AUTOMATIC QUESTION GENERATION USING NATURAL LANGUAGE PROCESSING TECHNIQUES

A Thesis Submitted to
the Graduate School of Engineering and Sciences of
İzmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in Computer Engineering

by
Onur KEKLİK

July 2018
İZMİR

We approve the thesis of **Onur KEKLİK**

Examining Committe Members:

Prof. Dr. Oğuz DİKENELLİ
Department of Computer Engineering, Ege University

Asst. Prof. Dr. Zerrin IŞIK
Department of Computer Engineering, Dokuz Eylul University

Asst. Prof. Dr. Serap ŞAHİN
Department of Computer Engineering, Izmir Institute of Technology

---

Asst. Prof. Dr. Tuğkan TUĞLULAR
Department of Computer Engineering, Izmir Institute of Technology

---

Asst. Prof. Dr. Selma TEKİR
Department of Computer Engineering, Izmir Institute of Technology

10 July 2018

Asst. Prof. Dr. Tuğkan TUĞLULAR
Supervisor, Department of Computer Engineering, Izmir Institute of Technology

---

Asst. Prof. Dr. Selma TEKİR
Co-Supervisor, Department of Computer Engineering, Izmir Institute of Technology

---

Assoc. Prof. Dr. Yusuf Murat ERTEN
Head of the Department of Computer Engineering

Prof. Dr. Aysun SOFUOĞLU
Dean of the Graduate School of Engineering and Sciences

ACKNOWLEDGMENTS

I would like to thank my thesis advisors Asst. Prof. Dr. Tuğkan TUĞLULAR and Asst. Prof. Dr. Selma TEKİR of the Computer Engineering Department at İzmir Institute of Technology. The door to Dr. TUĞLULAR and Dr. TEKİR office was always open whenever I had difficulties or had a question about my research or writing. They consistently guide me throughout the whole process.

I would also like to thank the experts who were involved in the human evaluation study for this research project. Without their passionate participation and input, the human evaluation study could not have been successfully completed.

Finally, I would like express my gratitude to my family for supporting me spiritually throughout researching and writing this thesis. Without their encouragement and support, this accomplishment would not have been completed.

ABSTRACT

AUTOMATIC QUESTION GENERATION USING NATURAL LANGUAGE PROCESSING TECHNIQUES

This thesis proposes a new rule based approach to automatic question generation. The proposed approach focuses on analysis of both syntactic and semantic structure of a sentence. The design and implementation of the proposed approach are also explained in detail. Although the primary objective of the designed system is question generation from sentences, automatic evaluation results shows that, it also achieves great performance on reading comprehension datasets, which focus on question generation from paragraphs. With respect to human evaluations, the designed system significantly outperforms all other systems and generated the most natural (human-like) questions.

ÖZET

DOĞAL DİL İŞLEME TEKNİKLERİNİ KULLANARAK OTOMATİK SORU ÜRETME

Bu tez otomatik soru üretimi için yeni bir kural tabanlı yaklaşım önermektedir. Önerilen yaklaşım, bir cümlenin hem sözdizimsel hem de semantik yapısının analizine odaklanmaktadır. Ayrıca, önerilen yaklaşımın tasarımı ve uygulanması ayrıntılı olarak açıklanmıştır. Tasarlanan sistemin temel amacı cümlelerden soru üretmek olmasına rağmen, otomatik değerlendirme sonuçları, sistemin anlama becerisi gerektiren paragraflar üzerinde de büyük bir performans sergilediğini gösterdi. İnsan değerlendirmelerine gelince, tasarlanan sistem diğer tüm sistemlerden önemli ölçüde daha iyi bir performans gösterdi ve en doğal (insan benzeri) soruları üretti.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS

QG . . . . . Question Generation
QA . . . . . Question Answering
NLU . . . . . Natural Language Understanding
QGSTEC . . . . . Question Generation Shared Task Evaluation Challenge
MT . . . . . Machine Translation
TREC . . . . . Text REtrieval Conference
SQuAD . . . . . Stanford Question Answering Dataset
ROUGE . . . . . Recall-Oriented Understudy for Gisting Evaluation
POS . . . . . Part-Of-Speech Tagging
NER . . . . . Named Entity Recognition
GPE . . . . . Geographical Location
SRL . . . . . Semantic Role Labeling

CHAPTER 1

INTRODUCTION

Humans have curios nature. We ask question to gain more knowledge and try to link them with other things that we know. Our daily lives include asking questions in conversations. For example, student questions play an important role in their learning process and help them to learn more from their teachers. In addition, teacher questions help students to assess their performance. In a nutshell, questions are one of the primary sources of learning from daily conversations to verbal tutorings and assessments.

Most of learners are not good at asking questions. Dunlosky and Graesser (1998) states that learners have a problem with identifying their own knowledge deficits. Therefore, they ask very few questions. Automation of question generation systems can help learners to find out their own knowledge gaps by helping them to reach their valuable inquiries. Learners are not the only one who encounter limitations in their question generation skills. Other examples are:

- Tutors have trouble to generate good hints to encourage students think and talk (Chi et al., 2001; Corbett & Mostow, 2008; DiPaolo et al., 2004). Tutors need to assess the students knowledge by asking good questions and address their knowledge deficits (Corbett & Mostow, 2008).
- Users of Google and other search engine users tend to input only few words rather than full sentences and queries (Lin, 2008; Marciniak, 2008). Guided question reformulation is needed to quickly retrieve answers.
- Frequently Asked Questions (FAQ) are usually developed by designers of the system, rather than real questions that are asked by clients.

Automation of question generation systems help automated question answering systems such as IBM Watson to perform self training (IBM Watson Ecosystem, 2014). Instead of relying on human experts to manually define ground truth answers from the questions, automatic question generation systems automatize this process. Intelligent tutoring systems can also get benefit from that. Rather relying on human experts to manually extract questions from study materials, each end user can define its own tutoring system automatically from the study material. Finally, question generation systems help the development of annotated datasets for question answering and reading comprehension.

Question generation is a fundamental activity in educational learning. Questions serve to different levels of complexity in the educational learning process. The task of question generation in computational linguistics defines two categories to address deep and shallow questions. In the first place, question generation for reading comprehension uses a paragraph as an information source and creates deep questions that test its understanding. Question generation from a sentence, on the other hand, poses factual questions related, which are considered as shallow, to the given input sentence.

Most of the question generation systems tackles the problem with the rule based approach. In this approach, sentence-to-question transformation is performed by applying rules or templates to the syntactic representation of the given input sentence. However, syntactic representations are not sufficient to reach high-level abstractions in question generation. There is a strong need to consider the semantic roles of words to increase the comprehension level of generated questions. Thus, question generation should be enriched by the process of understanding the meaning behind a sentence. Mazidi and Tarau (2016) highlights this fact by stating that natural language understanding (NLU) is a missing piece of the puzzle in question generation.

This work proposes a rule based approach to question generation from sentence. To be specific; dependency based, Named Entity Recognition (NER) based, and semantic role (SRL) labeling based templates/rules are used. In terms of rules, our contribution can be explained as follows:

1. For dependency based rules, the following patterns are newly added:
    - S-V-oprd is an object predicate that defines the subject.
    - S-V-xcomp is an open clausal complement without an internal subject.
    - S-V-ccomp (clausal complement) is a clause with an internal subject.
2. NER based rules, which are S-V-number, S-V-location, S-V-date, S-V-person, are added.
3. More importantly, new semantic role labeling based templates/rules are constructed:
    - S-V-ARGM-CAU: cause clause.
    - S-V-ARGM-MNR: manner marker.
    - S-V-ARGM-PNC: purpose clause.
    - S-V-ARGM-LOC: locative.
    - S-V-ARGM-TMP: temporal marker.

This work mainly aims to generate more comprehensive questions by exploiting the semantic roles of words. Figure 1.1 shows the illustrative example of yes/no question generation. Section 3.2 describes this process in detail. By using semantic role labeling, sentence is decomposed into its parts. Then, tense is detected and verb is converted to its base form. With construction stage, question generation becomes complete.

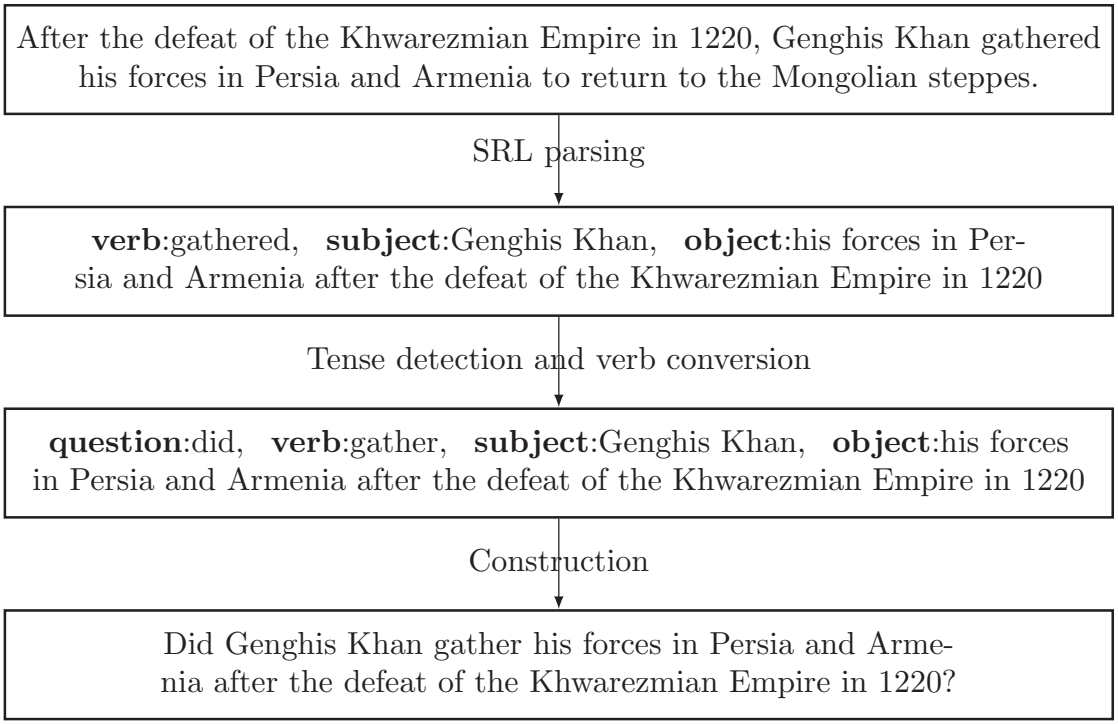

Figure 1.1. Illustrative example of Yes/No question generation.

