Tag-Set-Sequence Learning for Generating Question-answer Pairs

Cheng Zhang and Jie Wang
Department of Computer Science, University of Massachusetts, Lowell, MA 01854, U.S.A.

Keywords: Tag-Set-Sequence Learning, Question-answer Pairs, Natural Language Processing.

Abstract: Transformer-based QG models can generate question-answer pairs (QAPs) with high qualities, but may also generate silly questions for certain texts. We present a new method called tag-set sequence learning to tackle this problem, where a tag-set sequence is a sequence of tag sets to capture the syntactic and semantic information of the underlying sentence, and a tag set consists of one or more language feature tags, including, for example, semantic-role-labeling, part-of-speech, named-entity-recognition, and sentiment-indication tags. We construct a system called TSS-Learner to learn tag-set sequences from given declarative sentences and the corresponding interrogative sentences, and derive answers to the latter. We train a TSS-Learner model for the English language using a small training dataset and show that it can indeed generate adequate QAPs for certain texts that transformer-based models do poorly. Human evaluation on the QAPs generated by TSS-Learner over SAT practice reading tests is encouraging.

1 INTRODUCTION

Multiple-choice questions (MCQs) are often used to assess if students understand the main points of a given article. An MCQ consists of an interrogative sentence, a correct answer (aka. answer key), and a number of distractors. A QAP is an interrogative sentence and its answer key. We study how to generate QAPs from declarative sentences.

The coverage of the main points of an article may be obtained by selecting important declarative sentences using a sentence ranking algorithm such as CNATAR (Contextual Network and Topic Analysis Rank) (Zhang et al., 2021) on the article. QAPs may be generated by applying a text-to-text transformer on a declarative sentence, a chosen answer key, and the surrounding chunk of sentences using, for example, TP3 (Transformer with Preprocessing and Postprocessing Pipelines) (Zhang et al., 2022), which generates interrogative sentences for the answer keys with much higher success rates over previous methods. TP3, however, may generate silly QAPs for certain chunks of texts.

Training a deep learning model like TP3 for generating QAPs may be viewed as learning to speak in a language environment, akin to how kids learn to talk from their environment. On the other hand, learning to write well would require kids to receive formal educations. This analogy motivates us to explore machine-learning mechanisms that could mimic explicit rule learning.

To this end we devise a method using a tag-set sequence to represent the pattern of a sentence, where each tag set consists of a few language-feature tags for the underlying word or phrase in the sentence. Given a declarative sentence and a corresponding interrogative sentence, we would like to learn the tag-set sequences for each of these sentences and derive the tag-set sequence for the answer key, so that when we are given a new declarative sentence as input, we could generate a QAP by first searching for a learned tag-set sequence that matches the tag-set sequence of the input sentence, then use the learned tag-set sequence of the corresponding interrogative sentence to map the context of the input sentence to produce an interrogative sentence and derive a correct answer.

We construct a general framework called TSS-Learner (Tag-Set-Sequence Learner) to carry out this learning task. We train a TSS-Learner model for the English language over a small initial training dataset using SRL (semantic-role-label), POS (part-of-speech), and NER (named-entity-recognition) tags, where each data entry consists of a well-written declarative sentence and a well-written interrogative sentence. We show that TSS-Learner can generate adequate QAPs for certain texts on which TP3 generates silly questions. Moreover, TSS-Learner can generate efficiently a reasonable number of adequate QAPs.

Zhang, C. and Wang, J.
Tag-Set-Sequence Learning for Generating Question-answer Pairs.
DOI: 10.5220/0011586800003335
In *Proceedings of the 14th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2022) - Volume 1: KDIR*, pages 138-147
ISBN: 978-989-758-614-9; ISSN: 2184-3228

On QAPs generated from the official SAT practice reading tests, evaluations by human judges indicate that 97% of the QAPs are both grammatically and semantically correct.

The initial training dataset is not big enough to contain tag-set sequences that match the tag-set sequences of a number of declarative sentences in the passages of the SAT reading tests. We manually add new interrogative sentences for some of these declarative sentences, and show that TSS-Learner is able to learn new rules and use them to generate additional adequate QAPs.

The rest of the paper is organized as follows: We summarize in Section 2 related work and present in Section 3 a general framework of tag-set-sequence learning. We then present in Section 4 an implementation of TSS-Learner for the English language and describe evaluation results in Section 5. Section 6 concludes the paper.

2 RELATED WORK

Automatic question generation (QG), first studied by Wolfe (Wolfe, 1976) as a means to aid independent study, has attracted much research in two lines of methodologies with a certain success; they are transformative and generative methods.

