https://doi.org/10.1051/shsconf/202110201011

Design and development of a question generator for learners of English

Duc Tran Vu[1,*] and *John* Blake[2,**]

[1]Human Interface Lab, University of Aizu, Aizuwakamatsu, Japan
[2]Center for Language Research, University of Aizu, Aizuwakamatsu, Japan

Abstract. In this paper, we describe the design and development of the first release of an online question generator. This pedagogic tool enables learners of English to generate open-ended, closed-ended and tag questions for a target sentence. Learners input a sentence (i.e. declarative statement) and select the type or types of questions to generate. Question generation is a non-trivial task involving numerous processes including syntactic transformation and pronoun selection. Syntactic transformation was achieved through the use of rules based on parse trees while the selection of interrogative pronouns was achieved using matching potential question foci with a linguistic knowledge encoded condition set. Lessons learned are detailed to help other researchers avoid or attempt to ameliorate the pitfalls and problems encountered in this study.

1 Introduction

Questioning has plays a central role in education. The Socratic method developed in Ancient Greece aimed to develop critical thinkers through the use of questions rather than lecturing [1]. Teacher training courses nowadays focus on effective use of questioning techniques [2]. Classroom questions perform various functions including eliciting information, checking knowledge, evoking responses and seeking agreement [3]. Questions comprise of around 15% of all dialogues with that figure rising to 19% in classroom situations, but dropping to 11% in interviews according to a research team at University College London [4] who analyzed the British subcorpus of the International Corpus of English [5]. Given the high frequency of questions, their importance cannot be denied.

An extensive search of the literature revealed that question answering is a popular topic in natural language processing (NLP) and data mining. There is, however, far less research on question generation. Some studies, which are detailed in the related works section, have addressed discrete elements, such as syntactic and semantic patterns. No published papers were discovered that focused on the development of an online question generation tool to help learners of English. Internet searches for online tools that could automatically create questions also proved futile. This study aims to fill this niche in the research literature and provide learners of English with a useful online resource. The Question Generator is the first online tool to enable users to automatically generate questions based on an inputted sentence.

This paper describes the development of the Question Generator. Users input a sentence (i.e. a declarative statement) and select the type of question or questions to be generated. The Question Generator then displays all questions that it can generate based on the input sentence. This pedagogic tool can be used in multiple ways. One use case is for users to create their own questions based on a sentence, and then input the sentence into the Question Generator. The differences between the user-created and automatically-generated questions can be compared.

Section 2 contextualizes this study and describes the problems that learners of English face with regards to learning how to formulate questions and the difficulty facing teachers of English to provide appropriate practice and models. Section 3 describes the related works. However, as this online tool is breaking new ground, the related works focus more on generating answers to questions rather than creating questions from answers. The design of the Question Generator is detailed in Section 4. The development of the codebase is described in Section 5. Section 6 provides a brief conclusion and then lists five lessons that the authors learned through the development of this first prototype. This paper concludes with Section 7, which provides a summary of this project and plans for future work.

2 Background

This section begins by illustrating the types of problems that learners of English face when attempting to create interrogative statements. The causes of the problems are then identified and finally a potential technological solution is suggested.

2.1 Problem

During an ice-breaking activity, each of the following questions was produced by a group of three or four undergraduate computer science majors studying at a public university in Japanese.

*e-mail: s1252012@u-aizu.ac.jp
**e-mail: jblake@u-aizu.ac.jp

(1) Why did you came to Japan?*

(2) Why you come to Japan?*

(3) Why are you come to Japan?*

(4) Why do come to Japan?*

(5) Why did you come to Japan?

An asterisk marks sentences which are ungrammatical. Only example (5) is grammatically accurate. Despite studying English for over six years, these examples show that some learners have difficulty creating even simple questions.

2.2 Complexity

Learners of English need to be able to understand and use questions. Questions often, but not always, take the form of interrogative statements. Many learners of English as an additional language have difficulty formulating interrogative statements. Interrogative statements have been the focus of many linguistic studies. [6–8] created taxonomies based on corpus investigations of the lexical and syntactical features of interrogative statements, revealing their complexity. For example, open-ended questions can be created by replacing the grammatical subjective with an interrogative pronoun. Open-ended questions (e.g. those using *what, who, how,* etc.) differ in grammatical structure depending on whether the focus of the question is on the grammatical subject or object. Closed-ended questions (e.g. those answered by *yes* or *no*) differ in structure dependent on the tense, voice and aspect of the finite verb. Tag questions are arguably the most complex due to the negation of the main clause, selection of a suitable auxiliary verb and appropriate anaphoric pronoun.