To test the effectiveness of the proposed approach, it is compared against two state-of-the-art systems: Du et al.'s (2017) learning-based system for question generation for reading comprehension and the best rule-based system by Heilman and Smith (2011). Our automatic evaluation through objective neural translation metrics show that our system has superior performance and outperforms H&S and Du's systems in BLEU-2, METEOR, and ROUGE-L metrics. The superior performance of the proposed approach especially in METEOR metric can be attributed to its recall based nature. By diversifying and extending the rule sets, our approach produces an expanded set of questions out of those that can possibly be asked.

We performed human evaluation as well. In human evaluations, the designed system significantly outperforms H&S and Du's systems and generated the most natural (human-like) questions.

The organization of this thesis is as follows:

- Chapter 2 provides information about related work and parsers that are used by the designed system.
- Chapter 3 explains predefined rules (templates) that are used for generating questions. Each type of rule is categorized with respect to the parsing method(s) they used. Also, in this chapter, the designed system is explained in detail, algorithms and flowcharts are given.
- Chapter 4 discusses both automatic evaluation and human evaluation results. These results are compared with other competitive systems.
- Final chapter gives conclusion and discusses on directions for future work.

CHAPTER 2

BACKGROUND

2.1. Related Work

Recently, Question Generation (QG) and Question Answering (QA) in the field of computational linguistics have got enormous attention from the researchers (Rus and Graesser, 2009). Twenty years ago, receiving answers to the same questions would take hours or weeks through documents and books. After the computers and internet, wealth of information becomes available and this field holds great promise for making sophisticated question asking and answering facilities mainstream in the future.

There are various types of questions and researchers proposed different taxonomies for organizing them. Bloom (1956) defined taxonomy of education objectives. While ”knowledge” level focuses on facts, ”comprehension” level involves deeper understanding. Similar to Bloom's objectives, Rus and Graesser (2009) divided questions into two categories, namely deep questions and shallow questions. If a learner wants to acquire difficult scientific and technical material, deep questions (such as why, why not, how, what-if, what-if-not) can be asked. These questions involve more logical thinking than shallow questions. Conversely, shallow questions focus more on facts (such as who, what, when, where, which, how many/much and yes/no questions). While some studies shows strong advantages for asking higher-level questions, these are discussed by Redfield and Rousseau (1981). Others shows no significant different between asking shallow and deep questions. Deep and shallow questions may supplement each other (Heilman, 2011). While lower-level questions ensure learner to acquire basic knowledge about the subject, higher-level questions let learner to gather various pieces of information to formulate an answer.

Pioneering work in QG is done by Wolfe (1976), who demonstrated the automatically generated questions could be as effective as human generated questions. After that, sporadic researches are done by Kunichika et al. (2004), Mitkov et al. (2006) and Rus et al. (2007). After workshop on the Question Generation Shared Task and Evaluation Challenge (Rus & Graesser, 2009), QG have got enormous attention from the researchers.

In 2011, Heilman addressed various question generation challenges in his paper:

- Syntactic challenges: First syntactic challenge occurs when using NLP tools (parsers)

for analyzing syntactic structure of the sentence. Sadly, current parsers are imperfect and sometimes they produce incorrect parse results. Thus, any QG system that relies on an incorrect parse result is most likely to generate ungrammatical questions. Complex syntactic constructions is an another problem. Even if a parser gives an accurate syntactic representation of a sentence, information extraction or transforming that structure becomes nontrivial.

- Lexical challenges: For lexical challenges, Heilman addressed mapping answers to question word is an another issue. Especially deciding between who and what question is problematic. We address this issue and propose solution in Section 3.1.2. Another lexical challenge is non-compositionality. In English, meaning of phrases is not always just a simple accumulation of its component words. Multiword expressions are one of the active research areas in NLP (Sag et al., 2002). It is a problem because most of syntactic parsers represents each word token independently. So, this makes it difficult to detect such phrases such as idioms. For instance, the sentence ”Mary has to learn to bite the bullet and face her fears of flying” results in the generated question: ”What does Mary have to learn to bite?”. We also address this issue and propose solution in Section 3.2.
- Discourse challenges: For discourse challenges, Heilman addressed information that is conveyed from one sentence to other is an another problematic issue. Consider the second sententece in the following example: ”The American professor Robert H. Goddard had worked on developing solid-fuel rockets since 1914, and demonstrated a light battlefield rocket to the US Army Signal Corps only five days before the signing of the armistice that ended World War I. He also started developing liquid-fueled rockets in 1921.” If we generate questions from the second sentence, with using simple transformations, one of the generated questions will be ”When did he start developing liquid-fueled rockets?” If this question specifically asked a reader after he immediately reads this paragraph, this question can be valid. However, most of the time these type of questions is vague and out of context, since there are many possible answers.

Heilman (2011) also produced a system that generates factual questions for educational instructions. Sentence simplification in his system removes complex syntactic structures, that could be used to generate questions. His work focuses on quality over quantity. However, for IBM Watson this approach is not optimal. When defining ground truths for automatic question generation systems, main purpose is to cover the corpus material as large as possible. So, quality of generated questions have less concern (Yates,

2016).

Heilman and Smith (2011) use a multi-step process to generate factual questions from text. The process begins with NLP transformations for the input sentence. Then, manually encoded transformation rules are applied for sentence-to-question conversion, and finally a linear regression based ranker assigns acceptability scores to questions to eliminate the unacceptable ones. This approach increased their percentage of acceptable questions from 23% to 49%. The ranker approach is also tried by other researchers, but with less success.

Most of the prior works mentioned above arrange sentence constituents with respect to grammar rules to generate as many possible questions as they can. In contrast, Mazidi and Tarau (2016) introduced NLU-approach that focuses on constituent patterns in a sentence. These patterns are key to detect the type of question that should be asked. They also used multiple parsers (both syntactic and semantic parsers) instead of depending on only one because each parser tells its own particular viewpoint about the sentence. In their evaluation of the top 20 questions, their system generated 71% more acceptable questions than other state of the art question generation systems by augmenting the generation process with NLU techniques.

Lubutov et al. (2015) used a completely different approach. They used crowd sourcing method to generate deep comprehension question templates. First, they obtain the high-level question templates from the crowd. Then, they retrieve the subset of collected templates. For example, category-section pairs for an article about Albert Einstein contains (Person, Early life), (Person, Awards), and (Person, Political views). Articles about persons have similar subsections, so that templates formed for one person should transfer reasonably well to others. Their relevance classifier decides on whether category-section pairs match or not. However, in some cases, it transfers irrelevant content and raises the false positive errors.

Du et al. (2017) used another innovative approach to generate questions for reading comprehension. They used a neural language model with a global attention mechanism to generate questions. They study several variations of this model, from sentence focused models to paragraphs, reading passages and other variations to determine the importance of pre-trained vs. learned word embeddings. Moreover, in their approach, they don't rely on hand-crafted rules.

The first Question Generation Shared Task Evaluation Challenge (QGSTEC, 2010) is one of the campaigns that follows the same tradition of STECs (such as Text REtrieval Conference, TREC) in Natural Language Processing. The campaign consists of two tasks.

The first task focuses on question generation from paragraphs, whereas the second task focuses on question generation from sentences. Data sets, evaluation criteria and guidelines are prepared with respect to these tasks. For the second task, input sentences were selected from Wikipedia, OpenLearn and Yahoo! Answers (30 inputs from each source). Participants were also provided with the list of target question types (who, where, when, which, what, why, how many/long, yes/no) that they need to generate. Finally, human evaluators evaluate the submitted questions according to evaluation criteria, which are relevance, question type (who/what/why), syntactic correctness, fluency, ambiguity and variety.

On the other hand, using human evaluators to evaluate machine generated questions is a time and resource consuming process. However, automatic evaluation metrics are key to manage this process much efficiently. Since the release of IBM's BLEU metric (Papineni et al, 2002) and the closely related NIST metric (Doddington, 2002), automatic evaluation metrics have been widely recognized and extensively used by machine translation (MT) community. Compared to the human evaluations, evaluating an MT system using such automatic metrics is a cheaper, faster and time-saving. This is why Du et al. (2017) used BLEU, METEOR and ROUGE-L metrics to evaluate their neural question generation system. They also perform human evaluations to complement their results.

BLEU metric is first proposed by IBM (Papineni et al, 2002). It is the exact matches of words and matches against a set of reference translations for greater variety of expressions. It calculates geometric average of the n-gram scores (size 1 to 4) for precisions. With respect to n-gram scores, BLEU metric named as BLEU-1, BLEU-2, BLEU-3 or BLEU-4. It has no recall, but uses exponential brevity penalty to reduce the score of overly short sentences in order to compensate for recall.