2.1 Transformative Methods

Transformative methods transform key phrases from a single declarative sentence into interrogative sentences, including rule-based, semantics-based, and template-based methods.

Rule-based methods parse sentences using a syntactic parser to identify key phrases and transform a sentence to an interrogative sentence based on syntactic rules, including methods to identify key phrases from input sentences and use syntactic rules for different types of questions (Varga and Ha, 2010), generate QAPs using a syntactic parser, a POS tagger, and an NE analyzer (Ali et al., 2010), transform a sentence into a set of interrogative sentences using a series of domain-independent rules (Danon and Last, 2017), and generate questions using relative pronouns and adverbs from complex English sentences (Khullar et al., 2018).

Semantics-based methods create interrogative sentences using predicate-argument structures and semantic roles (Mannem et al., 2010), semantic pattern recognition (Mazidi and Nielsen, 2014), subtopics based on Latent Dirichlet Allocation (Chali and Hasan, 2015), or semantic-role labeling (Flor and Riordan, 2018).

Template-based methods are used for special-purpose applications with built-in templates, including methods based on Natural Language Generation Markup Language (NLGML) (Cai et al., 2006), on phrase structure parsing and enhanced XML (Rus et al., 2007), on self questioning (Mostow and Chen, 2009), on enhanced self-questioning (Chen, 2009), on pattern matching and templates similar to NLGML (Wyse and Piwek, 2009), on templates with placeholder variables (Lindberg, 2013), and on semantics turned to templates (Lindberg et al., 2013).

2.2 Generative Methods

Recent advances of deep learning have shed new light on generative methods. For example, the attention mechanism (Luong et al., 2015) is used to determine what content in a sentence should be asked, and the sequence-to-sequence (Bahdanau et al., 2014; Cho et al., 2014) and the long short-term memory (Sak et al., 2014) mechanisms are used to generate each word in an interrogative sentence (see, e.g., (Du et al., 2017; Duan et al., 2017; Harrison and Walker, 2018; Sachan and Xing, 2018)). These models, however, only deal with question generations without generating correct answers. Moreover, training these models require a dataset comprising over 100K interrogative sentences.

To generate answers, researchers have explored ways to encode a passage (a sentence or multiple sentences) and an answer word (or a phrase) as input, and determine what interrogative sentences are to be generated for a given answer (Zhou et al., 2018; Zhao et al., 2018; Song et al., 2018). Kim et al. (Kim et al., 2019) pointed out that these models could generate a number of answer-revealing questions (namely, questions contain in them the corresponding answers). They then devised a new method by encoding answers separately, at the expense of having substantially more parameters. This method, however, suffers from low accuracy, and it is also unknown whether the generated interrogative sentences are grammatically correct.

Recently, a new method is presented to perform a downstream task of transformers with preprocessing and postprocessing pipelines (TP3) for generating QAPs (Zhang et al., 2022). They showed that TP3 using pretrained T5 models (Raffel et al., 2020) outperforms previous models. Human evaluations also confirm the high qualities of QAPs generated by this method. However, TP3 may generate silly questions for certain chunk of texts. This calls for further in-

vestigations for improving the qualities of generated QAPs.

3 GENERAL FRAMEWORK

Let L be a natural language. Without loss of generality, we assume that L has an oracle O_L that can perform the following tasks:

1. Identify simple sentences with no subordinate clauses and complex sentences with subordinate clauses.
2. Segment a complex sentence into simple sentences for each clause.
3. Segment a sentence into a sequence of basic units, where a basic unit could be a word, a phrase, or a subordinate clause.
4. Assign each basic unit in a sentence with one or more feature tags including, but not limited to, POS, NER, SRL, and SEI (sentiment-indicator) tags.

Existing NLP tools for the English language, for example, provide a reasonable approximation to such an oracle.

3.1 Definitions

Definition 1. *Let $k \geq 2$ be a number of tags that O_L can assign to a basic unit. A k-tag set is a set of k tags, denoted by $[t_1/t_2/\cdots/t_k]$ with a fixed order of tags: t_1 is an SRL tag, t_2 is a POS tag, t_3 is an NER tag, and t_i ($i > 3$) represent the other tags or a special word such as an interrogative pronoun.*

Two consecutive tag sets A and B with $A.1 = B.1$ (i.e., they have the same SRL tag) and A is left to B in a sentence may be merged to a new tag set C as follows: (1) If $A = B$, then let $C \leftarrow A$. (2) Otherwise, based on the underlying language L, either let $C.2 \leftarrow A.2$ (i.e., use the POS tag on the left) or let $C.2 \leftarrow B.2$. For the rest of the tags in C, select a corresponding tag in A or B according to L. We have the following proposition: For any sequence of tag sets, after merging, the new sequence of tag sets does not have two consecutive tag sets with the same SRL tag.