In the language classroom, teachers of English may provide learners with controlled question practice using simple substitution and transformation drills that require the learners to devise questions based on a prompt. In more communicative methods, scenarios may be created requiring learners to use questions to complete a task. Communicative activities, however, tend to focus on fluency rather than accuracy. In both the question drills and scenario practice activities, the teacher exerts controls over the content and/or context. An ideal controlled practice activity is one in which learners select their own sentence, create as many questions as possible related to the sentence and then the class teacher checks the accuracy of their questions. In a one-to-one or very small group tuition, this can be achieved, but the activity is not scalable.

2.3 Potential solution

The initial idea for this project stemmed from a common classroom activity in which a teacher of English shows learners a sentence, such as in example (6).

(6) The president had an accident in the morning.

The learners then work individually, in pairs or groups to create as many questions as possible related to the sentence. The sentence prompt provides sufficient content for learners to activate their existing knowledge of question formats. However, given that interests, learning strategies, learning styles and rates of learning differ, the ideal scenario is one in which learners select their own sentence prompt. This is particularly important for young learners and adolescents who are still in the process of forming their identity.

When each student is working on a different sentence, the teacher is unable to provide much personalized advice to learners or show learners which questions could be asked. This is where we can use the scalability of intelligent computer-assisted language learning (iCALL) [9]. Intelligent CALL harnesses the power of NLP to create pedagogic solutions for language learners. Often times this is achieved by using an NLP pipeline that utilizes rule-based or probabilistic pattern matching.

3 Related works

While Automatic Question Generation (AQG) is not a new domain, studies conducted on this area largely diverse. A large proportion of existing works considers AQG as a solution to the continuous need for assessment questions in standardised tests, school and online courses as well as adaptive learning systems [10–13]. With the wave of neural networks, recent research also sees AQG as a method to improve the performance of Question Answering system by producing a large amount of labeled training data with less cost and human effort [14, 15]. As the purpose varies, the form of generated questions and adopted methodology also differs.

Most educational AQG systems focus on the generation of non-trivial factual fill-in-the-blank or multiple choice questions [11, 12], while systems in other areas often aim to generate free response texts [14, 15].

As for the methodology, most common approaches for AQG include syntax based, semantic based and template based. Both syntax and template based approaches rely on the input syntactic structure, with the syntax based approach generating questions by transforming the original structure [12, 13], while the template based approach builds questions by completing some pre-designed templates with components from the input [15]. On the other hand, question construction in the semantic approach is solely based on the input underlying semantic structure, which is usually represented by some object languages [11, 16].

Among all the aforementioned various categories, syntax based free response factual question generation is the best match for our interest, as the questions produced by this type of system allow educators to clearly illustrate the patterns in question formation and selection of interrogative pronouns to elementary level language learners. Heilman [13] presented the most comprehensive analysis on the challenges and possible approaches to syntax based factual question generation. In this paper, we aim to complement the former works by adding some notable issues that have not been investigated, and also to propose some possible improvements over existing methods.