METEOR metric is first proposed by Denkowski and Lavie (2009). It is a recall oriented metric, which combines recall and precision as weighted score components. It is designed to cover several weaknesses in IBM's BLEU metric. METEOR calculates the similarity score between generations, references and semantic equivalents such as inflections, synonyms and paraphrases. Instead of relying on higher order n-grams, METEOR uses a direct word-ordering penalty.

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) metric, which is first proposed by Lin (2004), is designed to compare n-grams recall of machine produced translations against a human-produced translations. ROUGE-L is measured according to the longest common subsequence.

Our implementation uses the evaluation package released by Chen et al. (2015)

that includes implementation of BLEU-1, BLEU-2, BLEU-3, BLEU-4, METEOR and ROUGE-L metrics.

Constructing a question answering or reading comprehension dataset is also a challenging problem. Richardson et al. (2013) curated MCTest, which each question is paired with 4 answer choices. Despite the dataset contains challenging human generated questions, its size is too small to support data-demanding question answering models. After that, many datasets are released and most of them generate the questions in a synthetic way. bAbI (Weston et al, 2015) is a fully synthetic reading comprehension dataset which features 20 distinct tasks. In order to solve these tasks different types of reasoning is required. Hermann et al. (2015) constructed a corpus of cloze style questions by predicting entities in abstractive summaries of Daily News / CNN articles. With this approach they collected a million new stories. However, Chen et. al (2016) concluded that the dataset is quite easy and current neural networks almost reached ceiling performance.

Finally in 2016, Rajpurkar et al. (2016) released the Stanford Question Answering Dataset (SQuAD). SQuAD is a reading comprehension dataset that consists of 100,000+ questions on 500+ articles. Questions are posed by crowdworkers on a set of Wikipedia articles. It overcomes the semi-synthetic and small size data issues that mentioned above. Example data from SQuAD can be seen in Figure 2.1. For each paragraph, there are respective human generated questions. Also for each human generated question, answers are given with their answer start index with respect in the paragraph that the question is generated.

2.2. Parsers

In this section parsers are explained one by one, with their own particular viewpoint examples.

2.2.1. Part-Of-Speech Tagging

The part-of-speech tagging (POS tagging) is a task that intends to identify syntactic role of each word in the given sentence such as singular noun, adjective. The part-of-speech tagging uses different tagsets based on language and corpus that was collected from different sources. An example part-of-speech of the sample sentence ”The Bill of Rights gave the law federal government greater legitimacy” can be seen in Table 2.1.

```
{
   "paragraphs":[
      {
         "context": "Super Bowl 50 was an American football game to
         determine the champion of the National Football League (NFL)
         for the 2015 season. The American Football Conference (AFC)
         champion Denver Broncos defeated the National Football
         Conference (NFC) champion Carolina Panthers 24\u201310 to
         earn their third Super Bowl title. The game was played on
         February 7, 2016, at Levi's Stadium in the San Francisco
         Bay Area at Santa Clara, California. As this was the 50th
         Super Bowl, the league emphasized the \"golden anniversary\"
         with various gold-themed initiatives, as well as temporarily
         suspending the tradition of naming each Super Bowl game with
         Roman numerals (under which the game would have been known
         as \"Super Bowl L\"), so that the logo could prominently
         feature the Arabic numerals 50.",
         "qas": [
            {
               "answers": [
                  {
                     "answer_start": 177,
                     "text": "Denver Broncos"
                  }
               ],
               "id": "56be4db0acb8001400a502ec",
               "question": "Which NFL team represented the AFC at
               Super Bowl 50?"
            },
            {
               "answers": [
                  {
                     "answer_start": 403,
                     "text": "\"golden anniversary\""
                  },
                  {
                     "answer_start": 521,
                     "text": "gold-themed"
                  },
                  {
                     "answer_start": 521,
                     "text": "gold"
                  }
               ],
               "id": "56bea9923aeaaa14008c91b9",
               "question": "What was the theme of Super Bowl 50?"
            }
         ]
      }
   ]
}
```


Figure 2.1. Example data from SQuAD.

2.2.2. Chunking

Chunking (also called shallow parsing) is a process that first detects constituents in a sentence, then segments them to chunks of syntactically related word groups. Each word is labeled with its own unique tags in the groups. Chunk tags have two parts. The first part indicates the beginning, continuation or end of a chunk. The second part indicates the type of the current word group. For example, beginning of a chunk noun phrase is labeled with B-NP, continuation of a chunk verb phrase is labeled with I-VP. An example chunking tags of the sample sentence ”The Bill of Rights gave the law federal government greater legitimacy” can be seen in Table 2.1.

Table 2.1. An example of POS and chunk tags of sample sentence.

Token	POS Tag	Chunk Tag
The	DT	B-NP
Bill	NNP	E-NP
of	IN	S-PP
Rights	NNPS	S-NP
gave	VBD	S-VP
the	DT	B-NP
new	JJ	I-NP
federal	JJ	I-NP
government	NN	E-NP
greater	JJR	B-NP
legitimacy	NN	E-NP

2.2.3. Dependency Parsing

The main idea of dependency parsing is that each word is connected to each other by directed links. These links are called dependencies in linguistics. The main goal is to reveal the syntactic structure of the sentence by looking at these dependencies. Structure is determined by the relation between a head word and its dependents (childs). The most widely used syntactic structure is a parse tree. It allows to navigate through generated parse tree using head and child tokens. An example dependency tree of the sentence ”REM sleep is characterized by darting movement of closed eyes” can be seen in Figure 2.2.

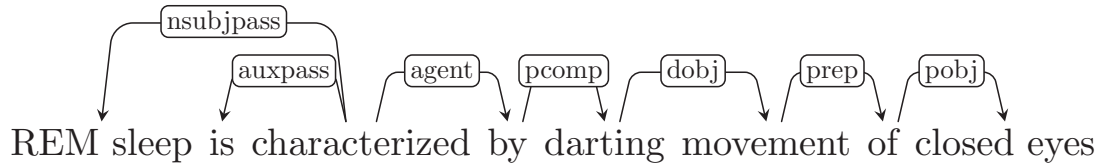

Figure 2.2. An example dependency tree of sample sentence.

2.2.4. Named Entity Recognition

Named Entity Recognition (NER) is an information extraction task that labels words into various semantic categories such as person, date, location, facility. There are various NER approaches based on rule based techniques and statistical models, i.e. machine learning. Rule based approaches rely on hand-crafted rules which is done by experienced linguists, whereas statistical approaches need large set of training data. Statistical models also allow to train custom models and define new categories according to problem domain. Found named entity tags of the sentence ”Designer Ian Callum, originally from Dumfries in Scotland, studied at the Glasgow School of Art and at the Royal College of Art in London” can be seen in Figure 2.3.

Designer Ian Callum (PERSON), originally from Dumfries (GPE) in Scotland (GPE), studied at
the Glasgow School of Art (ORGANIZATION) and at the Royal College of Art (ORGANIZATION) in London (GPE).

Figure 2.3. An Example NER tags in a sample sentence.

2.2.5. Semantic Role Labeling

Semantic Role Labeling (SRL) is a process that aims to reveal the semantic structure of a sentence by labeling word groups and phrases. It is an essential part of NLU and extracts the semantic word groups in a sentence. In SRL representation, predicate is the root and word groups accompanying the predicate is considered as arguments. Depending on their role in the sentence, predicates are assigned to different semantic categories. These categories, which adds predicate-argument layer to syntactic structures of the Penn

Treebank, are decided by the Proposition Bank (Palmer, Kingsbury, Gildea, 2005). The Proposition Bank provides numbered argument tags (such as ARG0, ARG1) and assigns functional tags to all verb modifiers, such as cause (CAU), temporal (TMP), purpose (PNC) and others (Malaya, 2005). An example SRL representation of the sentence ”He is driving the car aggressively” can be seen in Figure 2.4.

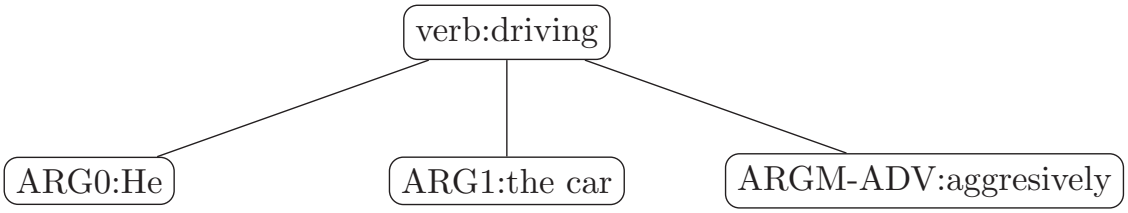

Figure 2.4. An example SRL representation of sample sentence ”He is driving the car aggresively”.

CHAPTER 3

PROPOSED APPROACH

3.1. Predefined Rules (Templates)

In this section, templates and their generation are explained in detail. In order to consider a sentence pattern as a template, it should satisfy the criteria which are previously stated by Mazidi and Tarau (2016). These are: (1) The sentence pattern should be working on different domains. (2) It should extract important points in the source sentence and create an unambiguous question. (3) Semantic information that is transferred by sentence pattern should be consistent across different instances.

Templates, both existing and newly proposed, are categorized below with respect to the parsing method(s) they used. Each sentence pattern is detected by examining synergies between different parsers.