Definition 2. *A tag-set sequence is a sequence of interrogative pronouns (if any) and tag sets such that each SRL tag appears in at most one tag set.*

Remark. To accommodate improper segmentation of phrasal verbs in applications, we may modify this definition of tag-set sequence by allowing a fixed number of consecutive tag sets to have the same SRL tag.

Since O_L can segment a complex sentence into simple sentences for each clause, we treat such a sentence as a set of simple sentences. If a clause itself is a complex sentence, it can be further segmented as a set of simple sentences. A declarative sentence consists of at least three different SRL tags corresponding to subject, object, and predicate.

3.2 Architecture

TSS-Learner learns tag-set sequences from a training dataset, where each data entry consists of a simple declarative sentence and a interrogative sentence. It consists of two phases: the learning phase and the generation phase. In the learning phase, TSS-Learner learns tag-set sequence pairs from an initial training dataset to generate an initial TS-sequence pair database (TSSP-DB). In the generation phase, it takes a declarative sentence as input and generates QAPs using TSSP-DB. Figure 1 depicts the architecture and data flow of TSS-Learner, which consists of five components: Preprocessor, TS-Sequence Generator, Duplicate Remover, TS-Sequence Matcher, and QAP Generator (see Section 4 for detailed explanations of these components in connection to an implementation of TSS-Learner for the English language).

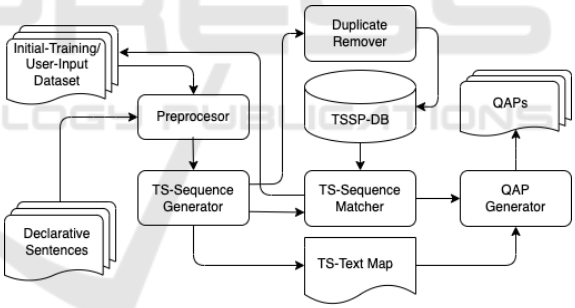

Figure 1: TSS-Learner architecture and data flow.

3.3 Learning Phase

The learn phase and the generation phase use the same Preprocessor and TS-Sequence Generator. Preprocessor is responsible for creating tag sets for a given sentence and segmenting complex sentences into a set of simple sentences. TS-Sequence Generator is responsible for merging tag sets to form a tag-set sequence. Moreover, for an input sentence in the generation phase, it also maps each tag set after merging to the underlying text to be used later for generating QAPs.

The Duplicate Remover component checks if a new pair of tag-set sequences generated by TS-Sequence Generator, one for a declarative sentence and the other the corresponding interrogative sen-

tence, is already in TSSP-DB. If yes, ignore it. Otherwise, deposit it in TSSP-DB.

3.4 Generation Phase

Let T be a tag-set sequence and T' the set of tag sets contained in T. Denote by $|T'|$ the size of T'. Let X_s be the tag-set sequence for s generated by TS-Sequence Generator, Recall that the text for each tag set is stored in the TS-Text Map.

Step 1. Find a tag-set sequence X for a declarative sentence in TSSP-DB with the *best match* of X_s, meaning that the longest common substring of X and X_s, denoted by $Z = \text{LCS}(X, X_s)$, is the longest among all tag-set sequences in TSSP-DB. A substring is a sub-sequence of consecutive tag sets. If Z contains tag sets for, respectively, a subject, a verb, and an object, then we say that it is a *successful* match. If furthermore, $Z = X = X_s$, then we say that it is a *perfect* match. If Z is missing a subject tag set, a verb tag set, or an object tag set, then it is an *unsuccessful* matching. If a match is successful, go to Step 2. If a match is unsuccessful or successful but not perfect, then notify the user that TSS-Learner needs to learn a new pattern and ask for interrogative sentences for s from the user. Go to Step 2.

Step 2. Recall that X is the best match with X_s and there may be multiple pairs (X, Y) in TSSP-DB because multiple interrogative sentences may be generated from the same declarative sentence. Generate all possible interrogative sentences for s as follows: For each pair of tag-set sequences $(X, Y) \in$ TSSP-DB, generate a tag-set sequence Y_s from Y with

$$Y'_s = [Y' - (X' \cap Y' - X'_s)] \cup (X'_s - Z').$$

This means that Y'_s is obtained from Y' by removing tag sets that are in both tag-set sequences in the matched pair (X, Y), but not in the input sentence, and adding tag sets in the input sentence but not in $Z = \text{LCS}(X, X_s)$. We have

$$X'_s - Z' = X'_s - X'.$$

Order tag sets in Y'_s to form Y_s, which may require localization according to the underlying language. If a tag set in Y'_s has the corresponding text stored in the TS-Text Map, then replace it with the text. If not, then it would require localization to resolve it. This generate an interrogative sentence Q_s for s.