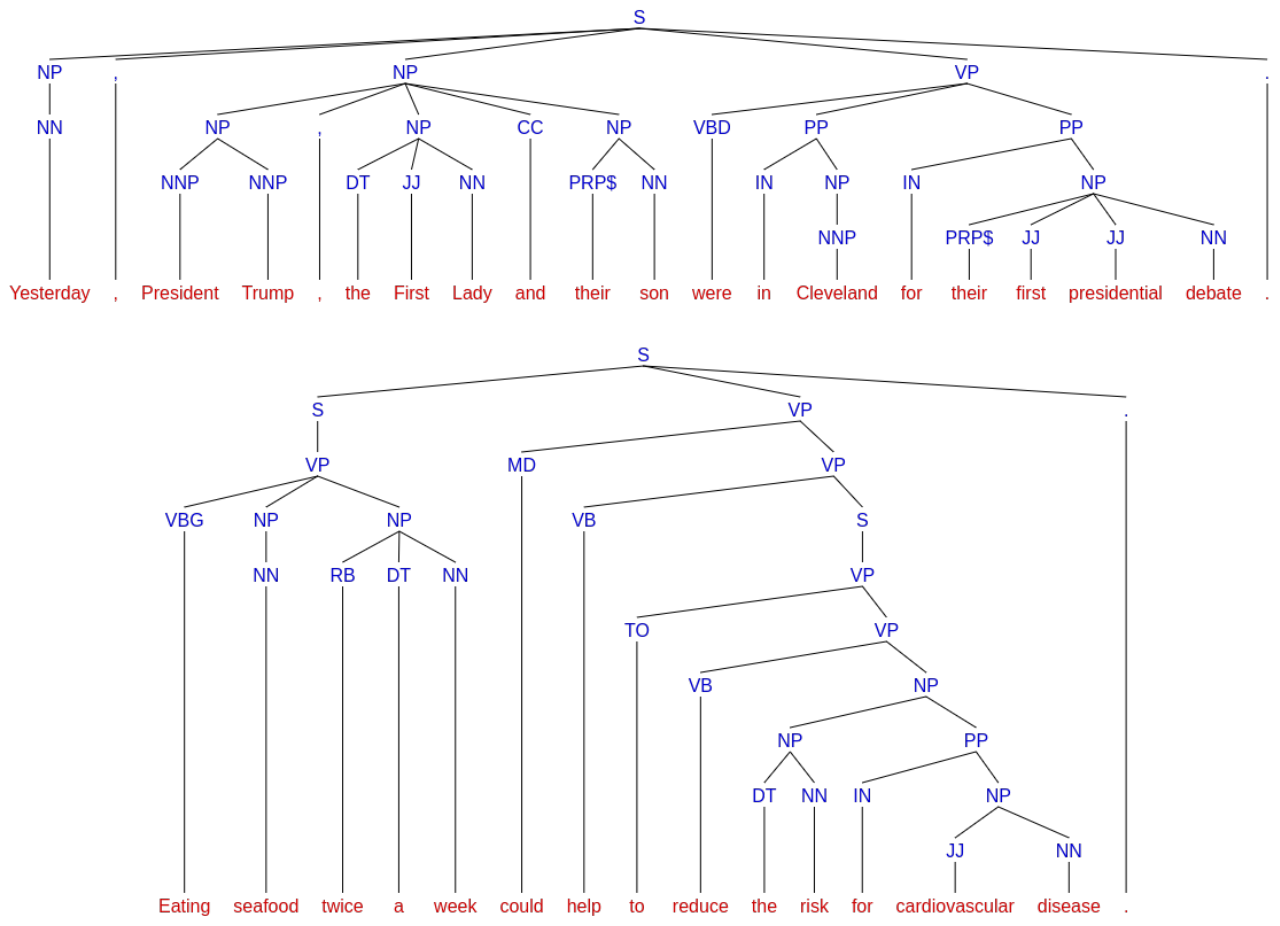
S
NP , NP VP .
NN NP , NP CC NP VBD PP PP
NNP NNP DT JJ NN PRP$ NN IN NP IN NP
NNP PRP$ JJ JJ NN
Yesterday , President Trump , the First Lady and their son were in Cleveland for their first presidential debate .
S
S VP .
VP MD VP
VBG NP NP VB S
NN RB DT NN VP
TO VP
VB NP
NP PP
DT NN IN NP
JJ NN
Eating seafood twice a week could help to reduce the risk for cardiovascular disease .

Figure 1. The syntactic parse trees generated by the Berkeley Parser for the two sentence "*Yesterday, President Trump, the First Lady and their son were in Cleveland for their first presidential debate.*" and "*Eating seafood twice a week could help to reduce the risk for cardiovascular disease.*"

4 Design

We employed a syntax-based approach using transformation rules to create questions from raw text inputs. We aim to generate as many grammatical questions as possible from the given context, with the outputs including three different types: open-ended question, closed-ended question and tag question. As our main focus is on demonstrating the movement of sentence components and the mapping from question foci to interrogative pronouns, we generate questions from a single sentence given by the user, contrary to most text-to-text systems in question generation literature which typically used a body of text as input. We present all the tasks involved in two stages of preprocessing and question construction.

4.1 Preprocessing

In the preprocessing stage, we aim to analyze the sentence syntax and construct a systematic structure convenient for manipulation in later steps. Sentence parse tree is well-known in the research community to be a suitable solution. Constituency parsers have seen major improvement in accuracy over the recent years, with deep neural parsers described in [17–20] surpassing human-level performance. Leveraging these existing tools allows us to build a solid foundation for our system.

We use the authors' Python implementation of Berkeley Neural Parser [17] to generate parse trees for sentences. The implementation provides interfaces for two most popular NLP libraries in Python: NLTK and spaCy, and therefore can be easily integrated with other components of the system.

It should also be noted that while current parsers successfully provide correct syntactic representations for most cases, there are some properties of the syntax tree we need to pay attention to in the downstream tasks.

One problem which usually gets excluded in constituency parser based AQG research is that because the parsers solely focus on tagging sentence's components with their correct phrasal categories while ignoring their syntactic functions, the results can be deceiving, as shown in Figure 1. In these cases, the noun phrases *yesterday* and *twice a week* should be treated as adverbial phrases during question construction process.