3.1.1. Dependency based templates

Using semantic role labeling, first the semantic structure of a sentence is revealed, then using dependency parsing, semantic arguments are matched with the found dependency tags to check if the sentence template corresponds to any template. In Table 3.1, dependency based templates can be seen with example sentences. S-V-acomp (adjectival complement), S-V-attr (attribute), S-V-dobj (direct object), S-V-pcomp (complement of a preposition) and S-V-dative (three entities) are previously mentioned by Mazidi and Tarau (2016). In the proposed approach, S-V-xcomp (open clausal complement) and S-V-ccomp (clausal complement) are changed and S-V-oprd (object predicate) is added according to the criteria that we have mentioned in Section 3.1.

Complement is a term used for phrase(s) or word(s), which is needed to complete the meaning of the verb including direct objects in a traditional English grammar (Huddleston et al., 2002). The universal dependency has various labels, which are stated by McDonald et al. (2013): acomp (adjectival complement), oprd (object predicate), attr (attribute), xcomp (open clausal complement), dobj (direct object), ccomp (clausal complement) dative (indirect object), pcomp (complement of a preposition). The predicate

determines its participants and arguments (Kroeger, 2005). For instance, S-V-dative pattern has a three entities: subject, direct object and indirect object. Linguists often used the term xcomp to denote predicate complements of various syntactic categories (Kroeger, 2005). In contrast, the universal dependency relations divide the complements into acomp for adjective phrase, attr for noun phrase, ccomp for complement clauses, leaving xcomp for verb phrase. Understanding what syntactic category a complement belongs provides important semantic clues to figure out what the clause is really saying.

Table 3.1. Dependency based template examples.

Template and Example
1. S-V-acomp is an adjective phrase that describes the subject. **S:** It has been argued that the term ”civil disobedience” has always suffered from ambiguity and in modern times, become utterly debased. **Q:** Indicate characteristics of the term ”civil disobedience”.
2. S-V-oprd is an object predicate that defines the subject. **S:** Brain waves during REM sleep appear similar to brain waves during wakefulness. **Q:** How would you describe brain waves during REM sleep?
3. S-V-attr is a noun phrase, usually following copula and defines the subject. **S:** The fourth Yuan emperor, Buyantu Khan (Ayurbarwada), was a competent emperor. **Q:** How would you describe the fourth Yuan emperor, Buyantu Khan (Ayurbarwada)?
4. S-V-xcomp is an open clausal complement without an internal subject. **S:** Some of Britain’s most dramatic scenery is to be found in the Scottish Highlands. **Q:** What is some of Britain ’s most dramatic scenery?
5. S-V-dobj (direct object) is a noun phrase that is the accusative object of a verb. **S:** In 1996, the trust employed over 7,000 staff and managed another six sites in Leeds and the surrounding area. **Q:** What did the trust manage in Leeds and the surrounding area in 1996?
6. S-V-ccomp (clausal complement) is a clause with an internal subject. **S:** He says that you like to swim. **Q:** What does he say?
7. S-V-dative indicates relationship between three entities. **S:** The Bill of Rights gave the new federal government greater legitimacy. **Q:** What did give the new federal government greater legitimacy?
8. S-V-pcomp is a complement of a preposition that modifies the meaning of a prepositional phrase. **S:** REM sleep is characterized by darting movement of closed eyes **Q:** What is REM sleep characterized by?

Mazidi and Tarau (2016) also provided distribution of these patterns from collections of expository text. This can be seen in Table 3.2. While S-V-dobj pattern is the most frequent pattern in a expository text, S-V-dative pattern, which indicates the relationship between three entities, is the most rare pattern.

Table 3.2. Sentence pattern Distribution in expository text stated by Mazidi and Tarau (2016)

Pattern	Meaning	Frequency
S-V-acomp	adjective phrase that describes the subject	8%
S-V-attr	noun phrase, usually following copula and defines the subject	14%
S-V-ccomp	clause with an internal subject	7%
S-V-dobj	noun phrase that is the accusative object of a verb.	28%
S-V-dative	indicates an indirect object.	1%
S-V-pcomp	modifies the meaning of a prepositional phrase	17%
S-V-xcomp	open clausal complement without an internal subject	8%
S-V	indicates an action of the entity	14%
other	combinations of constituents	4%

3.1.2. NER based templates

Using semantic role labeling, first the semantic structure of a sentence is revealed, then using named entity recognition, the designed system detect words that are labeled as person, location, date or number. So, the designed system can ask who, where, when or how many questions accordingly. Then, the tagged word is removed from the corresponding semantic argument. Finally, using the rest of the semantic arguments, the designed system checks if the constructed sentence has reasonable components (subject, direct object, verb,etc.).

However, ”who” question is problematic. Assume that we want to generate questions for the following sentence: ”Atop the Main Building’s gold dome is a golden statue of the Virgin Mary.” In this scenario, NER detects ”Virgin Mary” as a person. However, if we look at the full noun phrase: ”statue of the Virgin Mary” is an object, not a person. So, ”who” question is not an appropriate choice in this situation. The designed system tackles this problem through the use of chunking. So, the full noun phrase can be detected and this problematic case can be eliminated. With the use of chunking, the designed system also detects relative clauses to ensure a sentence really mentions a person, not an object. If an object detected instead of a person, question can’t be generated.

NER based templates and examples can be seen in Table 3.3. For instance, in the sample 1, number is detected by NER and the detected phrase is removed from the corresponding semantic representation. After, the remaining parts of the semantic representation are checked. If it has valid sentence parts, it can generate the corresponding template. In addition, for location templates, the designed system includes the following NER tags: facility (buildings, airports, highways, bridges, etc.) organization (companies, agencies, institutions, etc.), gpe (countries, cities, states), location (mountain ranges, bodies of water) from SpaCy's implementation.

Table 3.3. NER based template examples.

Template and Example
1. S-V-number. **S:** In 1996, the trust employed over 7,000 staff and managed another six sites in Leeds and the surrounding area. **Q:** How many staff did the trust employ in 1996?
2. S-V-location. **S:** In 1996, the trust employed over 7,000 staff and managed another six sites in Leeds and the surrounding area. **Q:** Where did the trust manage another six sites in 1996?
3. S-V-date. **S:** In 1996, the trust employed over 7,000 staff and managed another six sites in Leeds and the surrounding area. **Q:** When did the trust employ over 7,000 staff?
4. S-V-person. **S:** The fourth Yuan emperor, Buyantu Khan (Ayurbarwada), was a competent emperor. **Q:** Who was a competent emperor?

3.1.3. SRL based templates

Using only semantic role labeling, the designed system reveals the semantic structure of the sentence under consideration. SRL based templates and examples can be seen in Table 3.4. Using annotation of modifiers (Malaya, 2005) following templates are generated with according to the criteria that mentioned in Section 3.1:

- ARGM-CAU is a cause clause and indicates the reason of action. We conclude that asking ”why” question is suitable.
- ARGM-MNR is a manner marker and specifies how the action is performed. We conclude that asking ”how” question is suitable.
- ARGM-CAU is a purpose clause and shows the motivation of action. We conclude that asking ”for what purpose” question is suitable.
- ARGM-LOC indicates where some action takes place
- ARGM-TMP shows when the action took place.

Table 3.4. SRL based template examples.

Template and Example
1. S-V-ARGM-CAU: cause clause. **S:** On 24 March 1879, Tesla was returned to Gospic under police guard for not having a residence permit. **Q:** Why was Tesla returned to Gospic on 24 March 1879?
2. S-V-ARGM-MNR: manner marker. **S:** Tesla’s work for Edison began with simple electrical engineering and quickly progressed to solving more difficult problems. **Q:** How did Tesla’s work for Edison progress quickly to solving more difficult problems?
3. S-V-ARGM-PNC: purpose clause. **S:** To secure further loans, Westinghouse was forced to revisit Tesla’s AC patent, which bankers considered a financial strain on the company (at that point Westinghouse had paid out an estimated $200,000 in licenses and royalties to Tesla, Brown, and Peck).. **Q:** For what purpose was Westinghouse forced to revisit Tesla’s AC patent, which bankers considered a financial strain on the company?
4. S-V: don’t have any modifier, using only numbered argument labels to generate yes/no questions. **S:** The Bill of Rights gave the new federal government greater legitimacy. **Q:** Did the Bill of Rights give greater legitimacy?
5. S-V-ARGM-LOC: locative. **S:** Mr. Bush met him privately, in the White House, on Thursday. **Q:** Where did mr. Bush meet him on Thursday?
6. S-V-ARGM-TMP: temporal marker. **S:** Mr. Bush met him privately, in the White House, on Thursday. **Q:** When did mr. Bush meet him in the White House?

3.2. The Designed System

In this section, the designed system is explained in general, as well as its each component. Mazidi and Tarau (2016) state that natural language understanding (NLU) is a missing piece of the puzzle in question generation. So, in a similar vein, the main focus of this study is NLU. It is achieved by exploiting the semantic roles of words.

The designed system focuses both on deep and shallow questions which have mentioned in Section 2.1. As shallow questions, the designed system can generate the following question sentences:

- What...?
- Who...?
- Where...?
- How many...?
- When...?

As deep questions, the designed system can generate:

- Why...?
- How...?
- How would you describe...?
- Indicate characteristics of...?
- For what purpose...?

As seen in Figure 3.1, the designed system takes an input sentence, which is first preprocessed. In the preprocessing stage, contractions are expanded first. For instance, ”would’ve” is expanded into ”would have”. Contractions can be problematic for parsers, as most of them get parsed wrong and generate unexpected results. So, in order to detect verb groups perfectly and reduce errors, the designed system uses our predefined dictionary to handle contractions.