Step 3. For each interrogative sentence Q_s generated in Step 2, the tag sets in $A'_s = X' - Y'$ represent a correct answer. Place tag sets in A'_s in the same order as in X'_s and replace each tag set with the corresponding text in s to obtain an answer A_s for Q_s.

4 TSS-LEARNER FOR ENGLISH

SRL, POS, and NER tags are used in this implementation. Existing NLP tools for generating these tags are for words, not for phrases. Proper phrase segmentation can resolve this by merging. In particular, it is critical to obtain segmentation for phrasal verbs for generating interrogative sentences. A phrasal verb consists of a preposition or an adverb, or both. A straightforward method to segment phrasal verbs is to use an extensive list of phrasal verbs.

4.1 Essential NLP Tools

We use the following NLP tools to generate tags: Semantic-Role Labeling (Shi and Lin, 2019) for SRL tags, POS Tagging (Toutanova et al., 2003) for POS tags, and Named-Entity Recognition (Peters et al., 2017) for NER tags.

SRL tags are defined in PropBank[1] (Bonial et al., 2012; Martha et al., 2005), which consist of three types: ArgN (arguments of predicates), ArgM (modifiers or adjuncts of the predicates) , and V (predicates). ArgN consists of six tags: ARG0, ARG1, ..., ARG5, and ArgM consist of multiple subtypes such as LOC as location, EXT as extent, DIS as discourse connectives, ADV as general purpose, NEG as negation, MOD as modal verb, CAU as cause, TMP as time, PRP as purpose, MNR as manner, GOL as goal, and DIR as direction.

POS tags[2] are defined in the Penn Treebank tagset (Toutanova et al., 2003; Marcus et al., 1993). For example, NNP is for singular proper noun, VBZ for third-person-singular-present-tense verb, DT for determiner, and IN for preposition or subordinating conjunction.

NER tags include PER for persons, ORG for organization, LOC for locations, and numeric expressions for time, date, money, and percentage.

4.2 Preprocessor and TS-Sequence Generator

On top of what is described in Section 3, Preprocessor first replaces contractions and slangs with complete words or phrases to help improve tagging accuracy. For example, contractions *'m, 's, 're, 've, n't, e.g., i.e., a.k.a.* are replaced by, respectively, *am, is, are, have, not, for example, that is, also known as.* Slangs

[1] https://verbs.colorado.edu/ mpalmer/projects/ace/EPB-annotation-guidelines.pdf

[2] https://www.cs.upc.edu/ nlp/SVMTool/PennTreebank.html

Abraham [ARG1/NNP/PER/] Lincoln [ARG1/NNP/PER/] was [V/VBZ//] the [ARG2/DT//] 16th [ARG2/JJ//] president [ARG2/NN//] of [ARG2/IN//] the [ARG2/DT//] United [ARG2/NNP/LOC/]

Figure 2: An example of a sentence and tag sets, where the tag set for each word is listed right after the word.

gonna, wanna, gotta, gimme, lemme, ya are replaced by, respectively, *going to, want to, got to, give me, let me, you*. It then segments sentences and tags words in sentences using SRL, POS Tagging, and NER for the training dataset and later for input sentences for generating QAPs. It also uses SRL to segment a complex sentence into a set of simple sentences and discards all simple sentences with a subject or an object missing. Moreover, for each sentence, it removes all the words with a CC (coordinating conjunction) as POS tag before its subject, including *and, but, for, or, plus, so, therefore, because*.

TS-Sequence Generator merges the tag sets for words in each basic unit as follows: (1) If the unit contains a V-tag set (i.e., a tag set with V as the SRL tag), then use this V-tag set for the entire unit. (2) If the unit contains no V-tag set, then it must contain a noun; use the right most tag set that contains a noun POS tag.

It then merges the remaining tag sets if two consecutive tag sets are identical. If they are not identical but have the same SRL tag, then use this SRL tag in the merged tag set, and the POS tag in the from the rightmost tag set. Moreover, the NER tag in the merged tag set is null if both tag sets contain no NER tags; otherwise, use the rightmost non-empty NER tag.

4.3 TS-Sequence Matcher

This component takes a tag-set sequence X_s of a sentence s as input and executes Step 1 in the generation phase described in Section 3.4. The Suffix-Tree algorithm (Ukkonen, 1985) is used to compute a longest common substring of two tag-set sequences, which runs in linear time. The POS tags NN, NNP, NNS, and NNPS for various types of nouns are treated equal. The POS tags VBP and VBZ for third-person-singular-present verbs are treated equal.