Another issue worth mentioning is that due to the nature of the corpus they were trained on, the parsers have a tendency to label infinitive and gerund verb complements and subjects as incomplete sentences (refer to the second parse tree in Figure 1).

We will present our approach to these phenomena in the sections describing the specific algorithms for generating different types of questions.

4.2 Question construction

This section explains our algorithm for constructing questions from the tree representation returned by the parser in stage 4.1. Three types of question are created - each has different syntax tree manipulation rules and requires solving different composition of subtasks. General-purpose tasks shared among question types are described in detail only once when they are first encountered.

4.2.1 Closed-ended question

The process of generating closed-ended questions can be outlined as shown in Algorithm 1.

Algorithm 1: Closed-ended question generation

```
Results ← ∅;
ClosedQuestion (S)
    inputs : A S parse tree for the source text
    if Separable (S) then
        [S] ← Separate (S);
        ▹ Separate independent clauses in compound
          sentences
        foreach $S_n \in [S]$ do
            ClosedQuestion ($S_n$);
            ▹ Create questions in recursion
        end
    end
    Rearrange (S);
    ▹ Rearrange the sentence if necessary
    subject ← ExtractSubject (S);
    Decompose (S);
    ▹ Decompose the main verb
    aux ← ExtractAuxiliary (S);
    S.Invert (subject, aux);
    ▹ Invert the subject and auxiliary verb
    Results ← S;
```

Clause separation Compound sentence can be identified by checking if there are two or more S nodes in the first level children from the root. Each S represents a single independent clause, which can have its own nested structure.
Subject extraction As mentioned in Subsection 4.1, we cannot naively assume the first NP to be the subject of the sentence. Take the first parse tree in Figure 1 as an example. The first NP in the upper-most level is *yesterday*, whose function is an adverbial phrase, while the second NP is *President Trump, the First Lady and their son*, which is the subject of the sentence. Another possibility is when the subject of the sentence is a gerund or an infinitive verb or a composition of any of those, as in the second parse tree of Figure 1. In this case, there is no NP child at the first level of the parse tree, where it is directly under the sentence root S. The subject has the label S instead, for the reason that we mentioned in Subsection 4.1.

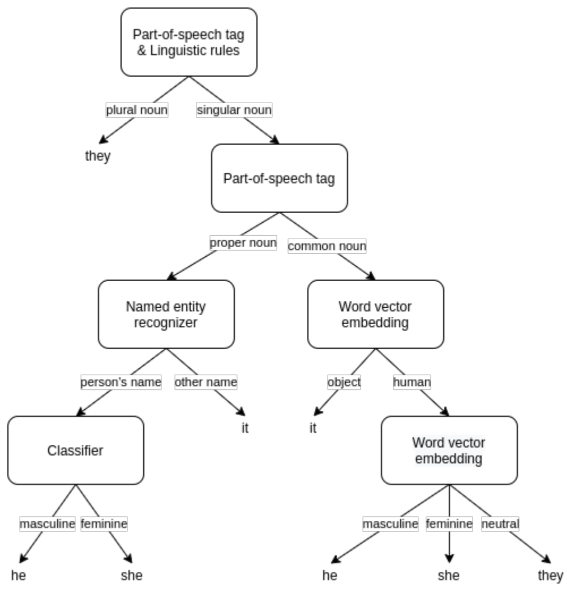

Figure 2. The framework for classifying the sentence subject to the correct pronoun category. The whole pipeline consists of several NLP tools, each in turns assigns a grammatical or semantic role to the subject.

In the first case where there are multiple direct NP of the sentence root, our task is to construct a method to select the sentence subject from these NP. Although the parse tree contains only phrasal labels, according to [21], grammatical relations can be derived from this constituent structure. It can be noticed from the parse trees in Figure 1 that while dependency relations are not explicitly expressed, words and phrases are grouped so that together they make up an unit that carries a single syntactic function (e.g., *President Trump*, *the First Lady* and *their son* belong to the same NP group, but *yesterday* does not.) We can exploit this property to extract the sentence subject by selecting the NP group that is closest to the left of the main verb phrase in the upper-most level.

In the second case where there is a node S at any position before the main verb phrase VP in the first level, we select that node as the subject. The statement above applies for any cases, with or without the presence of an NP node.

After identifying the subject, as a final step, we move all the other NP, along with leading prepositional and adverb phrases, to the end of the sentence, in order to generate slightly more natural questions [13].