In the second step of preprocessing, idiomatic language is eliminated. Idiomatic language is another problematic case for question generation systems. It is mentioned in Section 2.1. The designed system detects idioms by using our predefined dictionary and SpaCy’s dependency parsing algorithm [1] and eliminates these problematic cases. For

[1] https://github.com/explosion/spaCy/

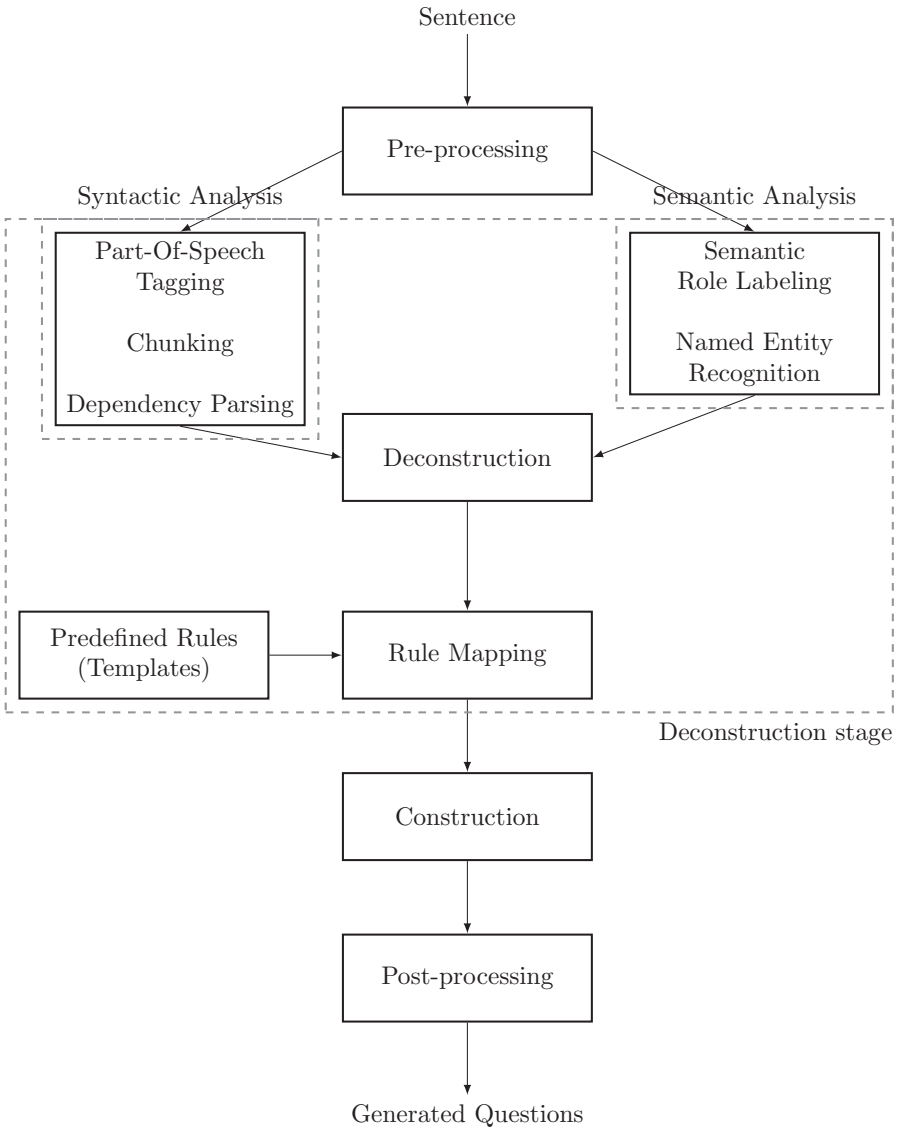

Figure 3.1. Flowchart of the designed system

instance, the sentence ”These days, many people spend time surfing social networking sites.” results in the generated question: ”What do many people spend these days?”. However, the generated question is vague and out of context. In this case, ”time” grammatically is the direct object, which is why this question was generated, but ”spend time” is an idiom. An example dependency tree of this sentence can be seen in Figure 3.2. As you can see there is a relation between ”spend” and ”time”. Dobj (direct object) relation can be used to generate dependency based questions as we mentioned in Section 3.1.1. However, with our predefined dictionary, ”spend time” is detected and eliminated from question generation process. Example data from our predefined dictionary can be seen in Figure 3.3. ”Spend time” is also defined in our predefined dictionary, so it can be easily

eliminated.

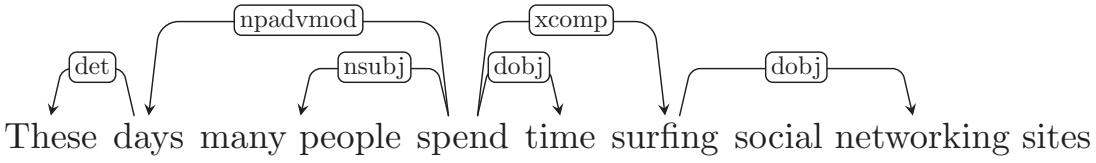

Figure 3.2. An example dependency tree of sample sentence.

```
{
      "spend":"time",
      "bite":"the bullet",
      "speak":"of the devil",
      "under":"the weather",
      "hit":"the sack",
      "break":"a leg"
}
```

Figure 3.3. Example data from our predefined idiom dictionary.

The designed system uses multiple semantic and syntactic parsers because each parser tells its own particular viewpoint about the sentence and add to its understanding. Following the preprocessing, in the deconstruction stage, dependencies between the words, the named entity information and also semantic word groups are extracted. For extracting dependencies between the words and gathering the named entity information, the designed system relies on Spacy's algorithms. Finally, the designed system uses AllenNLP's (Gardner et al., 2017) semantic role labeler for extracting semantic word groups in a sentence. We also tried SENNA's SRL algorithm and Stanford's CoreNLP project which includes NER and dependency algorithm. Because of literature recommendations and personal experience, we have decided to use other parsers that we have mentioned above.

The main objective of deconstruction stage is to get an intermediate representation in order to determine the sentence pattern. Sentence pattern is crucial to detect the type of question to be generated. Systematic arrangement of words (such as Subject + Verb + Object + Complement) in a sentence is called the sentence pattern.

Intermediate representation consists of sentence elements. In English, a part of the sentence is classified as a certain sentence element such as subject, direct object, verb, subject complement, etc. Full verb detection is also needed in this stage. The designed system relies on chunking algorithm to extract verb groups and phrases. For chunking algorithm, the designed system uses SENNA's (Collobert et al., 2011) implementation.

Algorithm 1 shows the pseudo code of the deconstruction stage. The deconstruction stage exploits synergies between SRL and dependency parser, SRL and NER parser. Also, it exclusively examine SRL parser to look for more sentence patterns. First, dependency list, NER list and SRL list are defined with constant string names. These are the templates mentioned in Section 3.1. First loop (in lines 9-12) focuses on searching dependency based templates mentioned in Section 3.1.1. Dependency Found tags are checked one by one. If any tag matches with our predefined list of dependencies, the designed system checks the validity of the sentence by examining the corresponding semantic representation. This process is performed by examining numbered argument tags (such as ARG0, ARG1) in the semantic representation (please see Section 2.2.5 for details). The least numbered argument becomes the subject of the sentece and the other one becomes the object. If the semantic representation has no problem (has a subject, object, verb) and head node of the dependency found is matches with verb of a semantic representation, then deconstruction stage begins. Using semantic representation and chunk tags, sentence is separated into its parts, then added to the deconstructed list. If extra fields are exist in the semantic representation (such as ARGM-TMP and ARGM-LOC), they are also added. Next loop starts in line 13, which focuses on searching NER based templates mentioned in Section 3.1.2. It has a similar process with the previous loop. Instead, this time, found NER phrase is removed from the corresponding semantic representation and remaining parts of it becomes the object of the question. In SRL based templates (starting in line 19), there is an extra option for generating yes/no question. If it is active, validity of the corresponding SRL object becomes enough to generate questions. The time complexity of Algorithm 1 is $O(n^3)$, because we have three nested for loop in line 13.

In construction stage, the designed system matches the sentence pattern with predefined rules, i.e. templates. If a rule matches with the sentence pattern, a question can be generated. By examining extracted verb groups with part-of-speech tagging, grammatical tense is detected. The designed system relies on SENNA's (Collobert et al., 2011) part-of-speech tagging algorithm, which uses English Penn Treebank tagset (Marcus, Marcinkiewicz and Santorini, 1993). Then, using Python package named as Pattern, which is developed by Smedt and Daelemans (2012), the designed system gets the base

form of the verb. Finally, sentence elements are put together to form a question with respect to the detected template type. The time complexity of Algorithm 2 is $O(n)$.

Algorithm 1 Deconstruction

```
1: function DECONSTRUCT(sentence)
2:     depList ← ["dobj", "acomp", "attr", "pcomp", "ccomp", "oprd", "dative"]
3:     nerList ← ["location", "date", "person", "number"]
4:     srlList ← ["argm-cau", "argm-pnc", "argm-mnr", "argm-loc", "argm-tmp"]
5:     srlTags ← set of found srl lists in the sentence.
6:     depTags ← dependency representation of the sentence.
7:     nerTags ← detected ner tags in the sentence.
8:     chunkTags ← shallow chunking representation of the sentence.
9:     for each d ∈ depTags do
10:        for each s ∈ srlTags do
11:            if depList.match(d.dep_) and s['V'] == d.head.text and isValid(s) then
12:                HANDLEDECONSTRUCTION(s, chunkTags, null, d.dep_)
13:    for each d ∈ nerTags do
14:        for each s ∈ srlTags do
15:            for each value ∈ s do
16:                if nerList.match(n.label_) and value.match(n.text) and isValid(s) then
17:                    value ← remove(n.text,value)        ▷ remove the found ner from s
18:                    HANDLEDECONSTRUCTION(s, chunkTags, value, n.label_)
19:    for each s ∈ srlTags do
20:        for each key ∈ s do
21:            if srlList.match(key) and and isValid(s) then
22:                HANDLEDECONSTRUCTION(s,chunkTags)
23:            else if isValid(s) then                  ▷ option for creating yes/no question
24:                HANDLEDECONSTRUCTION(s, chunkTags, null, key)
25:
26:
27: function HANDLEDECONSTRUCTION(srlObject,chunkTags,modifiedValue,
    questionType)
28:    fullVerb ← getFullVerb(srlObject['V'],chunkTags)        ▷ get sentence parts
29:    if modifiedValue != null then
30:        object ← modifiedValue
31:    else
32:        object ← getObject(srlObject)
33:    subject ← getSubject(srlObject)
34:    extaField ← getExtraField(srlObject)
35:    ADDDECONSTRUCTEDPARTS(fullVerb,subject,object,extaField,
        questionType)
```

In post processing stage, the designed system eliminates the same generated questions. In addition, for each generated question, the designed system checks and arranges punctuations and capital letters. After checking them one by one, the system returns the generated questions as an output.