Each tag set is in the form of $[t_1/t_2/t_3/t_4]$, where t_1, t_2, t_3, t_4 represent an SRL tag, a POS tag, an NER tag, and an interrogative pronoun, and the latter two tags could possibly be null. Fig. 2 displays the sentence "Abraham Lincoln was the 16th president of the United States" and the tag set for each word.

The resulting tag-set sequence for this sentence, after merging, is the following:
[ARG1/NNP/PER/] [V/VBZ//] [ARG2/NNP/LOC/].

4.4 QAP Generator

QAP Generator executes Steps 2–3 in the generation phase described in Section 3. Recall that $Z = \text{LCS}(X, X_s)$ is the longest match among all $(X, Y) \in$ TSSP-DB. After Y'_s is generated, we form Y_s as follows:

Case 1: $Z = X_s$. Then $Y_s = Y$.

Case 2: Z is a proper substring of X_s. Then each tag set in $X'_s - Z$ appears either before Z or after Z. Let Y_b and Y_a denote, respectively, the the tag set that appear before and after Z in the same order as they appear in X_s. Let

$$Y_s = [Y - (X' \cap Y' - X'_s)]Y_a Y_b,$$

where $Y - (X' \cap Y' - X'_s)$ means to remove from Y the tag sets in $X' \cap Y' - X'_s$.

For each tag set in Y_s, if a corresponding text can be found in the TS-Text Map, then replace it with the text. A tag set that does not have a matched text in the TS-Text Map is the extra helping verbs added to the interrogative sentence that generates Y. There are five POS tags for verbs: VBG for gerund or present participle, VBD for past tense, VBN for past participle, VBP for non-third-person singular present, and VBZ for third-person singular present. Present participle, past participle, and the negative forms of past tense and present tense come with helping verbs. Thus, only positive forms of the past tense VBD and the present tense VBP and VBZ do not come with helping verbs.

Suppose that there are two V-tag sets in Y, then the first V-tag set is for a help verb. If it does not have a matching text in TS-Text Map, then it is an added helping word. In this case, check the POS tag in the ARG0-tag set and determine if it is singular or plural. Then check the POS tag in the first V-tag set in Y to determine the tense. Use the information of these POS tags to determine the correct form of the helping verb, and replace the second V-tag set with the matched verb in the TS-Text MAP in its original form.

For example, suppose that the following declarative sentence "John traveled to Boston last week" and its interrogative sentence about location "Where did John travel to last week" are in the training dataset. They have the following tag sets before merging:

"John [ARG0/NNP/PER/] traveled [V/VBD//] to [ARG1/IN//] Boston [ARG1/NNP/LOC/] last

[TMP/NN//] week [TMP/NN//]", and "Where [///where] did [V/VBD//] John [ARG0/NNP/PER/] travel [V/VB//] to [ARG1/IN//] last [TMP/NN//] week [TMP/NN//]".

With proper phrase segmentation we know that "travel to" is a phrasal verb. Thus, after merging, we have "John [ARG0/NNP/PER/] traveled to [V/VBD//] Boston [ARG1/NNP/LOC/] last week [TMP/NN//]", and "Where [///Where] did [V/VBD//] John [ARG0/NNP/PER/] travel to [V/VB//] last week [TMP/NN//]".

After merging, the pair of tag-set sequences (X,Y) is deposited in TSSP-DB, where $X =$ "[ARG0/NNP/PER/] [V/VBD//] [ARG1/NNP/LOC/] [TMP/NN//]", and $Y =$ "[///where] [V/VBD//] [ARG0/NNP/PER/] [V/VB//] [TMP/NN//]".

Suppose that we are given a sentence $s =$ "Mary flew to London last month." Its tag-set sequence X_s is exactly the same as X, with [ARG0/NNP/PER/] for "Mary", [V/VBD//] for "flew to", [ARG1/NNP/LOC/] for "London", and [TMP/NN//] for "last month", which are stored in the TS-Text Map. Thus, $Y_s = Y$. We can see that the tag set of [V/VB//] in Y is not in the TS-Text Map and so cannot be matched. To resolve this, check the POS tag in the ARG0-tag set, which is NNP, indicating a singular noun. The POS tag in the first V-tag set is VBD, indicating past tense. Thus, the correct form of the helping verb is "did". The text for [V/VBD//] is "flew to" in the TS-Text Map. The original form of the verb is "fly". Thus, the second V-tag set is replaced by "fly". This generates the following interrogative sentence: "Where did Mary fly to last month?" The tag set for the answer is $X' - Y'$, which is [ARG1/NNP/LOC/], corresponding to "London" in the TS-Text Map.

