and Cassell, J. *Social Dialogue with Embodied Conversational Agents*. Springer Netherlands, Dordrecht, 2005, pp. 23–54.

[4] Chali, Y., and Hasan, S. A. Towards topic-to-question generation. *Computational Linguistics 41*, 1 (2015), 1–20.

[5] Chiticariu, L., Li, Y., and Reiss, F. R. Rule-based information extraction is dead! long live rule-based information extraction systems! In *Proceedings of the 2013 conference on empirical methods in natural language processing* (2013), pp. 827–832.

[6] Clark, L., Pantidi, N., Cooney, O., Doyle, P., Garaialde, D., Edwards, J., Spillane, B., Murad, C., Munteanu, C., Wade, V., et al. What makes a good conversation? challenges in designing truly conversational agents. *arXiv preprint arXiv:1901.06525* (2019).

[7] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. Natural language processing (almost) from scratch. *Journal of machine learning research 12*, Aug (2011), 2493–2537.

[8] Fasya, E. L. Automatic question generation for virtual humans. Master's thesis, University of Twente, 2017.

[9] Flor, M., and Riordan, B. A semantic role-based approach to open-domain automatic question generation. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications* (2018), pp. 254–263.

[10] Gao, J., Galley, M., and Li, L. Neural approaches to conversational ai. *Foundations and Trends® in Information Retrieval 13*, 2-3 (2019), 127–298.

[11] Greenbaum, S., and Nelson, G. *An introduction to English grammar, Second Edition*. Pearson Education, 2002.

[12] Heilman, M. *Automatic factual question generation from text*. PhD thesis, Language Technologies Institute School of Computer Science Carnegie Mellon University, 2011.

[13] Higashinaka, R., Imamura, K., Meguro, T., Miyazaki, C., Kobayashi, N., Sugiyama, H., Hirano, T., Makino, T., and Matsuo, Y. Towards an open-domain conversational system fully based on natural language processing. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers* (2014), pp. 928–939.

[14] Huang, K., Yeomans, M., Brooks, A. W., Minson, J., and Gino, F. It doesn't hurt to ask: Question-asking increases liking. *Journal of personality and social psychology 113*, 3 (2017), 430.

[15] Jurafsky, D., and Martin, J. H. *Speech and language processing*, vol. 3rd. Draft, August 28, 2017.

[16] Kirschner, M., and Bernardi, R. Exploring topic continuation follow-up questions using machine learning. In *Proceedings of Human Language Technologies: Annual Conference of the North American Chapter of the Association for Computational Linguistics* (2009), Association for Computational Linguistics, pp. 13–18.

[17] Lindberg, D. L. Automatic question generation from text for self-directed learning. Master's thesis, Simon Fraser University, 2013.

[18] Mannem, P., Prasad, R., and Joshi, A. Question generation from paragraphs at upenn: Qgstec system description. In *Proceedings of QG2010: The Third Workshop on Question Generation* (2010), pp. 84–91.

[19] Mazidi, K., and Nielsen, R. D. Linguistic considerations in automatic question generation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (2014), vol. 2, pp. 321–326.

[20] Mazidi, K., and Tarau, P. Infusing nlu into automatic question generation. In *Proceedings of the 9th International Natural Language Generation conference* (2016), pp. 51–60.

[21] Mihalcea, R., and Tarau, P. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing* (2004).

[22] Mostafazadeh, N., Misra, I., Devlin, J., Mitchell, M., He, X., and Vanderwende, L. Generating natural questions about an image. *arXiv preprint arXiv:1603.06059* (2016).

[23] Schulman, D., and Bickmore, T. Persuading users through counseling dialogue with a conversational agent. In *Proceedings of the 4th international conference on persuasive technology* (2009), ACM, p. 25.

[24] Shang, L., Lu, Z., and Li, H. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (Beijing, China, July 2015), Association for Computational Linguistics, pp. 1577–1586.

[25] Su, M.-H., Wu, C.-H., Huang, K.-Y., Hong, Q.-B., and Huang, H.-H. Follow-up question generation using pattern-based seq2seq with a small corpus for interview coaching. *Proc. Interspeech 2018* (2018), 1006–1010.

[26] Sugiyama, H. *Building Open-domain Conversational Agent by Statistical Learning with Various Large-scale Corpora.* PhD thesis, Nara Institute of Science and Technology, 2016.

[27] Wang, Y., Liu, C., Huang, M., and Nie, L. Learning to ask questions in open-domain conversational systems with typed decoders. *arXiv preprint arXiv:1805.04843* (2018).

[28] Woo, S., Li, Z., and Mirkovic, J. Good automatic authentication question generation. In *Proceedings of the 9th International Natural Language Generation conference* (2016), pp. 203–206.