Our implementation is written in Python 3.6. The designed system relies on:

- SENNA's part-of-speech tagging and chunking algorithm,

Algorithm 2 Construction

```
1: function CONSTRUCT
2:     for i in 0...types.length do
3:         verbParts ← convertVerbTense(fullVerb[i])
4:         question ← buildQuestion(types[i], verbParts, objects[i], subjects[i],
               extraFields[i])
5:         formattedQuestion ← postProcess(question)
6:         foundQuestions.push(formattedQuestion)
       return formattedQuestion
7:
```

- AllenNLP's semantic role labeling algorithm,
- SpaCy's dependency parsing and NER algorithm, and
- Python package named as Pattern to get the base form of verbs.

Moreover, two dictionaries are constructed to handle contractions and detect idioms. The source code is available at github[2].

Assume that we want to see question generation process for the sample sentence: "In 1996, the trust employed over 7,000 staff and managed another six sites in Leeds and the surrounding area." The sentence does not have any contractions to expand, neither idioms to detect. So, we skip the preprocessing stage. Before we begin the deconstruction stage, sentence is parsed with using various parsers.

Chunking representation of the sentence: [('In', 'S-PP'), ('1996', 'S-NP'), (',', 'O'), ('the', 'B-NP'), ('trust', 'E-NP'), ('employed', 'S-VP'), ('over', 'S-PP'), ('7,000', 'B-NP'), ('staff', 'E-NP'), ('and', 'O'), ('managed', 'S-VP'), ('another', 'B-NP'), ('six', 'I-NP'), ('sites', 'E-NP'), ('in', 'S-PP'), ('Leeds', 'S-NP'), ('and', 'O'), ('the', 'B-NP'), ('surrounding', 'I-NP'), ('area.', 'E-NP')]

Detected NER tags: {1996:DATE, 7,000:CARDINAL, six:CARDINAL, Leeds:ORG}

Dependency representation of the sentence:

pobj(In-1, 1996-2)

nsubj(employed-4, the trust-3)

prep(employed-4, In-1)

dobj(employed-4, over 7 000 staff-5)

cc(employed-4, and-6)

conj(employed-4, managed-7)

dobj(managed-7, another six sites-8)

[2]https://github.com/OnurKeklik/Qg-Iztech

prep(another six sites-8, in-9)
pobj(in-9, Leeds-10)
cc(Leeds-10, and-11)
conj(Leeds-10, the surrounding area-12)

Set of found semantic representations in the sentence: [{'ARGM-TMP': 'In 1996', 'ARG0': 'the trust', 'V': 'employed', 'ARG1': 'over 7,000 staff'}, {'ARGM-TMP': 'In 1996', 'ARG0': 'the trust', 'V': 'managed', 'ARG1': 'another six sites', 'ARGM-LOC': 'in Leeds and the surrounding area'}, {'V': 'surrounding', 'ARG1': 'area'}]

Then, we deconstruct the sentence. We will look for dependency based templates in Algorithm 1 in lines 9-12. First, we look at the dependency representation of the sentence. There is a dobj (direct object) dependency between "employed" and "over 7,000 staff". Then, we look at the first set of found semantic representation. The least numbered argument in SRL becomes subject of the new sentence and the other one becomes object. So, "the trust" becomes subject and "over 7,000 staff" becomes object of the new sentence. If we look at the chunking representation of the sentence, there is no auxiliary verb. So we just take "employed" as a verb of the new sentence. "over 7,000 staff" also belongs to object of the new sentence. We also said that there is a dobj (direct object) dependency between "employed" and "over 7,000 staff". Since we are going to ask the object of the new sentence for S-V-dobj template that we have mentioned in Section 3.1.1, deconstructed parts are become valid. We also add "In 1996" at the end of the sentence, because it belongs to ARGM-TMP semantic tag which indicates time and completes the meaning of the sentence. So, deconstructed parts {"type": "dobj", "subject": "the trust", "verb": "employed", "object": "over 7,000 staff", "extraField":"In 1996"} are added to the queue for construction stage.

If we look at the second set of found semantic representation of the sentence, another dobj (direct object) pattern is found between "managed" and "another six sites". Semantic representation is also valid. So, deconstructed parts {"type": "dobj", "subject": "the trust", "verb": "managed", "object": "another six sites", "extraField":"in Leeds and the surrounding area In 1996"} are added to the construction queue.

Next, we will look for NER based templates in lines 13-18. As you may have noticed, "six" is detected as CARDINAL in the detected NER tags. If we look at the Algorithm 1 in line 17, we remove the found NER tags from the semantic representation. Then, we check the remaining semantic representations if they are valid or not. If we look at the second set of found semantic representation of the sentence, "another six sites"

is labeled as ARG1. In contrast to other templates, this argument becomes our subject, because we are going to ask information about it. We remove ”another six” from the object. ”Another” word is also taken along with ”six”, because chunking representation of the sentence clearly shows that they form a syntactically related word group. So, similar to previous deconstruction part, subject, verb and other fields are selected. So, deconstructed parts {”type”: ”number”, ”subject”: ”sites”, ”verb”: ”managed”, ”object”: ”the trust”, ”extraField”:”in Leeds and the surrounding area In 1996”} are added to the construction queue. Other found NER based templates are also deconstructed and added to the construction queue.

Next, we will look for SRL templates in lines 19-24. These templates have similar deconstruction stages with NER based templates and dependency based templates. After deconstruction, found deconstructed parts are added to the construction queue.

For each deconstructed parts, question word is created with respect to detected template type. For dobj (direct object) template, ”what” question is taken and for number template, ”how many” question is taken. Also, grammatical tense is detected. If base form of the verb is needed, it is converted into its base form. After arranging punctuations and capital letters (with respect to proper nouns), question becomes complete. So, for the examples that we have mentioned above, three sentences are generated.

For deconstructed parts {”type”: ”dobj”, ”subject”: ”the trust”, ”verb”: ”employed”, ”object”: ”over 7,000 staff”, ”extraField”:”In 1996”}, the following question is generated: ”What did the trust employ in 1996?”

For deconstructed parts {”type”: ”dobj”, ”subject”: ”the trust”, ”verb”: ”managed”, ”object”: ”another six sites”, ”extraField”:”in Leeds and the surrounding area In 1996”}, the following question is generated: ”What did the trust manage in Leeds and the surrounding area in 1996?”

For deconstructed parts {”type”: ”number”, ”subject”: ”sites”, ”verb”: ”managed”, ”object”: ”the trust”, ”extraField”:”in Leeds and the surrounding area In 1996”}, the following question is generated: ”How many sites did the trust manage in Leeds and the surrounding area in 1996?”

CHAPTER 4

EVALUATION

4.1. Automatic Evaluation

In the automatic evaluation stage, the designed system's performance is measured with using automatic evaluation metrics. Despite the designed system focuses on question generation from sentences, its performance is compared with the Du's reading comprehension system (Du et al. 2017) that focuses on question generation from paragraphs. If the designed system does not use any predefined rules, it can only generate yes/no questions (with using semantic role labeling). This is the baseline performance for the designed system. In order to compare two systems, the designed system sets up with the same evaluation environment as the other system. Both systems use Stanford Question Answering Dataset (SQuAD), explained in Section 2.1. Both systems run their own system on dev dataset of SQuAD. Dev dataset has 2067 paragraphs. These paragraphs include 10458 sentences with 253778 words and 10570 questions with 107990 words. Also, both systems use the same evaluation package released by Chen et al. (2015) that includes implementation of BLEU-1, BLEU-2, BLEU-3, BLEU-4, METEOR and ROUGE-L metrics. Table 4.1 shows BLEU 1-4, METEOR and ROUGE-L scores of various systems. Our automatic evaluation through objective neural translation metrics show that our system has superior performance and outperforms others in BLEU-2, METEOR, and ROUGE-L metrics. Especially for METEOR metric, the designed system gets highly significant difference. Banerjee et al. (2005) demonstrated that METEOR has significantly enhanced correlation with human evaluators. They also demonstrated that when obtaining high level correlation with human evaluators, recall plays more significant role than precision. The designed system uses diversity of templates, that are consist of variety of parsers. We believe that, this diversity increases recall and METEOR score.

Du et al. (2017) used a configured setup to evaluate their own system, with systems that are IR_{BM25}, $IR_{Edit\ Distance}$, MOSES+, DirectIn, H&S and Vanilla seq2seq. Their results can be seen in Table 4.1. In order to compare their system with H&S rule based system, which is previously mentioned in Section 2.1, they take the top question in the ranked list. In order to preserve the experimental setup, the designed system also takes

Table 4.1. BLEU 1-4, and ROUGE-L scores of different systems.