5 EVALUATIONS

5.1 Training Dataset

We construct an initial training dataset by composing 112 pairs of declarative sentences and the corresponding interrogative sentences to cover the common tense, participles, voice, modal verbs, and some common phrasal verbs such as "be going to" and "be about to" for the following six interrogative pronouns: *Where, Who, What, When, Why, How many*. A total of 112 tag-set-sequence pairs are learned and deposited in the initial TSSP-DB.

Note that SQuAD (Rajpurkar et al., 2016) is a dataset commonly used for training and evaluating generative methods for QG. However, a certain number of QAPs in SQuAD are formed improperly or lack correct answers. There are also about 20% of questions in the dataset that require paragraph-level information. Thus, SQuAD is unsuitable to train a TSS-Learner model.

5.2 Comparisons with TP3

Recall that TP3 (Zhang et al., 2022) takes a declarative sentence with a specified answer key contained in it and the surrounding declarative sentences as input, TP3 generates QAPs with better qualities then previous methods, but it also generates silly QAPs for certain chunks of text. It would be interesting to know whether TSS-Learner may be used to complement TP3. For this purpose we use the same dataset used to evaluate TP3, and we show that the trained TSS-Learner model with the small initial training dataset can indeed generate adequate QAPs for certain texts that TP3 does poorly. Table 1 depicts some of these results.

5.3 Human Evaluation

Computed metrics such as BLUE (Papineni et al., 2002), ROUGE (Lin, 2004), and Meteor (Lavie and Denkowski, 2009) are commonly used to evaluate summarization and machine translation against benchmark datasets, which have also been used to evaluate the qualities of machine generated QAPs. These metrics, however, do not evaluate grammatical correctness, and so human judgments are needed

We asked three human judges to evaluate the qualities of the QAPs on a shared document based on the following criteria: For questions: Check both grammar and semantics: (1) correct; (2) acceptable (i.e., a minor fix would make it correct); (3) unacceptable. For answers: (1) correct; (2) acceptable; (3) unacceptable. If judges have discrepancies on an item, they resolved it through discussions. In so doing, they jointly produced one final evaluation result for each QAP.

We use the official SAT practice reading tests[3] for evaluating TSS-Learner. These tests provide a large number of different patterns of declarative sentences in the underlying passages. There are a total of eight SAT practice reading tests, where each test consists of five articles and each article consists of around 25 sentences. There are 40 articles and 1,136 sentences. After removing interrogative sentences and other non-declarative sentences, there are 1,025 declarative sentences. Using the initial training dataset of 112 pairs of declarative and interroga-

[3]https://collegereadiness.collegeboard.org/sat/practice/full-length-practice-tests

Table 1: Examples that TSS-Learner does well but TP3 does not, where Q-by-TP3 and Q-by-TSSL stand for, respectively, "Question generated by TP3" and "Question generated by TSS-Learner". Note that in Example 3, in the input context, Eudora Welty took a couple of friends to a restaurant and an unknown woman came and joined them.

Example 1	
Input text:	"However, on September 12, 1933, physicist Leo Szilard invented the neutron-chain reaction."
Q-by-TP3:	"When did the 11th of September happen?"
Q-by-TSSL:	"When did physicist Leo Szilard invent the neutron-induced nuclear chain reaction?"
Answer:	"on September 12, 1933"

Example 2	
Input text:	"My mother bought the beautiful basket and put it safely in some hiding place I couldn't find."
Q-by-TP3:	"What did my mother say I needed?"
Q-by-TSSL:	"What did my mother buy?"
Answer:	"the beautiful basket"

Example 3	
Input text:	"Without a second thought, the woman joined the Welty party."
Q-by-TP3:	"What party did Eudora Welty join without thinking about it?"
Q-by-TSSL:	"What did the woman join?"
Answer:	"the Welty party"

Table 2: Human evaluation results on the SAT practice reading tests.

	Where	Who	What	When	Why	How	Total
TSSP-DB entry pairs	18	45	23	22	6	8	122
QAPs generated	26	216	466	51	15	22	796
All correct	21	208	458	51	15	20	773
Syntactically acceptable	4	4	3	0	0	2	13
Semantically acceptable	1	2	5	0	0	0	8
Unacceptable	0	2	0	0	0	0	2

tive sentences, TSS-Learner generates a total of 796 QAPs. Table 2 depicts the evaluation summaries with detailed breakdowns in each interrogative pronoun, where "all correct" means that the question in the QAP is both grammatically and semantically correct, conforms to what native speakers would say, and the answer is correct; "syntactically acceptable" means that either the question or the answer has a minor grammar error, and a small effort such as changing, removing, or adding one word will fix the problem; "semantically acceptable" means that either the question or the answer is problematic in semantics, but a minor effort will fix problem; and "unacceptable" means that either the question or the answer is unacceptable grammatically or semantically.

