Automatic Quiz Generation

Brandon Cuadrado, 109237297
Alex Scarlatos, 109031143
Eugenia Soroka, 111494044
Department of Computer Science
Stony Brook University, Stony Brook NY 11790 USA
brandon.cuadrado@stonybrook.edu, alexander.scarlatos@stonybrook.edu,
ysoroka@cs.stonybrook.edu

Abstract

This application addresses the NLP task of automatic quiz generation from passages of text. For our system, we have implemented two separate question generation techniques, one based on the dependency parses of sentences and the other one based on Semantic Role Labeling, and have compared the two. Additionally, we incorporated an Answer Scoring technique to assess responses provided by a user and evaluate their performance on free response questions. As a culmination of these techniques, we successfully built an end-to-end quiz generation system and analyzed the differences in performance between the various approaches.

1 Introduction

Our task of automatic question generation from text operates ideally on expository passages. We developed these algorithms based on the content from textbooks in the Humanities, as well as user input to the questions provided, ultimately scoring them accordingly. The input textbooks are at the educational college-level, or a select passage from it, as a text file, as well as the user-provided answer to the resulting questions. The output is a text file of question and answer pairs generated for the passage, as well as grades on the user-provided answers.

The motivation for our work is to explore the potential of Automatic Question Generation in online learning applications that lack a formal instructor, or by students who want to review for exams based on textbook content. The benefit of such unsupervised systems is to provide a method of accurate quiz generation in preparation for an exam, or to grade electronic exams in an automatic or supervised-automatic pipeline. We strive to automate the evaluation of students' reading comprehension as an alternative to performing the task manually. The two best applications for our system would be for use in e-learning applications, as well as a method of automatic dataset generation for use in large scale question answering training models.

2 Related Work

Although automatically generated questions have become an actively developing topic in application-oriented NLP research, it does not have a long history, and most of the approaches rely on the syntactic structure of a sentence (Afzal et al. 2011). This problem is challenging since, to start with, making automated systems generate text (not only semantically and syntactically correct, but also related to the topic at hand) is challenging. Additionally, training such systems can be hard, since usually questions are stored in a form of text, while many automatic question generation methods rely on usage of templates, i.e. specific structures, which means we must convert all the questions present in the training data set into a certain pre-defined structure.

One of the most popular question generation approaches involves parsing text with a PSG (phrase structure grammar) parser and then forming questions using templates (Mazidi, Tarau 2016). Another technique found to be useful is applying a neural network to the Freebase knowledge base to transduce facts to questions.

One more possibility is to use unsupervised information extraction methods with the purpose of discovering the most significant concepts and relations in the domain texts, without any prior knowledge of their types or their exemplar instances (Mazidi, Tarau 2016), with the purpose of then forming questions. In another study describing Multiple Choice Question generation, the provided system selects the informative sentence and the keyword to be asked based on the semantic labels and named entities that exist in the sentence, the distractors are chosen based on a similarity measure between sentences in the data set.

As the baseline system, we have selected the system implemented in (Mazidi, Tarau 2016), which we use through their demo available online in a web interface. The main objective of that system is to use a novel algorithm (called "DeconStructure") to reveal the pattern of constituent arrangements, to use it for determining what types of questions should be asked about the sentence; as well as to rank question importance, an often-overlooked criterion. The key idea of this algorithm is to:

1. Deconstruct a sentence with SRL and dependency parses (while gathering POS, word lemmas and NE information along the way)
2. Structure it by assigning a corresponding functional label to clause components, identified by both parsers
3. Compare structured sentence object to ~60 pre-defined patterns; if there is a match, then generate a question and rank question importance

We compare the quality of questions generated by our own system to the baseline as one of the means by which we evaluate our system.

3 Quiz Generation System

3.1 System Overview

Our quiz generator system uses textbooks for source sentences. These textbooks were retrieved from the 2012 Book Archive[1], and we selected books on psychology, public speaking and law to provide variety, which is important for the ranking algorithm mentioned below. We converted them from HTML to a readable format containing only content sentences using a Python library called PyQuery[2].