4.2.2 Tag question

The process of generating tag questions can be outlined as in Algorithm 2.
Pronoun classification Matching subjects with their correct pronouns is unexpectedly a very challenging task for computers, as it requires not only the knowledge about the subjects' syntactic structure but also the understand-

Algorithm 2: Tag question generation

```
Results ← ∅;
TagQuestion (S)
    inputs : A S parse tree for the source text
    if Separable (S) then
        [S] ← Separate (S);
        foreach S_n ∈ [S] do
            TagQuestion (S_n);
        end
    end
    Rearrange (S);
    subject ← ExtractSubject (S);
    Decompose (S);
    aux ← ExtractAuxiliary (S);
    pronoun ← PronounClassify (subject);
    neg ← Negative (aux);
    Results ← S + neg + pronoun;
```

Table 1. Subject-pronoun agreement rules.

Subject	Pronoun
Singular / Plural `NP` + Complement	Singular / Plural
Singular / Plural Indefinite	Singular / Plural
`NP` *and* `NP`	Plural
Either `NP` *or* Singular / Plural `NP`	Singular / Plural
The number of `NP`	Singular
A number of `NP`	Plural

ing of their meaning. While pronoun resolution is a well-established research field, there has been no significant work on the reverse inference task within our knowledge.

We proposed a framework for this problem based on a composition of several NLP techniques, as showed in Figure 2.

The first step is to determine whether the sentence subject is plural or singular. This can be achieved by deriving a set of generalized rules leveraging linguistic knowledge about subject-verb agreement. Example for some of the cases are showed in Table 1.

Plural subjects are assigned with the pronoun *they*, while singular subjects are separated into *proper nouns* and *common nouns* based on their associated labels in the parse tree. In the case when the subject is a *proper noun*, we use spaCy's implementation of the named entity recognizer (NER) described in [22] to distinguish person names from other entity types. A support vector machine classifier [23, 24] is trained on a self-collected dataset of 7000 Western and 500 Japanese names to identify the most likely gender for the names. *Masculine* or *feminine names* are then mapped to the pronoun *he* or *she* respectively, while any other named entities are mapped to the pronoun *it*.

On the other hand, in the case when the subject is a *common noun*, then it can be any of the three following types. *Gender specific* nouns (e.g., *brother*, *sister*) go with their corresponding gender pronouns, while *gender neutral* nouns ideally are mapped to the generic pronoun *they*. Any nouns which refer to non-human entity will be assigned with the pronoun *it*.

In order to classify common nouns into these categories, we employ word embedding [25] for lexical semantics analysis. For the head noun of the noun phrase, we acquire a 300-dimension vector semantic representation using the pre-trained GloVe model in [26]. The similarity score between the head noun vector and the vector $\overrightarrow{\text{object}}$ is then calculated using the L2 norm and compared against the score between the head noun vector and the vector $\overrightarrow{\text{human}}$. If the head noun vector is closer to the vector $\overrightarrow{\text{object}}$ in the semantic space, we can conclude it refers to an inanimate object and vice versa. Similarly, we can compare the distance from the head noun vector to the vectors $\overrightarrow{\text{male}}$ and $\overrightarrow{\text{female}}$ to figure out its grammatical gender. If the difference is less than a certain threshold, then we may conclude that the noun is gender neutral [27].

4.2.3 Open-ended question

We follow a similar approach as described in [13], with a larger set of rules and a wider range of question types. Before performing any transformation on the syntax tree, we use spaCy's NER to assign entity annotations to all the words in the source sentence. These annotations will be used in combination with GloVe vector semantic representation and sentence syntactic information to map potential question foci to WH-words.

The process of generating open-ended questions can be outlined as in Algorithm 3.

Answer selection At the linguistic level, the tool creates questions for the subject, all indirect and direct objects, subjective and objective complements as well as adverbial modifiers directly under any independent clauses in the source sentence. At the syntactic level, it is equivalent to creating questions for all direct `NP`, `ADVP`, `PP` and `SBAR` children of the root `S`, as well as all `NP`, `ADJP`, `PP` and `SBAR` sibling nodes of the main verb. Note that any direct `NP` child of the root `S`, except for the subject, should be treated as an adverbial phrase, as mentioned in Subsection 4.1.

Question phrase generation Candidate answers extracted by the above selection process are matched with their question phrases using the rules listed in Table 2. Words, phrases and clauses that do not satisfy any of the rules are ignored. Dummy subjects (e.g., *there, it*) are also excluded in this process.