The percentage of generated questions that are both syntactically and semantically correct is 97%. We noticed that there is a strong correlation between the correctness of the questions and their answers. In particular, when a generated question is all correct, its answer is also all correct. When a question is acceptable, its answer may be all correct or acceptable. Only when a question is unacceptable, its answer is also unacceptable.

The 13 syntactically acceptable questions are mostly due to some minor issues in segmenting a complex sentence into simple sentences, where a better handling of sentence segmentation is expected to correct these issues. Two questions whose interrogative pronoun should be "how much" are mistakenly using "how many". Further refinement of POS tagging that distinguish uncountable nouns from countable nouns would solve this problem. The eight semantically acceptable questions are all due to NER tags that cannot distinguish between persons, location, and things. Further refinement of NER tagging will solve this problem. The two unacceptable questions are due to serious errors induced when segmenting complex sentences. This suggests that we may need to find a better way to deal with complex sentences. Using a recursive list to represent complex sentences might be useful in this direction.

There are 589 sentences for which no matched tag-set sequences are found from the initial training dataset. By learning new tag-set sequences from user inputs, 535 of these sentences found perfect matching, which generate QAPs that are both syntactically and semantically correct. For the remaining 84 sentences,

it is hard to segment them into a set of simple sentences and so no appropriate tag-set sequences were learned. This suggests that we should look into better sentence segmentation methods or explore tag-set trees as recursive lists of tag-set sequences to represent complex sentences as a whole for future studies.

5.4 Efficiency

Finally, we evaluate the running time for our trained TSS-Learner to generate QAPs over 100 sentences on a desktop computer with an Intel Core I5 2.6 Ghz CPU and 16 GB RAM. The average running time is 0.55 seconds for each input sentence, which is deemed satisfactory for online applications. For a given article, assuming that it would take the reader several minutes to read. By then all the QAPs for MCQs would have been generated.

6 CONCLUSIONS

Tag-set sequence learning is a novel attempt for generating adequate QAPs. It can generate adequate QAPs that text-to-text transformers fail to generate. Numerical analysis shows that this approach is promising. It achieves satisfactory results for the English language using existing NLP tools on SRL, POS, and NER tagging. Further improvement of NER tagging may be able to eliminate a small number of semantic errors we encountered.

Tag-set sequence learning is a text-to-rule learning mechanism and a text-to-text model via explicit rules. These explicit rules are automatically learned with moderate human involvement – users are expected to write and provide to the system an interrogative sentence when a declarative sentence has a unseen tag-set sequence. This procedure continues to enrich the collection of tag-set sequence pairs in TSSP-DB, When almost all possible patterns of declarative sentences and the corresponding interrogative sentences are learned (there are only finitely many of them to be learned), TSS-Learner is expected to perform well on generating adequate QAPs from declarative sentences that can be segmented appropriately into simple sentences.

However, not all complex sentences can be segmented using existing tools. In particular, about 7.4% of the declarative sentences in the official SAT practice reading tests are in this category. This calls for, as mentioned near the end of Section 5, a better NLP method to dissect complex sentences.

Applying TSS-Learner to a logographic language would require robust and accurate segmentation at all levels of words, phrases, and sentences, semantic labeling, POS tagging, and named-entity recognition for the underlying languages. It would also require appropriate localization for merging tag sets. It is interesting to explore how well TSS-Learner performs on a language other than English. It is also interesting to investigate whether other NLP tools that better represent the structures of sentences, such as dependency trees and constituency trees, can be combined with tag-set sequences to generate QAPs with a higher quality.

REFERENCES

Ali, H., Chali, Y., and Hasan, S. A. (2010). Automation of question generation from sentences. In *Proceedings of QG2010: The Third Workshop on Question Generation*, pages 58–67.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bonial, C., Hwang, J., Bonn, J., Conger, K., Babko-Malaya, O., and Palmer, M. (2012). English propbank annotation guidelines. *Center for Computational Language and Education Research Institute of Cognitive Science University of Colorado at Boulder*, 48.

Cai, Z., Rus, V., Kim, H.-J. J., Susarla, S. C., Karnam, P., and Graesser, A. C. (2006). Nlgml: a markup language for question generation. In *E-Learn: World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*, pages 2747–2752. Association for the Advancement of Computing in Education (AACE).

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

Chen, W. (2009). Aist, g., mostow, j.: Generating questions automatically from informational text. In *Proceedings of the 2nd Workshop on Question Generation (AIED 2009)*, pages 17–24.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Danon, G. and Last, M. (2017). A syntactic approach to domain-specific automatic question generation. *arXiv preprint arXiv:1712.09827*.