[29] Yao, X., Bouma, G., and Zhang, Y. Semantics-based question generation and implementation. *Dialogue & Discourse 3*, 2 (2012), 11–42.

[30] Yao, X., Tosch, E., Chen, G., Nouri, E., Artstein, R., Leuski, A., Sagae, K., and Traum, D. Creating conversational characters using question generation tools. *Dialogue & Discourse 3*, 2 (2012), 125–146.

[31] Yao, X., and Zhang, Y. Question generation with minimal recursion semantics. In *Proceedings of QG2010: The Third Workshop on Question Generation* (2010), Citeseer, pp. 68–75.

Appendix A

POS Tagging

Table A.1: Alphabetical list of part-of-speech tags used in the Penn Treebank Project

No	**Tag**	**Description**	**Example**
1	CC	Coordinating conjunction	and
2	CD	Cardinal number	1, third
3	DT	Determiner	the
4	EX	Existential *there*	there is
5	FW	Foreign word	les
6	IN	Preposition or subordinating conjunction	in, of, like
7	JJ	Adjective	green
8	JJR	Adjective, comparative	greener
9	JJS	Adjective, superlative	greenest
10	LS	List item marker	1)
11	MD	Modal	could, will
12	NN	Noun, singular or mass	table
13	NNS	Noun, plural	tables
14	NNP	Proper noun, singular	John
15	NNPS	Proper noun, plural	Vikings
16	PDT	Predeterminer	both the boys
17	POS	Possessive ending	friend’s
18	PRP	Personal pronoun	I, he, it
19	PRP$	Possessive pronoun	my, his

No	**Tag**	**Description**	**Example**
20	RB	Adverb	however, usually
21	RBR	Adverb, comparative	better
22	RBS	Adverb, superlative	best
23	RP	Particle	give up
24	SYM	Symbol	. ! ?
25	TO	Infinitive ‘to’	togo
26	UH	Interjection	ah, oops
27	VB	Verb, base form	be, eat
28	VBD	Verb, past tense	was, ate
29	VBG	Verb, gerund or present participle	being, eating
30	VBN	Verb, past participle	been, eaten
31	VBP	Verb, non-3rd person singular present	am, eat
32	VBZ	Verb, 3rd person singular present	is, eats
33	WDT	Wh-determiner	which
34	WP	Wh-pronoun	who, what
35	WP$	Possessive wh-pronoun	whose
36	WRB	Wh-adverb	where, when

Appendix B

Question Templates

This appendix provides the full list of templates used for follow-up question generation. Questions templates specify the text, verb forms, and semantic arguments from the source sentence to form the question. A question will be generated according to the question templates every time there's a matching rule (if required arguments are present or if filter conditions are fulfilled). Table B.1 provides required fields and filter conditions in a rule adapted from [19].

TABLE B.1: Required fields and filter conditions in a rule

Field	**Meaning**
Ax*	Sentence contains any Ax
!Ax*	Sentence does not contain any Ax
Ax	Sentence contains an Ax
!Ax	Sentence does not contain an Ax
Ax[NER]	Ax contains a NER tag
Ax[POS]	Ax contains a POS tag
Ax[head=POS]	Ax starts with POS tag
seq[x]	SRL parse results are sequence of x
V=present	Verbs are in the present form
V=past	Verbs are in the past form
V=root	Verb must in the root form

Table B.2 presents initial question templates used in Evaluation 1. Each row consist of an ID, rule (each field is separated by a semicolon (;)), and a question template.

