Question Generation using Part of Speech Information

Jacob Zerr, Texas A&M University

***Abstract*—When testing students on knowledge from a story or article, a human must interpret the text to generate English questions. The difficulty in automating this process is producing a computational algorithm that can fully account for the syntactic and semantic complexities of human languages. Most approaches use big, costly semantic tools such as WordNets to achieve their semantic accuracy and rule-based approaches to achieve their syntactic accuracy. We propose an approach for generating knowledge-testing questions from textual English using machine learning to use part of speech pattern matching without using any large semantic tools.**

I. Introduction

Many attempts have been made to automate interpreting natural human languages, most of which have taken some small sub-problem and attempted to solve it. One such sub-problem is manipulating sentences to create question-answer pairs from a sentence, which we will be addressing. The main difficulty of question generation is that the method must maintain both semantic and syntactic accuracy. When formatting a question, we will need to change the structure of the sentence, add and remove words, change the tense or part of speech of words, or other complex operations. Moreover, through these operations we must keep the semantic integrity of the statement and select the correct answer to the resultant question. However, the applications of a proficient question generator could span domains from automated education tools to better AI conversation generation. We propose a new method of question generation that uses part of speech (POS) pattern matching based off of Inversion Transduction Grammars (ITG) from a sentence and question-answer pair corpus. We restrict our input to sentences containing one independent clause with the thought that this approach would work on any input if compressed first.

II. Problem Definition

Our input will be any collection of English sentences containing one independent clause. The sentences should be well formed and in correct English grammar for best results. The output will be a set of question-answer pairs that should be asking about the contextual knowledge of the original text. The output questions should also be grammatically correct. Here are a couple examples.

- John drove the car to work. $\rightarrow$ Who drove the car to work? John
- The pump is now operational. $\rightarrow$ Is the pump operational? Yes
- He waters the garden every day. $\rightarrow$ What does he do every day? waters the garden

III. Related Work

Question generation was brought to the attention of the natural language processing community by Wolfe [4] in 1976. He outlined the purpose and applications of a question generator and the potential challenges. Since then, many have produced question generators of a limited focus. Papasalouros [1] creates only multiple choice questions by producing a set of similar sentences where a key word has been replaced in the wrong selections. This reduces the complexity of the problem by avoiding interrogative sentence structure. Brown [3] focuses only on questions that test vocabulary and uses a WordNet to increase their question complexity without losing semantic accuracy. They also use part of speech (POS) tagging to maintain the syntactic accuracy of the question. Kunichika [2] provides the most general approach of all by dissecting both the syntactic and semantic structure of the original sentence before producing the question. After looking at both of these, their algorithm has a broad spectrum of questions it can generate about the original declarative sentence. However, this approach relies heavily on the accuracy of the interpretation of the sentence using tools like WordNets that may not be accurate in all cases. These are three representations of the current best solutions, none of which use machine learning. Our approach will rely heavily on a POS tagger for which we will be using the Stanford Parser outlined in Toutanove [9]. On a different note, Heilman [5] ranks generated questions which may be considered as a useful addition to our question generation process later on in our development.

IV. Inversion Transduction Grammars

Inversion Transduction Grammars are grammars that map two languages simultaneously and generally follow the format of a context free grammar. The main difference is the angle brackets in the grammar denote that the symbols should be read in left-to-right order for the first language and right-to-left for the second. This allows for the grammar to successfully map two languages with different part of speech orderings like SOV, SVO, or VSO languages. From there the lexicon has word pairs, one from each of the two languages, that should be direct translations of each other. This method uses basic

Jacob Zerr is participating in a National Science Foundation REU at the University of Colorado at Colorado Springs, Colorado Springs, CO 80918. email: zerr2@tamu.edu

word-to-word translations and the fact that most languages use similar part of speech models, just in a different order, to achieve an accurate machine translation. Wu [6] explains these grammars in detail and shows how they can be used as an accurate form of machine translation. Both Goto [7] and Neubig [8] use these techniques to successfully perform machine translations between complex languages.