For `NP` which contain multiple children of the same types, we recursively traverse down the left-most `NP` child until we reach the leaves of the syntax tree. We then generate the interrogative pronouns by matching the last `NP` in the path using the above rules. For the special case of noun phrase with partitive construction, we follow [13] and use the object of the adjoined prepositional phrase to generate the interrogative pronoun (e.g., *one of* ***John's friends***).

5 Development

The system consists of two main components: the back-end and the web interface. The logic for question gener-

Algorithm 3: Open-ended question generation

```
$Results \leftarrow \emptyset$;
OpenQuestion (S)
    inputs : A S parse tree for the source text
    if Separable (S) then
        [S] ← Separate (S);
        foreach $S_n \in [S]$ do
            OpenQuestion ($S_n$);
        end
    end
    Rearrange (S);
    $subject \leftarrow$ ExtractSubject (S);
    if not DummySubject (subject) then
        $qstPhrase \leftarrow$ QuestionPhrase (subject);
        ▹ Match subject with its corresponding
          interrogative phrase
        $S^* \leftarrow S$;
        ▹ Create a copy of the parse tree
        $S^*$.Replace (subject, qstPhrase);
        $Results \leftarrow S^*$;
    end
    Decompose (S);
    $aux \leftarrow$ ExtractAuxiliary (S);
    S.Invert (subject, aux);
    [A] ← ExtractAnswer (S);
    ▹ Extract all the answer candidates in the sentence
    foreach $A_n \in [A]$ do
        $qstPhrase \leftarrow$ QuestionPhrase ($A_n$);
        $S^* \leftarrow S$;
        $S^*$.Remove ($A_n$);
        $Results \leftarrow qstPhrase + S^*$;
    end
```

ation described in earlier sections is implemented in the server back-end. Users of the system can communicate with the application through the web interface by inputting a raw text sentence and select desired question types.

The web application was developed with Python using Flask micro-framework. It provides two HTTP methods `GET` and `POST`. The `GET` method responses with the HTML index page and the related static files, while the `POST` method receives requests with input data and returns JSON responses containing the generated questions.

We used Angular framework to manipulate HTML components dynamically. When users submit their inputs, the Angular controller sends an asynchronous request to the web server and renders the response on the result box without reloading the web page. If the users enter another sentence to the input bar, the result box is cleared automatically.

For memory-intensive resources such as the parser pretrained model and word embedding matrix, we initialize and load them into memory at start time before any requests arrive to reduce application response time.

6 Conclusion and lessons learned

The Question Generator is the first online pedagogic tool designed to help learners by provided unlimited examples of questions based on a stimulus sentence. There are, however, still a number of technical challenges to be resolved.

This project taught us five lessons, each of which is itemized below so that developers of similar systems can benefit from our experience and climb the learning curve prior to commencing their project.

Lesson 1***:*** *Expectation versus reality*

We naively approached the problem of question generation, and failed to estimate the true complexity of the task. Specifically, we underestimated the difficulty in selecting an appropriate interrogative pronoun. It is easy to explain the usage of interrogative pronouns. For example, we use *what* when asking about *things* and *who* when asking about *people*. However, when faced with a sentence such as in (7), the system needs to identity that the *Mary* is a person, while *them* and *her dog* are things.

(7) Mary wants to give them to her dog.

Another difficulty is that interrogative pronouns combine with adjectives to create more specific meanings. For example *how long* asks about length while *how old* asks about age.

Lesson 2***:*** *Law of diminishing returns*

It is relatively straightforward to generate interrogative statements from declarative statements with reasonable accuracy. However, to do so with a high degree of accuracy is extremely challenging. Natural language processing pipelines adhere to the law of diminishing returns, and so a cost-benefit analysis needs to be conducted to establish a value at which the accuracy of the program can be judged as sufficient for its purpose.

Lesson 3***:*** *Shortcomings of existing tools*

In the same vein as many other natural language processing pipelines, this system is dependent on resources created by third parties. This means that regardless of the accuracy rate of our system, inaccuracies introduced by these dependencies will impinge of the accuracy of our system.

Lesson 4***:*** *Necessity to define target users*

During the development of the code, it became necessary to more accurately identify the primary users. One example of this necessity was when dealing with name recognition. Creating a system that recognises only English names is easier than one that recognises both Japanese and English names, while a system that needs to recognise any name requires even more resources.