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE-L
IR_{BM25}	5.18	0.91	0.28	0.12	4.57	9.16
$IR_{Edit\ Distance}$	18.28	5.48	2.26	1.06	7.73	20.77
MOSES+	15.61	3.64	1.00	0.30	10.47	17.82
DirectIn	31.71	21.18	15.11	11.20	14.95	22.47
H&S	38.50	22.80	15.52	11.18	15.95	30.98
Vanilla seq2seq	31.34	13.79	7.36	4.26	9.88	29.75
Du's Model (no pre-trained)	41.00	23.78	15.71	10.80	15.17	37.95
Du's Model (w/ pre-trained)	43.09	25.96	17.50	12.28	16.62	39.75
The Designed System	41.90	26.90	16.90	10.61	25.01	40.38

the top ranked question in its list by pre-evaluating generated questions to run the experiment. In pre-evaluation, for each generated question for a sentence, BLEU-1 scores are calculated with respect to the ground truth questions in SQuAD. Then, top rated question is selected for the sentence. These top questions are used to evaluate the designed system. The designed system does not require any ranking algorithm like H&S system, because it does not overgenerate the questions. All generated questions with respect to detected templates should be accurate, unambiguous and relevant.

4.2. Human Evaluation

Human evaluation studies were also performed to measure the quality of generated questions. Human evaluators evaluate the submitted questions according to evaluation criteria, which are difficulty, relevance, syntactic correctness and ambiguity. Relevance, syntactic correctness and ambiguity were previously stated on QGSTEC (2010). Similarly, difficulty previously mentioned by Du et al. (2017). We simply combine previously defined criteria to perform human evaluation studies.

The evaluation criteria has two purposes. First, they shows guideline for human evaluators. Second, they help us to detect performance of each template.

- Difficulty: Difficulty is rated to ensure that there is a syntactic divergence between the input sentence and generated question. That is, some reasoning is necessary to answer the question. The difficult question should assess the reader's knowledge about the input sentence (Du et al., 2017).

- Relevance: Relevance is rated to ensure that the question can be answered based on the input sentence QGSTEC (2010).
- Correctness: Syntactic correctness is rated to ensure that the question generated is grammatically correct QGSTEC (2010).
- Ambiguity: Ambiguity is rated to ensure that the question makes sense when asked with no context. Typically, an unambiguous question will have one very clear answer QGSTEC (2010).

Average scores by each question type can be seen in Table 4.2. Although S-V-acomp question type gets the poorest score for the relevance metric, it achieves the best score for the difficulty and ambiguity metrics. Its correctness score is also high. In addition, S-V-person (who) question type achieves the best score for the correctness and relevance metrics. We can conclude that, we successfully eliminated the problematic case that we have mentioned in Section 3.1.2., because the grammatical correctness is very high.

Table 4.2. Average scores of the designed system by question type

Question Type	Difficulty	Ambiguity	Correctness	Relevance
S-V-dobj (what)	3.31	3.34	4.23	4.1
S-V-acomp	4.38	3.63	4.5	2
S-V-attr	4.31	3.19	4.31	4
S-V-pcomp (what)	3.5	2	2.75	3
S-V-date, S-V-ARGM-TMP (when)	2.79	3.36	4.01	3.89
S-V-number (how many)	2.75	2.44	2.81	3.56
S-V-person (who)	3.13	3.62	4.262	4.5
S-V-location, S-V-ARGM-LOC (where)	3	2.77	3.10	3.03
S-V (yes/no)	2.31	3.22	3.72	3.97
S-V-ARGM-MNR (how)	3.57	2.64	2.71	2.86

For the generated question set; the total number of questions, their types, and the total number of answers are given in Table 4.3. In order to evaluate the suitability of the evaluation criteria in measuring the performance of questions of different types, we applied analysis of variance (ANOVA). ANOVA is used to test the null hypothesis that

Table 4.3. The total number of questions, their types, and the total number of answers.

Question Type	Total Questions	Total Answers
S-V-dobj (what)	16	64
S-V-acomp	2	8
S-V-attr	4	16
S-V-pcomp (what)	1	4
S-V-date, S-V-ARGM-TMP (when)	14	56
S-V-number (how many)	4	16
S-V-person (who)	2	8
S-V-location, S-V-ARGM-LOC (where)	26	104
S-V (yes/no)	45	180
S-V-ARGM-MNR (how)	7	28
TOTAL	121	484

question types are independent from the evaluation criteria, which are difficulty, ambiguity, correctness, and relevance. As a result, all but one of these criteria got so small p-values that we can safely reject the null hypothesis meaning that there is dependency between these criteria and the generated question types. In other words, these three criteria are proved useful in distinguishing the performance among different question types. Only the criterion of ambiguity does not give a statistically significant p-value, which can be interpreted such that ambiguity is not a useful measure to test the performance of various factual questions. The calculated p-values along with question types are listed in Table 4.4.

Table 4.4. ANOVA p-values: Question types against the evaluation criteria.

Difficulty	Ambiguity	Correctness	Relevance
9.1e-10	0.0485	1.14e-09	4.22e-10

We also examine the previous human studies. In Table 4.5, we can see the score of four competitors that participated in QGSTEC (with penalty for missing questions). Input sentences are selected from Wikipedia, OpenLearn and Yahoo! Answers (30 input for each sentence). Originally, the human evaluation performed on a 1-4 scale (1 for the best). In order to compare it with the designed system and Du's reading comprehension

Table 4.5. Human evaluation results for participated systems in QGSTEC.

	Relevance	Correctness	Ambiguity
MRSQG Saarland	3.94	3.39	4.07
WLV Wolverhampton	2.63	2.29	3.23
JUGG Jadavpur	2.69	2.16	3.15
Lethbridge	2.95	3.36	2.52

system, results are converted into a 1-5 scale (5 for the best). This is done by using this formula: y = 6 - 5 * x / 4, where x is on a 1-4 scale (1 for the best), and y is a converted score which is on a 1-5 scale (5 for the best).

Du et al. (2017) also performed human evaluation studies. They evaluated the performance of the H&S system and their own system. In order to compare both systems effectively, they also rated ground truth (human generated) questions in SQuAD. They used two criteria: difficulty and naturalness. Naturalness indicates the grammatical correctness and fluency. It corresponds to correctness criteria that mentioned above. They randomly sampled 100 sentence-question pairs in SQuAD and asked four professional English speakers to rate the sentence-question pairs in terms of difficulty and naturalness on a 1-5 scale (5 for the best). In our experimental setup, 25 sentences are randomly sampled from SQuAD. From this set, 121 questions are generated and rated by four different professional English speakers on a 1-5 scale (5 for the best). We do not rate the ground truth questions in SQuAD. However, our experimentation setup is very similar to Du's setup, we can still compare their results with our own results. Table 4.6 shows human evaluation results for Du's system, H&S system and Table 4.7 shows human evaluation results for the designed system. Du's system outperforms H&S system in both criteria. However, ground truth questions get difficulty score of 2.63 and Du's system gets difficulty score of 3.03. As can be seen from the results, it generated more difficult questions than human. For difficulty criteria, the designed system gets score of 2.85 and real human-generated questions gets score of 2,63. Also, for correctness criteria, the designed system gets score of 3,60 and real human-generated questions gets score of 3,91. As can be seen from the results, the designed system significantly outperforms all other systems and turns out to be the most natural (human-like) system.

Sample questions generated by H&S system, Du's system and the designed sys-

Table 4.6. Human evaluation results for other systems

	Difficulty	Correctness
H&S	1.94	2.95
Du's System	3.03	3.36
Human	2.63	3.91

Table 4.7. Human evaluation results for the designed system

	Difficulty	Correctness
The Designed System	2.85	3.60

tem can be seen in Table 4.8. If we look at the sample sentence 1, Du's system generated more difficult question than human generated (natural) question. Also for the sample sentence 5, Du's system generated unrelated and difficult question. Again, human evaluation results in Table 4.6 confirm this. In the sample sentence 2 and 6, while H&S system completely failed to generate questions, other systems generated accurate questions. In the sample sentence 4, all systems generated accurate and similar questions.

The designed system is also not perfect. In the sample sentence 3, the designed system created an imprecise question that could have more than one possible answers. This issue is related to one of the syntactic challenges that have mentioned in Section 2.1. Complex syntactic structure makes an information extraction difficult. On the other hand, If we look at sample sentence 7, the designed system tries to ask information about the Eldon Square Shopping Centre but fails to detect object. This is one of the weaknesses of the designed system. In reading comprehension texts, it fails to detect paragraph-level information. This is one of the discourse challenges that we have mentioned in Section 2.1.

Table 4.8. Sample questions generated by H&S, Du et al., human (ground truth questions), and the designed system

Sentence 1: Free oxygen first appeared in significant quantities during the Paleoproterozoic eon (between 3.0 and 2.3 billion years ago).

(cont. on next page)

Table 4.8 (cont)

Human: During which eon did free oxygen begin appearing in quantity?
H&S: What first appeared in significant quantities during the paleoproterozoic eon?
Du: How long ago did the paleoproterozoic exhibit?
The Designed System: How did free oxygen appear during the Paleoproterozoic eon?

Sentence 2: Inflammation is one of the first responses of the immune system to infection.
Human: What is one of the first responses the immune system has to infection?
H&S: What is inflammation one of?
Du: What is one of the first objections of the immune system to infection?
The Designed System: How would you describe inflammation?