V. Our Approach

A. *Producing POS Pattern Templates from the Corpus*

The main approach that we will be pursuing to convert our declarative sentences to questions is through a POS pattern matching approach based off of ITGs. Though ITGs have mainly been used to convert a parsed sentence into another language, we will be using it to convert between declarative English and interrogative English. The difficulty in this process is that ITGs rely on the structure of the two sentences to be similar in all but ordering. However, there are structural parts of interrogative English that are not in declarative English and vice verse. Our approach will avoid this by ignoring the tree structure of the grammar and just map the movement of different phrases from the sentence to the question.

The first major step of processing our corpus instances is to identify the phrases that stay consistent in the transition from declarative sentence to question-answer pair. We do this by searching the instance for phrases of the exact same wording starting from the largest possible phrases and then incrementally decreasing the size until all of the common phrases have been identified. We call this process chunking. Figure 1 shows such an instance and the phrases that have been identified after chunking has been completed.

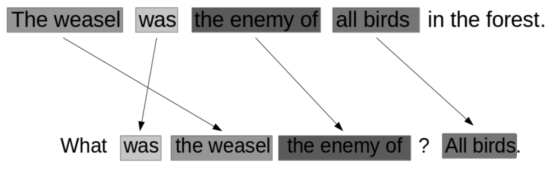

Figure 1. Sample of chunking the common phrases from an instance in our corpus.

Notice that there may be phrases in the sentence, question, or answer that are not in any of the other parts of the instance; in this case *in the forest* and *What*. These are kept and used by the algorithm in the process of finalizing the template; this will be explained later.

Our algorithm also can identify phrases that appear in all three parts of the instance as a part of the chunking process. This helps create templates for questions that quiz on adjectives of the sentence while still maintaining accuracy. Figure 2 is an example of such an instance where *plate* is repeated in all three parts of the instance to ensure that the answer makes sense.

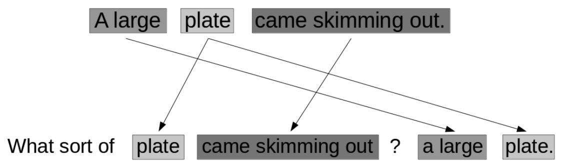

Figure 2. Sample of chunking an instance where a phrase is repeated in all three parts of an instance. This helps produce templates of questions that quiz on adjectives in the input sentence while still keeping accuracy.

Now that we have chunked our instances, we need to determine the part of speech of each of the phrases included in the sentence portion of the instance. For this we will be using the Stanford POS tagger [9]. There are two main approaches to attempting to tag these phrases with a POS: parsing it within the original context of the sentence or parsing it out of context. When parsing it out of context, we can conveniently get a single POS for the phrase. However, you forfeit accuracy with this method because the Stanford Parser solves ambiguities internally and it may return the wrong POS in an ambiguous case. For this reason, we chose to parse the phrase within the context of the original sentence. However, this is slightly more difficult, because we will now have to search the grammar parse tree of the whole sentence produced by the Stanford Parser. Our method for this was to search for the node of the tree that was the deepest ancestor of all of the words in the phrase. An example of this is shown in Figure 3. Here we can see that we identify the POS for *the enemy of* as a Noun Phrase in the context of the sentence *The weasel was the enemy of all birds in the forest*.

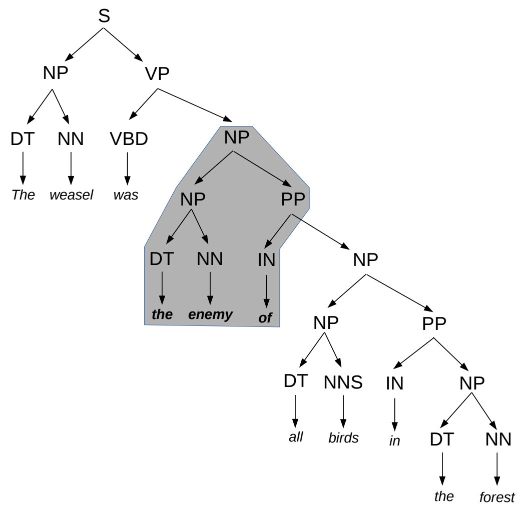

Figure 3. Our method for finding the POS of a phrase involves finding the deepest common ancestor of the words of the phrase. Here we can see *the enemy of* is being labeled as a Noun Phrase.

With this method of POS tagging, we then will label every phrase in the original sentence. This includes any phrases that were not repeated in the question or the answer. The result of this process from the example used in Figure 1 is shown in Figure 4.