Label	**Additional condition**	**Question words**
`ADJP`		*how*
`ADVP`	Head word is in the adverbs of time list	*when*
	Head word is in the adverbs of place list	*where*
	Head word is in the adverbs of frequency list	*how often*
`SBAR`	Subordinate clause is a direct child of the sentence root `S` and its subordinating conjunction is a WH-word	WH-word
	Subordinate clause is a sibling node of the main verb and its head word is a gerund or an infinitive verb (refer to the phenomena described in 4.1)	*what ... to do*
	Subordinate clause is a sibling node of the main verb	*what*
`NP`	Head word is an entity of type `LOC` or `GPE`	*where*
	Head word is an entity of type `PERSON`	*who*
	Head word is an entity of type `DATE`	*when*
`NP` (Sentence subject)	Head word is an named entity of any other types	*what*
	Distance from the head word vector to the vector $\overrightarrow{\text{object}}$ in the semantic space is less than the distance to the vector $\overrightarrow{\text{human}}$	*what*
	Distance from the head word vector to the vector $\overrightarrow{\text{human}}$ in the semantic space is less than the distance to the vector $\overrightarrow{\text{object}}$	*who*
`NP` (Other)	Head word is in the adverbs of time list (refer to the phenomena described in 4.1)	*when*
	Head word is in the adverbs of place list (refer to the phenomena described in 4.1)	*where*
	Head word is in the adverbs of frequency list (refer to the phenomena described in 4.1)	*how often*
	Head word is an entity of type `MONEY`	*how much*
	Head word is an entity of type `TIME`	*how long*
	Modifier has one of the following labels `CD, QP` and the head word has the label `NNS`	*how many* + head noun
	Modifier is in the uncountable quantifiers list	*how much* + head noun
	Modifier is either *some* or *any* and the head word has the label `NNS`	*how many* + head noun
	Modifier is either *some* or *any* and the head word has the label `NN`	*how much* + head noun
	Head word is the pronoun *it*	*what*
	Head word is the any other `PRP`	*who*
	Distance from the head word vector to the vector $\overrightarrow{\text{object}}$ in the semantic space is less than the distance to the vector $\overrightarrow{\text{human}}$	*what*
	Distance from the head word vector to the vector $\overrightarrow{\text{human}}$ in the semantic space is less than the distance to the vector $\overrightarrow{\text{object}}$	*who*
`PP`	Object of the preposition is an entity of type `TIME` or `DATE` and the preposition is one of the following: *in, at, on, before, after, from*	*when*
	Object of the preposition is an entity of type `TIME` and the preposition is *for*	*how long*
	Preposition is one of the following *on, in, at, under, over, inside, outside, above, below*	*where*

Table 2. The rule set that maps potential question foci to their corresponding interrogative phrases. Candidate answers are checked with every rule in the table following the listed order until the first match.

Lesson* 5*: Speed versus accuracy

The trade-off between speed and accuracy needs to be considered. Would a user prefer to get 10 questions with negligible wait time or 15 questions with a noticeable wait. This decision requires assessment of the user requirements and the available resources (e.g. human, time and financial). For this project, we aimed to provide optimal results while minimizing wait time.

7 Future work

Our next goal is to gain a clearer picture of the accuracy of the system. Once the precision of the system is known the main priority will be to minimize the number of false positive and false negative results. To make the tool more user-friendly, colour coding could be adopted to help users understand elements in the original statement have been reused, transformed or supplemented. This could help users notice the patterns and induce rules of syntax more easily.

Looking at the questions generated, it is immediately noticeable that the lack of capability to recognize and process question foci with high level of semantic depth (e.g. *how*, *why*, *which season*, *what country*) is one of the main factors hindering the productivity and performance of the system. Furthermore, word-based semantic representation like GloVe alone is inadequate, as the lexical semantics of the question foci sometimes depends on its surrounding context. For example, the phrase *family* in *Charles is living with his family*, *Family is the most important thing in life* and *I can't wait to see my family* ideally should have different embeddings, as they result in different interrogative pronouns *where*, *what* and *who* respectively.

A possible solution for both of the mentioned issues would be to leverage the novel Bi-directional Encoder Representations from Transformers (BERT) [28] pre-trained language model. Transformer [29] is a neural network architecture which uses self-attention mechanism to model the dependency among tokens in a sequence, instead of relying on conventional sequential dependency as in recurrent architecture. BERT utilizes the encoder in the transformer to build a language model, which is then trained on huge corpora of unlabelled text data using special artificial tasks. This training process encodes a great amount of language knowledge into the model, which then can be transferred to other NLP tasks.

Unlike previous pre-trained language representation methods like word2vec or GloVe, BERT takes into account both the left and the right context when producing embedding for a token, which will allow us to extract contextualized representations for our potential answer phrases. The feature vectors can then be used as inputs to train an encoder-decoder recurrent model (RNN, LSTM, GRU) which outputs the corresponding interrogative phrases. The data required for training the model can be achieved by restructuring existing corpora on question answering such as SQuAD or CoQA [30, 31].