Table B.2: Initial question templates

No	ID	Rules	Question Templates
	Category: POS & NER Tag		
1	WHC1	!A*; !AM*; [nns]	Which [lemma(nns)] is your favorite?
2	WHC2	!A*; !AM*; [LOC]	Which part of [LOC]?
3	WHC3	A1[head=dt nns]	Which [lemma (nns)] do you like best?
4	WHC4	!A0; !A1; A*[LOC]; !AM-LOC	Which part of [LOC]?
5	WHO1	!A*; !AM*; [PER]	Who is [PER]?
6	WHO2	A*|AM*[PER]	Who is [PER]?
7	WHR1	!A0; !A1; A*[in nnp]; !AM-LOC	Where is [nnp]?
8	WHR2	!A*; !AM*; [ORG]	Where is [ORG] located?
9	WHR3	A1[LOC]	Where in [LOC]?
10	WHR5	AM-LOC[LOC]	Where in [LOC]?
11	WHR6	AM[LOC]	Where in [LOC]?
12	WHR7	A*|AM*[ORG]	Where is [ORG] located?
13	WHT1	!A*; !AM*; [MISC]	What do you know about [MISC]?
14	WHT2	A1[ORG]	What is [ORG]?
15	WHT3	A*[ORG]	What is [ORG]?
16	WHT4	!A1; A*[LOC];	What do you think about [LOC]?
17	WHT5	AM-DIR[LOC]	What do you think about [LOC]?
18	WHT6	A*|AM*[LOC]	What do you think about [LOC]?
19	WHT7	A*|AM*[MISC]	What do you know about [MISC]?
20	WHT8	A*|AM*[nnp]	What do you think about [nnp]?
21	WKD1	A1[head=nn]	What kind of [nn]?
22	WKD2	A1[cd nns]; !AM-TMP	What kind of [lemma(nns)]?
23	WKD3	A1[dt nns]; !AM-TMP	What kind of [nns]?
24	WKD4	A1[head=nns]; !AM-MNR	What kind of [nns]?
25	WKD5	A1[head=in nn]	What kind of [nn]?
26	WKD6	A1[head=in nns]	What kind of [lemma(nns)]?
	Category: Semantic arguments		
27	HOW1	A0; V=past; AM-TMP	How did you [V=root] [AM-TMP]?
28	HOW2	A0; V=present; AM-TMP	How do you [V=root] [AM-TMP]?
29	HOW3a	V=past; !A2; !A3; !A4; !AM-MNR; !AM-DIS	How did [A0] [V=root] [A1]?
30	HOW3b	V=present; !A2; !A3; !A4; !AM-MNR; !AM-DIS	How [aux] [A0] [V=root] [A1]?
31	WHR4	V; A1[head=to vb]; A*[!LOC]; !AM-LOC	Where do you [V] [A1]?
32	WHY1	A1[head=in]	Why do you [V=root] [A1]
33	WHY2	A1[head=vbg]; !AM-PNC, !AM-CAU	Why are you [A1]?
34	WHY4a	V=past; !AM-PNC, !AM-CAU	Why did [A0] [V=root] [A1]?
35	WHY4b	V=present; !AM-PNC, !AM-CAU	Why do [A0] [V=root] [A1]?

No	ID	Rules	Question Templates
36	WHN1	V=past; seq[AM-TMP, V, A1]	When did you [V=root] [A1]?
37	WHN3a	!AM-TMP; V=past	When did you [V=root] [A1]?
38	WHN3b	!AM-TMP; V=present	When do you usually [V=root] [A1]?
	Category: Default questions		
39	DEF1	Mismatch index exception	Can you elaborate?
40	DEF2	!A*; !AM*; !nns; !nn; !LOC; !PER; !ORG; !MISC	What do you mean?
41	DEF4	!A*; !AM*; seq[prp, vbp, dt]	How do you like that so far?
42	DEF5	!A0; !A1; V=past	How was it?
43	DEF6	AM-LOC[!LOC]	Is that a good or a bad thing?
44	DEF7	A0	How is that for you?
45	DEF8a	!A*; AM-ADV; AM-TMP; V=past	How was it?
46	DEF8b	!A*; AM-ADV; AM-TMP; V=present	How is it?
47	DEF9	!A*; AM-LOC; !AM-TMP; V=past	When was that?
48	DEF3	None of the above conditions	Can you elaborate?

Table B.3 presents improved question templates used in Evaluation 2.

Table B.3: Improved question templates

No	ID	Rules	Question Templates
	Category: POS & NER Tag		
1	WHC1	!A*; !AM*; [nns]	Which [nns] do you like?
2	WHC2	!A*; !AM*; [LOC]	Which part of [LOC]?
3	WHC3	A1[head=dt nns]	Which [noun] do you like?
4	WHC4	!A0; !A1; A*[LOC]; !AM-LOC	Which part of [LOC]?
5	HOW4	!A*; !AM*; [LOC]	How is the weather in [LOC]?
6	WHO1	!A*; !AM*; [PER]	Who is [PER]?
7	WHO2	A*|AM*[PER]	Who is [PER]?
8	WHR1	!A0; !A1; A*[in nnp]; !AM-LOC	Where is [propn]?
9	WHR2	!A*; !AM*; [ORG]	Where is [ORG] located?
10	WHR3	A1[LOC]	Where in [LOC]?
11	WHR5	AM-LOC[LOC]	Where in [LOC]?
12	WHR6	AM[LOC]	Where in [LOC]?
13	WHR7	A*|AM*[ORG]	Where is [ORG] located?
14	WHT1	!A*; !AM*; [MISC]	What do you know about [MISC]?
15	WHT2	A1[ORG]	What is [ORG]?
16	WHT3	A*[ORG]	What is [ORG]?
17	WHT4	!A1; A*[LOC];	What do you think about [LOC]?
18	WHT5	AM-DIR[LOC]	What do you think about [LOC]?