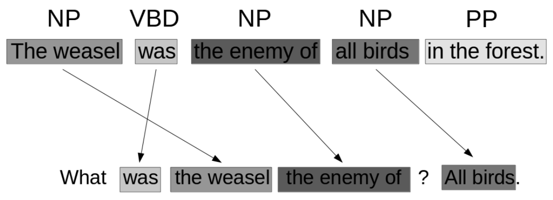

Figure 4. The example from Figure 1 with its sentence chunks labeled with their POS.

After the sentence chunks have been labeled, we drop all of the phrases that appeared in the sentence part of the instance. The phrases that only appeared in the question or answer are left as a part of the template. This is the final step of producing our POS template from an instance in our corpus. This process is completed for every instance in the corpus before we start trying to use these templates on our input sentences. Figure 5 shows this last step on our example.

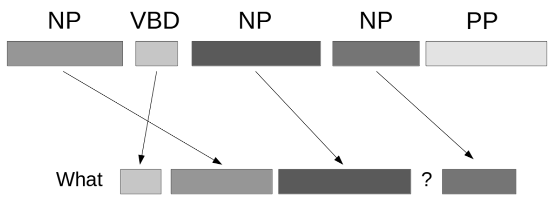

Figure 5. The final step of preparing the templates is drop all of the phrases that appeared in the sentence. Phrases that were just in the question or answer remain.

B. Generating Questions

Once we have converted the instances of our corpus into POS pattern matching templates, we can begin to try to fit input sentences into our templates. We do this by simply seeing if the input sentence can be divided into phrases that, when tagged with a POS, match the template. If we do find a match, we reorder the phrases by using the template's question-answer ordering to produce our question-answer pair. An example of a sentence fitting the template we produced above is in Figure 6.

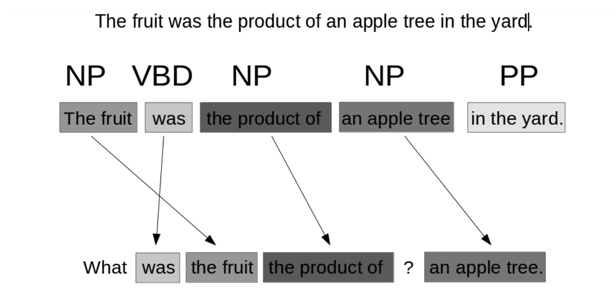

Figure 6. A sentence being matched to our example template and the question-answer pair it produced.

An interesting question that arose from this pattern matching method is which type of POS tagging we would use for this part of the algorithm, in-context or out-of-context. Initially, it seemed clear that we should follow the same method we did in producing our corpus and use in-context. However, when experimenting with out-of-context we sometimes would produce a wrongful tag to a phrase that would fit a template. The expectation was that from an erroneous matching we would produce an inaccurate question, but this was not always the case. Figure 7 shows an input sentence matching to the template we produced above with *render* erroneously being labeled a Noun Phrase, however the produced question is accurate. We explored this question and our answer is discussed later in the results section.

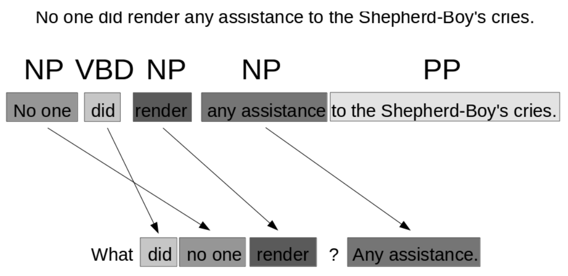
No one did render any assistance to the Shepherd-Boy's cries.
NP
VBD
NP
NP
PP
No one
did
render
any assistance
to the Shepherd-Boy's cries.
What
did
no one
render
?
Any assistance.

Figure 7. A sentence being wrongfully tagged and matched to our example template may still produce an accurate question-answer pair.

As a last note, if our algorithm can divide an input sentence to match a template more than one way, then it will produce a different question for each different legitimate divisions. An example of this is shown below in Figure 8.