Using Python library spaCy[3], we process each sentence in each "passage" (paragraph from textbook source). Spacy extracts POS tags and a dependency tree structure for each sentence. We explored two different techniques for generating questions from sentences: the first described in the study by Afzal et al., which we will refer to as "Dependency-based" question generation, and the second described in the study by Mazidi and Tarau, which we will refer to as "SRL-based" question generation. Each technique extracts a high-level structure from a sentence, which can be converted into question and answer text by matching the structure with a question generation template.

Our quiz software takes all questions generated from a given set of passages and feeds them to a user one at a time. Users answers are scored from 0-1 based on their similarity with the sample answers generated by the question generation systems.

3.2 Dependency-based Question Generation

This method is mostly based on the one described in the research conducted by Afzal et al. For each sentence, we first extract all "chains", combine all matchings chain pairs into "patterns", and rank the patterns using a statistical method. The paper does not mention how to convert these patterns into questions, so we used a custom method that matches patterns with regular expression-like templates to produce question and answer text.

A chain is defined as a named entity and its dependency path to its root verb. The motivation behind using this kind of structure is that named entities will often be the objects of interest for questions. However, we found spaCy's named entity recognition (NER) to be very sparse, and to not identify most objects that we would like to include in sentences. Instead of using named entities as suggested by the original paper, we used all nouns and adjectives. In addition, we did not include chains that were subsets of others, in order to reduce question redundancy and only select the most descriptive structures from a sentence. When all valid chains are extracted, patterns can be

[1] https://2012books.lardbucket.org/
[2] https://pythonhosted.org/pyquery/
[3] https://spacy.io/

Template: *"[ROOT][nsubj,amod][dobj,amod]": ("{2} {1} {0} what?", "{4} {3}", [0, 1, 3])*
Pattern: [have][corporations,Many][cultures,distinct]
Result:
Question: Many corporations have what?
Answer: distinct cultures
Relevant words: verb = have, subj = corporations, obj = cultures

Figure 1: Example template and output from Dependency-based Generation model

formed by merging pairs of chains with the same root verb. A pattern will then result in 3 parts:

1. **Stem.** The part of the chains at the head that is identical
2. **Left Branch.** Left chain diverging from the stem.
3. **Right Branch.** Right chain diverging from the stem.

Patterns are ranked to help determine which patterns contain unique, valid information and to distinguish important patterns from those that contain common knowledge. This is done by comparing the statistical uniqueness of a pattern relative to other patterns in a general corpus. The paper uses a variety of statistical methods, but finds that Chi-Square was the superior method, so we implemented Chi-Square as well. To rank the uniqueness of a pattern in a given "domain" corpus, one compares the number of times that pattern was seen in the domain corpus to the number of times one would expect to see it, by finding its frequency in a "general" corpus. The greater the difference is, the higher the pattern is ranked.

However, since many patterns are completely unique, this statistical method would not produce valid results without an enormous amount of data in the general corpus. As such, we implemented a modified method. When a pattern is matched to a template (this process is described below), common words are extracted from the pattern, such as the verb, subject and object. Instead of finding the statistical significance of an entire pattern, we find the statistical significance of these selected words individually, and construct a uniqueness vector for a pattern, which is summed to find a total relative uniqueness. For domain corpus, we use the passage that the sentence is currently in. And for general corpus, we use all sentences from the textbooks that the sentence is not in.

Our question generation technique takes a pattern and compares the stem and branches to predefined rules, which will match and extract words in a regex-like manner. Each rule examines the current word's dependency role in the sentence, and if it matches then moves onto the next word and rule. Rules can also have special properties, such as "any", which will match any word, "or", which will match with a set of dependency roles, and "*", which will match 0 or more words of any dependency type. While more complex rules could have been written, we found these to be sufficient for matching. The templates are formatted like a dictionary, with the matching pattern as the key, and the value being a tuple formatted as the following structure:

$$(question\_text, answer\_text, relevant\_words)$$

Figure 2: Dependency-based Generation model output format

3.3 SRL-based Question Generation

The concept of using SRL parsing to generate questions was based from the methods described in Mazidi et al.'s study (Mazidi, Tarau 2016). Derived from the DeconStructure Algorithm detailed in their study, our version of an SRL-based Question Generation system uses a combination of SRL arguments and dependency roles to match and generate questions from a list of predefined question templates. To achieve this task, rather than operate on entire sentences, we use the SRL parser provided by practNLPTools[4] to separate a sentence into independent clauses. Within these clauses, the SRL parser assigns semantic roles to the various components of the sentence. For the purpose of this project, we will use the term phrase to refer to an independent clause and use the term phrase entry to refer to a portion of the text assigned a semantic role.