Sentence 3: Tea, coffee, sisal, pyrethrum, corn, and wheat are grown in the fertile highlands, one of the most successful agricultural production regions in Africa.
Human: (1) Where is the most successful agricultural prodcution regions?
(2) What is grown in the fertile highlands?
H&S: What are grown in the fertile highlands in africa?
Du: What are the most successful agricultural production regions in africa?
The Designed System: Where are Tea, coffee, sisal, pyrethrum, corn, and wheat grown?

Sentence 4: As an example , income inequality did fall in the united states during its high school movement from 1910 to 1940 and thereafter.
Human: During what time period did income inequality decrease in the united states?
H&S: Where did income inequality do fall during its high school movement from 1910 to 1940 and thereafter as an example?
Du: When did income inequality fall in the us?
The Designed System: When did income inequality do fall in the United States?

Sentence 5: However, the rainforest still managed to thrive during these glacial periods, allowing for the survival and evolution of a broad diversity of species.
Human: Did the rainforest managed to thrive during the glacial periods?
H&S: What allowed for the survival and evolution of a broad diversity of species?
Du: Why do the birds still grow during glacial periods?
The Designed System: Did the rainforest manage to thrive during these glacial periods?

(cont. on next page)

Table 4.8 (cont)

Sentence 6: Maududi founded the jamaat-e-Islami party in 1941 and remained its leader until 1972.
Human: When did maududi found the jamaat-e-islami party?
H&S: Who did maududi remain until 1972?
Du: When was the jamaat-e-islami party founded?
The Designed System: When did Maududi found the Jamaat-e-Islami party?
Sentence 7: The largest of these is the Eldon Square Shopping Centre, one of the largest city centre shopping complexes in the UK.
Human: What is one of the largest city center shopping complexes in the UK?
H&S: What is the eldon square shopping centre one of?
Du: What is one of the largest city centers in the UK?
The Designed System: How would you describe the largest of these?

CHAPTER 5

CONCLUSION AND FUTURE WORK

This thesis presented a rule based automatic question generation system that focuses on both question generation from sentences and paragraphs. Especially, with respect to METEOR metric, the designed system significantly outperforms all other systems in automatic evaluation stage. Banerjee et al. (2005) demonstrated that METEOR has significantly enhanced correlation with human evaluators. So, our results confirm that statement by performing human evaluation study. In conclusion, the designed system significantly outperforms all other systems in human evaluation study by generating the most natural (human-like) questions.

For deciding between who and what questions, we proposed solution in Section 3.1.2. This problem is one of the lexical challenges that we have stated in Section 2.1. Our results in Table 4.2 shows that, with 4.262 correctness score, we correctly differentiate between who and what questions. Also, for another lexical challenge, non-compositionality that is stated in Section 2.1, we proposed solution in Section 3.2. Our predefined dictionary does not cover all idioms. Also, some types of idioms can not be covered with predefined dictionary. This issue will be explored in the future work.

Currently, our templates do not achieve the best performance across all question categories. If we look at Table 4.2, S-V-number and S-V-ARGM-MNR (how) type of questions has a low correctness score. In addition, in order to improve the performance of paragraph based questions in all templates, we need to investigate how to better use the paragraph-level information. This is one of the discourse challenges that we have mentioned in Section 2.1. Information conveyed from one sentence to other is an problematic issue. So, we leave this issue to future work. Finally, some templates fit better with some topics than others. For instance S-V-attr and S-V-oprd templates that is stated in Section 3.1.1, works better with noun phrases that are suitable with descriptive questions. For definition questions, other techniques need to be explored in the future work.

Also, adapting the designed system to Turkish language would not be easy due to lack of syntactic and semantic parsers. Without high-performance parsers, adapting

predefined rules into Turkish language would not give a similar performance.

REFERENCES

Babko-Malaya, O. (2005). Propbank annotation guidelines. *URL: http://verbs. colorado. edu*.

Banerjee, S. and A. Lavie (2005). Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pp. 65–72.

Bloom, B. and M. F. Englehart (2011). E., hill, w., & krathwohl, d.(1956). taxonomy of educational objectives: The classification of educational goals. handbook i: Cognitive domain.

Chen, D., J. Bolton, and C. D. Manning (2016). A thorough examination of the cnn/daily mail reading comprehension task. *arXiv preprint arXiv:1606.02858*.

Chen, X., H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollár, and C. L. Zitnick (2015). Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*.

Chi, M. T., S. A. Siler, H. Jeong, T. Yamauchi, and R. G. Hausmann (2001). Learning from human tutoring. *Cognitive Science 25*(4), 471–533.

Collobert, R., J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research 12*(Aug), 2493–2537.

Corbett, A. and J. Mostow (2008). Automating comprehension questions: lessons from a reading tutor. In *Proceedings of the 1st Workshop on Question Generation*.

DiPaolo, R. E., A. C. Graesser, D. J. Hacker, and H. White (2004). Hints in human and computer tutoring. *The design of instruction and evaluation: Affordances of using media and technology 155*.

Doddington, G. (2002). Automatic evaluation of machine translation quality using n-

gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*, pp. 138–145. Morgan Kaufmann Publishers Inc.

Du, X., J. Shao, and C. Cardie (2017). Learning to ask: Neural question generation for reading comprehension. *arXiv preprint arXiv:1705.00106*.

Gardner, M., J. Grus, M. Neumann, O. Tafjord, P. Dasigi, N. Liu, M. Peters, M. Schmitz, and L. Zettlemoyer. Allennlp: A deep semantic natural language processing platform.

Hacker, D. J., J. Dunlosky, and A. C. Graesser (1998). *Metacognition in educational theory and practice*. Routledge.

Heilman, M. (2011). *Automatic factual question generation from text*. Ph. D. thesis, Carnegie Mellon University.

Heilman, M. and N. A. Smith (2010). Rating computer-generated questions with mechanical turk. In *Proceedings of the NAACL HLT 2010 workshop on creating speech and language data with Amazon's mechanical turk*, pp. 35–40. Association for Computational Linguistics.

Hermann, K. M., T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom (2015). Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pp. 1693–1701.

Huddleston, R., G. K. Pullum, et al. (2002). The cambridge grammar of english. *Language. Cambridge: Cambridge University Press*, 1–23.

IBM (2014). *IBM Watson Ecosystem - Getting Started Guide*.

Kroeger, P. R. (2005). *Analyzing grammar: An introduction*. Cambridge University Press.

Kunichika, H., T. Katayama, T. Hirashima, and A. Takeuchi (2004). Automated question generation methods for intelligent english learning systems and its evaluation. In *Proc. of ICCE*.

Labutov, I., S. Basu, and L. Vanderwende (2015). Deep questions without deep understanding. In *Proceedings of the 53rd Annual Meeting of the Association for Compu-*

tational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Volume 1, pp. 889–898.

Lavie, A. and M. J. Denkowski (2009). The meteor metric for automatic evaluation of machine translation. *Machine translation 23*(2-3), 105–115.

Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Lin, C.-Y. (2008). Automatic question generation from queries. In *Workshop on the question generation shared task*, pp. 156–164.

Manning, C., M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky (2014). The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pp. 55–60.

Marciniak, T. (2008). Language generation in the context of yahoo! answers. In *online Proceedings of 1st Question Generation Workshop*.

Marcus, M. P., M. A. Marcinkiewicz, and B. Santorini (1993). Building a large annotated corpus of english: The penn treebank. *Computational linguistics 19*(2), 313–330.

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

McDonald, R., J. Nivre, Y. Quirmbach-Brundage, Y. Goldberg, D. Das, K. Ganchev, K. Hall, S. Petrov, H. Zhang, O. Täckström, et al. (2013). Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Volume 2, pp. 92–97.

Mitkov, R., H. LE AN, and N. Karamanis (2006). A computer-aided environment for generating multiple-choice test items. *Natural language engineering 12*(2), 177–194.

Palmer, M., D. Gildea, and P. Kingsbury (2005). The proposition bank: An annotated

corpus of semantic roles. *Computational linguistics 31*(1), 71–106.

Papineni, K., S. Roukos, T. Ward, and W.-J. Zhu (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pp. 311–318. Association for Computational Linguistics.

Rajpurkar, P., J. Zhang, K. Lopyrev, and P. Liang (2016). Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Redfield, D. L. and E. W. Rousseau (1981). A meta-analysis of experimental research on teacher questioning behavior. *Review of educational research 51*(2), 237–245.

Richardson, M., C. J. Burges, and E. Renshaw (2013). Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 193–203.

Rus, V. and C. G. Arthur (2009). The question generation shared task and evaluation challenge workshop report. In *The University of Memphis. National Science Foundation*. Citeseer.

Rus, V., Z. Cai, and A. C. Graesser (2007). Experiments on generating questions about facts. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pp. 444–455. Springer.

Rus, V., B. Wyse, P. Piwek, M. Lintean, S. Stoyanchev, and C. Moldovan (2010). The first question generation shared task evaluation challenge. In *Proceedings of the 6th International Natural Language Generation Conference*, pp. 251–257. Association for Computational Linguistics.

Rus, V., B. Wyse, P. Piwek, M. Lintean, S. Stoyanchev, and C. Moldovan (2012). A detailed account of the first question generation shared task evaluation challenge. *Dialogue & Discourse 3*(2), 177–204.

Rush, A. M., S. Chopra, and J. Weston (2015). A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Sag, I. A., T. Baldwin, F. Bond, A. Copestake, and D. Flickinger (2002). Multiword expressions: A pain in the neck for nlp. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pp. 1–15. Springer.

Smedt, T. D. and W. Daelemans (2012). Pattern for python. *Journal of Machine Learning Research 13*(Jun), 2063–2067.

Wolfe, J. H. (1976). Automatic question generation from text-an aid to independent study. *ACM SIGCSE Bulletin 8*(1), 104–112.

Yates, T. (2016). Automated generation of questions from factual, natural language sentences.