Du, X., Shao, J., and Cardie, C. (2017). Learning to ask: neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1342–1352, Vancouver, Canada. Association for Computational Linguistics.

Duan, N., Tang, D., Chen, P., and Zhou, M. (2017). Question generation for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 866–874, Copenhagen, Denmark. Association for Computational Linguistics.

Flor, M. and Riordan, B. (2018). A semantic role-based approach to open-domain automatic question generation. In *Proceedings of the thirteenth workshop on innovative use of NLP for building educational applications*, pages 254–263.

Harrison, V. and Walker, M. (2018). Neural generation of diverse questions using answer focus, contextual and linguistic features. Association for Computational Linguistics.

Khullar, P., Rachna, K., Hase, M., and Shrivastava, M. (2018). Automatic question generation using relative pronouns and adverbs. In *Proceedings of ACL 2018, Student Research Workshop*, pages 153–158.

Kim, Y., Lee, H., Shin, J., and Jung, K. (2019). Improving neural question generation using answer separation. volume 33, page 6602–6609. Association for the Advancement of Artificial Intelligence (AAAI).

Lavie, A. and Denkowski, M. J. (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. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Lindberg, D., Popowich, F., Nesbit, J., and Winne, P. (2013). Generating natural language questions to support learning on-line. In *Proceedings of the 14th European Workshop on Natural Language Generation*, pages 105–114.

Lindberg, D. L. (2013). *Automatic question generation from text for self-directed learning*. PhD thesis, Applied Sciences: School of Computing Science.

Luong, T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

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

Marcus, M., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of english: The penn treebank.

Martha, P., Dan, G., and Paul, K. (2005). The proposition bank: a corpus annotated with semantic roles. *Computational Linguistics Journal*, 31(1):10–1162.

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

Mostow, J. and Chen, W. (2009). Generating instruction automatically for the reading strategy of self-questioning. In *AIED*, pages 465–472.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL'02, page 311–318, USA. Association for Computational Linguistics.

Peters, M. E., Ammar, W., Bhagavatula, C., and Power, R. (2017). Semi-supervised sequence tagging with bidirectional language models. *arXiv preprint arXiv:1705.00108*.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392. Association for Computational Linguistics.

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

Sachan, M. and Xing, E. (2018). Self-training for jointly learning to ask and nnswer questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 629–640, New Orleans, Louisiana. Association for Computational Linguistics.

Sak, H., Senior, A. W., and Beaufays, F. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Proccedings of the 15th Annual Conference of the International Speech Communication Association*.

Shi, P. and Lin, J. (2019). Simple BERT models for relation extraction and semantic role labeling. *arXiv preprint arXiv:1904.05255*.

Song, L., Wang, Z., Hamza, W., Zhang, Y., and Gildea, D. (2018). Leveraging context information for natural question generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 569–574, New Orleans, Louisiana. Association for Computational Linguistics.

Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 conference of the North American chapter of the association for computational linguistics on human language technology-volume 1*, pages 173–180. Association for Computational Linguistics.

Ukkonen, E. (1985). Algorithms for approximate string matching. *Information and control*, 64(1-3):100–118.

Varga, A. and Ha, L. A. (2010). Wlv: a question generation system for the QGSTEC 2010 task b. *Boyer & Piwek (2010)*, pages 80–83.

Wolfe, J. H. (1976). Automatic question generation from text-an aid to independent study. In *Proceedings of the ACM SIGCSE-SIGCUE technical symposium on Computer science and education*, pages 104–112.

Wyse, B. and Piwek, P. (2009). Generating questions from openlearn study units.

Zhang, C., Zhang, H., Sun, Y., and Wang, J. (2022). Transformer generation of question-answer pairs with preprocessing and postprocessing pipelines. In *The 22th ACM Symposium on Document Engineering (DocEng)*. ACM.

Zhang, H., Zhou, Y., and Wang, J. (2021). Contextual networks and unsupervised ranking of sentences. In *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1126–1131. IEEE.

Zhao, Y., Ni, X., Ding, Y., and Ke, Q. (2018). Paragraph-level neural question generation with maxout pointer and gated self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3901–3910, Brussels, Belgium. Association for Computational Linguistics.

Zhou, Q., Yang, N., Wei, F., Tan, C., Bao, H., and Zhou, M. (2018). Neural question generation from text: a preliminary study. In Huang, X., Jiang, J., Zhao, D., Feng, Y., and Hong, Y., editors, *Natural Language Processing and Chinese Computing*, pages 662–671, Cham. Springer International Publishing.