[4] https://github.com/biplab-iitb/practNLPTools

constituent	text	arg	head	dependency role
agent	I	A0	live	nsubj
location	in New York	AM-LOC	live	prep
verb	live	V	live	ROOT

Figure 3: Example representation of a parsed phrase using SRL-based Generation system

Each phrase is assigned a predicate, which is the verb or action describing the independent clause. Within each predicate symbol, there are various phrase entries assigned a semantic role. A phrase entry with the semantic role of A0 through A5 will indicate an agent or other object being acted upon in the sentence. The semantic role of an adjunct represents a special argument that is optional to describe the situation surrounding a verb. Adjuncts are very helpful in semantically understanding a sentence because they can provide specific information such as if the sentence indicates location details (AM-LOC), temporal details (AM-TMP), causal relationships (AM-CAU), and other information. References are represented in the form of R- followed by another argument. This represents a phase entry that references another argument in this sentence or another.

When parsing this SRL information from a phrase, we also take dependency roles into account. In order to use dependency roles in the structure of entire phrase entries, spaCy is used to extract dependency roles of the head of a phrase entry and use the role to describe the phrase entry. This allows dependency parse information to be considered in the algorithm while treating each phrase entry as a single semantic unit. A list of phrase entries are extracted to form one phrase, whose pattern can now be represented as a combination of semantic roles with dependency roles attached.

With the phrase pattern established, the phrase can now be classified into its appropriate question templates. Phrases are able to match multiple templates, as multiple questions can be gathered from a single independent clause. To define a template, we define combinations of SRL arguments and dependency roles that must be met for each template requirement. For example, the sentence "I live in New York" would satisfy a template that requires a verb "V" argument, an agent "A0" argument with a dependency role of "nsubj", and an AM-LOC argument. This template defines the combination of SRL arguments and dependency roles needed to fulfill its requirements. It would be defined using the string "Where does {A0;nsubj} {V;_any}" to represent the question and the string "{AM-LOC;_any}" to represent the appropriate answer. Once these requirements are identified for all matching templates, the resulting question and answer pairs are stored and the next phrase is matched in the same process. Using the same example sentence, the question and answer pair resulting from "I live in New York" would be "Where does I live?" with the accepted answer being "in New York".

3.4 Answer Scoring

For the answer scoring system, we first decided to re-create a system similar to the one mentioned in [Keisuke et al. 2015], which combines response-based and reference-based approaches to short answer scoring task, through usage of stacking. And although the features generation part was fairly straightforward, we, however, encountered a problem when it came to setting up a two-layer learning model. Authors of [Keisuke et al. 2015] had access ETS database of nearly 2000 graded answers for each of their questions, which they used for learning, while we didn't have that, since our questions have been automatically generated from passages and we could only construct such a dataset manually.

Instead, we came up with another approach: assess user's answer using the combination of similarity between user's response and a list of acceptable answers, the keywords included (or not) in the response, and partial containment of the response (or some of its words) in the actual, correct answer(s).

Dependency-based scoring
$grade\_dep = w_1 Similarity(resp, ans) + w_2 keywords + w_3 containedIn$
SRL-based scoring
$grade\_srl = w_1 Similarity(resp, ans) + w_2 containedIn$

Figure 4: Scoring calculations for each model

Here, $w_i, i = 1,2,3$ are the weights for each term, $Similarity(resp, ans)$ is the measure of sentence similarity of the user's response and the answer from a list of possible answers, $keywords$ is the measure of how many keywords were included in the response, $containedIn$ is the measure of the percentage of the response contained in the target (acceptable) answers.

For the $Similarity(resp, ans)$, we used sent2vec[5] package introduced and developed by the authors of (Matteo et al. 2018), which is basically a sentence-level analogue of word2vec. For the score, we use the cosine similarity between two vectors.