An interesting observation on our method is that because we are simply reordering phrases, we keep the same vernacular of the original sentence. We have been operating in the domain of children's stories for this project and often times children stories will have odd wording that is not common vernacular anymore. These odd phrases will always be reflected in our output. The example in Figure 7 uses phrases like *render assistance* whereas most people would simply say *help*. This can be both a good and bad attribute of our approach. The good part is that our questions may contain slang or improper words of spoken English that takes our questions to a semantic level not normally achievable by a computer. However, it also can sometimes cause problems if these words are wrongfully

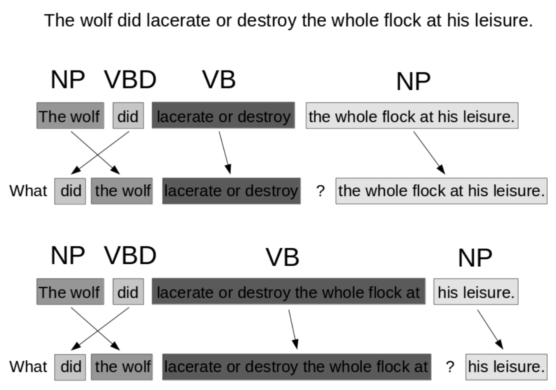
The wolf did lacerate or destroy the whole flock at his leisure.
NP VBD VB NP
The wolf did lacerate or destroy the whole flock at his leisure.
What did the wolf lacerate or destroy ? the whole flock at his leisure.
NP VBD VB NP
The wolf did lacerate or destroy the whole flock at his leisure.
What did the wolf lacerate or destroy the whole flock at ? his leisure.

Figure 8. If a sentence can be divided in more than one way to match a template then the algorithm will produce a different question for each way.

tagged and will miss the factual tone of SAT-style questions that a user may want because of odd diction in the original sentence.

Also, it is important to note that we restricted our input to sentences containing one independent clause. This is necessary, because with additional clauses the accuracy of our POS tagging for phrases goes drastically down. For instance, if we divide our input in a way that a phrase is an entire clause it may be be given a POS label of Sentence, which is too general for our templates to produce good results. Typically, the larger the phrases become, the higher up the parse tree you will have to go for a deepest common ancestor, and the less specificity of POS tags we will have. Thus, in order to maintain reasonable levels of accuracy, we must limit our input to one clause sentences.

VI. Datasets

A large part of our work was producing and manipulating the corpus that defined our POS matching templates. This corpus is a collection of instances that map a sentence to a question-answer pair. Initially our corpus had 254 instances. From initial testing using this corpus we observed several things; a small corpus could produce an ample number of questions even with just one sentence inputted, often the same questions were produced more than once, and some instances in our corpus were better at producing accurate sentences than others. From these observations we decided to stop expanding the corpus and to actually start eliminating some instances.

Firstly, we had learned that some of the instances in our corpus were producing the same templates. Thus we went through and found the instances producing the duplicate templates and deleted them. An example of this was the template below had been produced 34 times. After 93 deleting instances that were producing duplicate templates our corpus had been reduced down to 161 instances.

$$\textbf{NP} \quad \textbf{VP} \quad \rightarrow \quad Who \quad \textbf{VP} \quad ? \quad \textbf{NP}$$

Secondly, we observed that some templates produced by our instances were much better at producing successful question-answer pairs than others. To test this theory we ran our corpus against some preliminary testing examples and confirmed this. Some templates were producing many consistently accurate questions, some produced very few questions, and others produced many inaccurate questions. Based on these results we eliminated any instance from our corpus that was producing questions at a twenty percent accuracy level or worse. This reduced our corpus down to just 129 instances.

Contrary to most forms of corpus-based machine learning, we found this corpus to be more than enough to produce a high number of different questions and a wide breadth of different types of questions. This is one of the largest advantages of our approach; it takes a comparatively tiny amount of data to get good results especially when compared to most of the other approaches that use large WordNets or other large semantic tools.

VII. Results

We analyzed and made improvements based off the syntactic and semantic accuracy of the output of our approach. The proportion of output instances that are grammatically correct, accurately quizzes the reader on the original knowledge, and has the corresponding answer will be our main metric of success. We used unbiased volunteer evaluators that judged each produced question-answer pair on whether they were syntactic and semantic accurate or not. Our evaluators are native English speakers that are in the process of attaining a Bachelors Degree, thus they have a firm knowledge of the English language. Our input were single independent clause sentences from children's stories such as *The Princess and the Pea*, *The Boy Who Cried Wolf*, and other such children stories.