Once a sufficiently accurate and user-friendly system is operational, the pedagogic efficacy needs to be measured empirically. However, in the meantime the Question Generator is deployed online and open access.

References

[1] J.T. Chowning, Science teacher (Normal, Ill.) **76**, 36 (2009)

[2] M. Swan, *The Practice of English Language Teaching* (Oxford University Press, Oxford, 2018)

[3] T. Morell, English for specific purposes **23**, 325 (2004)

[4] B. Aarts, S. Wallis, I. Cushing, *Englicious: English language resources for schools* (2020), `http://www.englicious.org/`

[5] S. Greenbaum, English Today **7**, 3 (1991)

[6] D. Biber, S. Johansson, G. Leech, S. Conrad, E. Finegan, *Longman grammar of spoken and written English.* (Longman, London, 1999)

[7] R. Carter, M. McCarthy, *Cambridge grammar of English: a comprehensive guide; spoken and written English grammar and usage* (Cambridge University Press, Cambridge, 2006)

[8] R. Quirk, S. Greenbaum, *A university grammar of English* (Longman, 1993)

[9] J. Blake, in *New Technological Applications for Foreign and Second Language Learning and Teaching*, edited by M. Kruk, M. Peterson (IGI Global, 2020), pp. 1–23

[10] N. Afzal, R. Mitkov, Soft Computing **18**, 1269 (2014)

[11] R. Ai, S. Krause, W. Kasper, F. Xu, H. Uszkoreit, *Semi-automatic generation of multiple-choice tests from mentions of semantic relations*, in *Proceedings of the 2nd Workshop on Natural Language Processing Techniques for Educational Applications* (2015), pp. 26–33

[12] C.Y. Chen, H.C. Liou, J.S. Chang, *FAST–An Automatic Generation System for Grammar Tests*, in *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions* (2006), pp. 1–4

[13] M. Heilman, Ph.D. thesis, Carnegie Mellon University (2011)

[14] N. Duan, D. Tang, P. Chen, M. Zhou, *Question generation for question answering*, in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (2017), pp. 866–874

[15] A.R. Fabbri, P. Ng, Z. Wang, R. Nallapati, B. Xiang, arXiv preprint arXiv:2004.11892 (2020)

[16] X. Yao, G. Bouma, Y. Zhang, Dialogue & Discourse **3**, 11 (2012)

[17] N. Kitaev, D. Klein, *Constituency Parsing with a Self-Attentive Encoder*, in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (2018), pp. 2676–2686

[18] J. Zhou, H. Zhao, *Head-Driven Phrase Structure Grammar Parsing on Penn Treebank*, in *Proceedings*

of the 57th Annual Meeting of the Association for Computational Linguistics (2019), pp. 2396–2408

[19] K. Mrini, F. Dernoncourt, Q.H. Tran, T. Bui, W. Chang, N. Nakashole, *Rethinking self-attention: Towards interpretability in neural parsing*, in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings* (2020), pp. 731–742

[20] D. Klein, C.D. Manning, *Fast exact inference with a factored model for natural language parsing*, in *Proceedings of the 15th International Conference on Neural Information Processing Systems* (2002), pp. 3–10

[21] N. Chomsky, *Aspects of the Theory of Syntax*, Vol. 11 (MIT press, 2014)

[22] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, C. Dyer, *Neural Architectures for Named Entity Recognition*, in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2016), pp. 260–270

[23] C. Cortes, V. Vapnik, Machine learning **20**, 273 (1995)

[24] V. Vapnik, *The nature of statistical learning theory* (Springer science & business media, 2013)

[25] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean, *Distributed representations of words and phrases and their compositionality*, in *Advances in neural information processing systems* (2013), pp. 3111–3119

[26] J. Pennington, R. Socher, C.D. Manning, *Glove: Global vectors for word representation*, in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (2014), pp. 1532–1543

[27] T. Bolukbasi, K.W. Chang, J.Y. Zou, V. Saligrama, A.T. Kalai, Advances in Neural Information Processing Systems (2016)

[28] J. Devlin, M.W. Chang, K. Lee, K. Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (2019), pp. 4171–4186

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Advances in neural information processing systems **30**, 5998 (2017)

[30] P. Rajpurkar, J. Zhang, K. Lopyrev, P. Liang, *SQuAD: 100,000+ Questions for Machine Comprehension of Text*, in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (2016), pp. 2383–2392

[31] S. Reddy, D. Chen, C.D. Manning, Transactions of the Association for Computational Linguistics **7**, 249 (2019)