The $keywords$ argument is not present for the SRL-based system, since SRL parse does not give that information - it is phrase-based instead of word-based, so using $keywords$ there would have been infeasible.

The weights w_i were tuned manually, i.e. we ran experiments and tweaked the parameters so that the scoring system yields good result, but we also skewed it a little towards higher scoress, so that too low grades do not discourage the user. Which had an impact on the results of the evaluation, as will be seen further.

3.5 Quiz Generation

For generating a quiz, we take the list of all questions generated by both dependency-based and SRL-based system to form a quiz that presents the text passage one after another, with all the questions related to each passage in between. We also allow for the user to choose the "hint" option, which outputs the sentence which the question related to, so that the user can find it easily. After the user inputs their answer, the grade is displayed, as well as the list of possible answers to that question. At any time, user has an option of exiting the quiz and seeing their final, total grade.

4 Results

4.1 Evaluation System

Due to the largely qualitative nature of the Question Generation task, most of the evaluation required a manual overview of the generated results on a somewhat small scale. Similar work in this NLP task have used services such as Amazon Mechanical Turk to qualitatively assess large scale model output, however our team was restricted to working on a smaller scale due to the budgeting of man hours required. As such, evaluation also occurred during the development cycle as we refined the algorithms to produce increasingly natural and relevant results.

For evaluation, we used our Dependency-based Generation model and SRL-based Generation model on 1500 passages throughout three textbooks obtained from the 2012 Textbook Archive. Our criteria for a successful question generation hinges on three principles:

1. **Grammaticism.** The generated question and answer pairs should be grammatically correct and ideally indistinguishable from handwritten text.
2. **Passage Relevancy.** The generated question and answer pairs should properly reflect the essence of the associated sentence or passage.
3. **Question/Answer Relevancy.** A generated answer should properly reflect the information queried by its paired question.

In considering these principles, we also looked toward a baseline system as implemented by Karen Mazidi (Mazidi, 2016). Her study on automatically generating questions referenced a web interface which she was kind enough to grant our access, once contacted while researching her work. This demo provided a useful baseline model to compare against our work, though the model only produces questions and not their associated answers. As such, we can compare our first two principles against this baseline model. Nonetheless, it proved valuable in both comparison and the inspiration of new template formats through its example output.

Due to the lack of resources regarding Question Template matching, the templates used in this model have been made specifically for this project. As such, the algorithm matches against a list of templates that is largely incomplete; representing only the basic question formats able to be extracted from a given sentence or passage. Despite the wealth of information that both SRL parsing and Dependency parsing provides, a question generation system relies heavily on its template system to produce meaningful output. As such, this baseline provided valuable support in forming

[5] https://github.com/epfml/sent2vec

meaningful templates to help aid in filling the gaps left after our research into related work.

4.2 Analysis

The SRL-based Question Generation model sees strength in its ability to parse entire portions of a given sentence rather than individual words. This allows, on a sentence-basis, semantic roles in a question template to be fulfilled using either a single word or appropriate phrase. The information gathered from an SRL parse is also very helpful in capturing the specificity of a given passage, thus contributing to the evaluation criteria of passage relevancy. The use of adjunct arguments, such as modal verbs and temporal modifiers, allow a question to be better formatted to fit the essence of a passage. For example, the sentence "John is happy because of the weather", results in the question "Why is John happy?" and the answer "because of the weather," due to the causal adjunct detected. The SRL-based parser can also generate the question "What is John" due to recognizing an adjective A1 SRL argument as well. This level of specificity allows the SRL-based model to satisfy the relevancy evaluation criteria with more consistency than the Dependency-based model.

A noticeable weakness in both models is the breadth of their template definitions. As indicated before, the templates used in each of these models are manually created during the development cycle and refined through evaluation of our three core criteria. However, the implementation schedule necessitated only basic templates to be generated that do not properly represent the full depth of the sentence parsing models that they inherit. As such, variety and specificity of the generated questions are lacking and could potentially be much improved given a template system that fully utilizes its knowledge base.