No	**ID**	**Rules**	**Question Templates**
19	WHT6	A*|AM*[LOC]	What do you think about [LOC]?
20	WHT7	A*|AM*[MISC]	What do you know about [MISC]?
21	WHT8	A*|AM*[nnp]	What do you think about [propn]?
22	WHT9	!A0; !A1; A*[in nnps]; !AM-LOC	What do you think about [propn]?
	Category: Semantic Argument		
23	HOW3a	V=past; !A2; !A3; !A4; !AM-MNR; !AM-DIS	How did [A0] [V=root] [A1(!comma)]?
24	HOW3b	V=present; !A2; !A3; !A4; !AM-MNR; !AM-DIS	How [aux] [A0] [V=root] [A1(!comma)]?
25	WHN1	V=past; seq[AM-TMP, V, A1]	When did [A0] [V=root] [A1(!comma)]?
26	WHN3a	!AM-TMP; V=past	When did [A0] [V=root] [A1(!comma)]?
27	WHN3b	!AM-TMP; V=present	When [aux] [A0] usually [V=root] [A1(!comma)]?
28	WHN4	!AM-TMP; AM-LOC[!LOC]; V=past	When was that?
29	WHR4a	V=past; A1[head=to vb]; A*[!LOC]; !AM-LOC	Where did [A0] [V=root] [A1(!comma)]?
30	WHR4b	V=present; A1[head=to vb]; A*[!LOC]; !AM-LOC	Where [aux] [A0] [V=root] [A1(!comma)]?
31	WHY1	A1[head=in]	Why do you [V=root] [A1]
32	WHY2	A1[head=vbg]; !AM-PNC, !AM-CAU	Why are you [A1(!comma)]?
33	WHY3	seq[prp, vbp, nn]|seq[prp, vbp, nns]	Why [aux] [prp] [V] [noun]?
34	WHY4a	V=past; !AM-PNC, !AM-CAU	Why did [A0] [V=root] [A1(!comma)]?
35	WHY4b	V=present; !AM-PNC, !AM-CAU	Why do [A0] [V=root] [A1(!comma)]?
36	WKD1	A1[head=nn]	What kind of [noun]?
37	WKD2	A1[cd nns]; !AM-TMP	What kind of [noun]?
38	WKD3	A1[dt nns]; !AM-TMP	What kind of [noun]?
39	WKD4	A1[head=nns]; !AM-MNR	What kind of [noun]?
40	WKD5	A1[head=in nn]	What kind of [noun]?
41	WKD6	A1[head=in nns]	What kind of [noun]?
	Category: Negation		
42	NEG6	A0; !A1; !AM-PNC; !AM-CAU; AM-MOD	Why [AM-MOD]n't [A0]?
43	NEG4	!A0; A1; A2; !AM-PNC; !AM-CAU; AM-MOD	Why [AM-MOD]n't [A1] [V=root] [A2]?
44	NEG5	A0; A1; !AM-PNC; !AM-CAU; AM-MOD	Why [AM-MOD]n't [A] [V=root] [A1]?
45	NEG1a	A0; A1; !AM-PNC; !AM-CAU; V=past	Why didn't [A0] [V=root] [A1]?
46	NEG1b	A0; A1; !AM-PNC; !AM-CAU; V=present	Why [aux]n't [A0] [V=root] [A1]?
47	NEG2a	!A0; A1; A2; !AM-PNC; !AM-CAU; V=past	Why didn't [A1] [V=root] [A2]?
48	NEG2b	!A0; A1; A2; !AM-PNC; !AM-CAU; V=present	Why [aux]n't [A1] [V=root] [A2]?
49	NEG3a	A0; !A1; !AM-PNC; !AM-CAU; V=past	Why didn't [A0]?
50	NEG3b	A0; !A1; !AM-PNC; !AM-CAU; V=present	Why [aux]n't [A0]?
	Category: Default		
51	DEF1	Mismatch index exception	Can you elaborate?
52	DEF2	!A*; !AM*; !nns; !nn; !LOC; !PER; !ORG; !MISC	What do you mean?
53	DEF4	!A*; !AM*; seq[prp, vbp, dt]	How do you like that so far?

No	**ID**	**Rules**	**Question Templates**
54	DEF5	!A0; !A1; V=past	How was it?
55	DEF6	AM-LOC[!LOC]	Is that a good or a bad thing?
56	DEF7a	A0; V=past	Did [A0]?
57	DEF7b	A0; V=present	How is that going for [A0]?
58	DEF8a	!A*; AM-ADV; AM-TMP; V=past	How was it?
59	DEF8b	!A*; AM-ADV; AM-TMP; V=present	How is it?
60	DEF3	None of the above conditions	Can you elaborate?