A. POS Tagging Methods

We would first like to address the question we presented earlier on whether POS tagging on our input sentences should be done in-context or out-of-context. We would first like to note that we used only in-context tagging for creating our templates so that we could create accurately tagged templates. However, as we noted before, we produced accurate questions using both methods when tagging the input sentences. Based off of this we decided to experiment using four different methods for determining the POS to tag the sentence phrases: using the in-context tag (IC), using the out-of-context tag (OC), using a POS tag only if the two methods agreed ($IC \ \&\& \ OC$), and using either method to try to fit a sentence into a template ($IC \ || \ OC$). We tried these four methods on a 20 sentence input children's story.

Based off of the above results we chose to use the $IC \ || \ OC$ method for the rest of our work because of the greatly increased total solution production despite a very similar accuracy rate. Based off of the numbers above, this method was producing, on average, 7.25 accurate questions per inputted sentence.

Table I
POS TAGGING METHODS

Method	Accurate	Total	Percent
IC	94	160	58.75
OC	87	150	58.00
IC && OC	58	90	60.00
IC || OC	145	243	59.67

B. Overall Accuracy

For our final accuracy test, we used an input 48 sentences long from children's stories. We produced an output of 435 question-answer pairs. This means that we averaged 9.06 question-answer pairs per inputted sentence. This puts into perspective that our corpus, at 129 instances, really can perform like a large semantic tool despite its small size. The produced question-answer pairs were assessed by 4 evaluators that ranged the accuracy from 57.01% to 59.67% with an average of 58.36%. This result is also encouraging considering the previous work in this area. Brown [3] produced an accuracy rate from 52.86% to 64.52% and Papasalourous [1] produced an accuracy of 75% from his best strategy, but averaged an accuracy of 47.55% between all of their strategies. It is also interesting to note that both of these approaches were slightly more restricted in domain than our approach and they both relied on advanced wordnets in order to maintain semantic accuracy.

VIII. POSSIBLE FUTURE WORK

A possible extension of this work would be automatically analyzing the questions produced and ranking them in some way. Depending on the accuracy of the rankings we may be able to achieve a higher accuracy of the questions that are ranked in some top fraction of the produced questions.

IX. CONCLUSION

By using a relatively simple machine learning method with a small dataset, we were able to out-perform previous rule-based methods that used large semantic tools. If used with an accurate sentence compressor, we believe this method for generating questions would be extremely accurate and convenient. Our approach is also not domain-specific and thus can be used in anything from automated education tools to better AI conversation generation.

REFERENCES

[1] A. Papasalouros, K. Kanaris, and K. Kotis. "Automatic Generation Of Multiple Choice Questions From Domain Ontologies." In *e-Learning, pp. 427-434. 2008.*

[2] H. Kunichika, T. Katayama, T. Hirashima, and A. Takeuchi. "Automated question generation methods for intelligent English learning systems and its evaluation." In *Proceedings of International Conference of Computers in Education 2004, pp. 2-5, Hong Kong, China, 2003.*

[3] J. Brown, G. Frishkoff, and M. Eskenazi. "Automatic question generation for vocabulary assessment." In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing, pp. 819-826, Vancouver, Canada, Association for Computational Linguistics, 2005.*

[4] J. Wolfe "Automatic question generation from text-an aid to independent study." In *ACM SIGCUE Outlook, vol. 10, no. SI, pp. 104-112, ACM, 1976.*

[5] M. Heilman, and N. Smith. "Good question! statistical ranking for question generation." In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pp. 609-617, Los Angeles, USA, Association for Computational Linguistics, 2010.*

[6] D. Wu. "Stochastic inversion transduction grammars and bilingual parsing of parallel corpora." In *Computational Linguistics 23, pp 377-403, 1997.*

[7] I. Goto, M. Utiyama, and E. Sumita. "Post-Ordering by Parsing with ITG for Japanese-English Statistical Machine Translation." In *ACM Transactions on Asian Language Information Processing (TALIP) 12, no. 4, 2013.*

[8] G. Neubig, T. Watanabe, S. Mori, and T. Kawahara. "Substring-based machine translation." In *Machine Translation 27, no. 2, pp 139-166, 2013.*

[9] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. "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, pp 173-180, Edmonton, Canada 2003.*