Due to the limitations in conjugation that both of our models offer, the baseline model consistently offers more grammatical output, satisfying our first evaluative measure. The Dependency-based model lacks in grammatical output, due to its reconstruction of the sentence word-by-word. As such, a question may use words parsed from the given passage regardless of its semantics as presented in the new sentence; potentially resulting in cases of unconjugated verbs or starting words of a sentence being uncapitalized. Similar problems arise in the SRL-based model as well, though all its templates begin with predefined words, thus avoiding the need to worry about uncapitalized output. However, this is a result of their template model more so than the parsing efficacy. The SRL-based model does, however, use large portions of the given sentence due to its use of entire phrases to represent a semantic role. Thus, the output question is reconstructed using entire strings of words, leading to the potential for a more naturally flowing sentence. However, this can still result in issues with unconjugated verbs for a sentence, like in the case of "Where does I live?" as referenced previously. The conjugation in this parsed sentence also fails to recognize entities across sentences, as shown by referring to the subject "I" in this question, rather than linking it to a Named Entity as reference elsewhere in the passage. Notably, however, the baseline model also suffers from this entity recognition oversight, resulting in "I live in what?" as the resulting question for the sentence "I live in New York".

Using the 3 criteria for successful question generation, again described as Grammaticism, Passage Relevancy, and Question/Answer Relevancy, we were able to evaluate the success of each question generation system by judging scores using qualitative measurements. A “good” score denotes a question that follows all three criteria, where a “bad” score denotes one or more of the criteria missing, and a “very bad” score denotes a question with little comprehensive value to the passage or grammatical efficacy.

The Dependency-based generation model produced a result of an approximate 35% success rate, meaning approximately 35% of generated question and answer pairs satisfied the evaluation criteria. Of the approximated 65% of bad results, approximately 23% of bad results were classified as very bad.

For the SRL-based generation system, the resulting questions generated produced a resulting approximate 48% success rate, thus almost half of the output were both grammatical and relevant. Of the approximated 52% of bad results, approximately 15% of bad results were classified as very bad. As such, the SRL-based generation system performed better than the Dependency-based generation system both qualitatively and through quantitative analysis of the three defined success criteria.

For evaluating the answer scoring system, we followed an empirical method, i.e. we ran the quiz of 10 randomly selected questions 20 times and

counted the amount of True Positive, False Positive, True Negative and False Negative examples. The results are shown below.

	Dependency	SRL
Accuracy	89%	85%
Precision	98%	75%
Recall	58%	67%
F1 Score	73%	71%

Figure 5: Statistical results of scoring output

As can be observed for the results, the scoring system was more inclined to make a False Positive mistake rather than False Negative, which is explained by our own parameters, selected to give "nicer" grades to the user. Of course, this can be easily changed to make a stricter grading system.

5 Conclusion

With regards to Question Generation, our models provide two ways of analyzing large sets of passages and generating questions which, though potentially ungrammatical, capture the essence of the passage semantics. There is more work to be done from here in order to create a system that truly captures the meaning of a passage in a way that mimics manual quiz generation. That said, through recognizing the strengths and weaknesses of both models, a model combining SRL-based templating and Dependency-based templating would better achieve this NLP task. The Dependency-based model, though limited in scope due to its word-level mechanics, can use dependency roles at a low level which allows potentially nuanced responses; complementing the broader scope of SRL-based parsing to foster specificity and relevancy in resulting question and answer pairs. This combination of phrase-level and word-level evaluation would benefit a future version of this generation model.

For templating, as expressed before, future work would surely involve expanding the list of templates to utilize the full capabilities of the information extracted from an SRL parse. Generating templates that complement the wide range of arguments, references, and subjects as portrayed through SRL would increase both the frequency of question generation and their level of relevancy to the passage at hand. When generating these templates, conjugation techniques can also be used to ensure that all output questions and answers are internally consistent with their grammar. Thus, strengthening performance in mirroring human-generated quizzes tremendously.

As for the Answer Scoring, we have successfully built a general tool for automatic student evaluation, which yields high accuracy and precision (while somewhat low recall and F1 score). It should be noted, that although this scoring system is general and can be used without an explicit question-generation system (of both approaches), it sometimes provides incorrect grade due to mistakes in parses themselves. Additionally, we have noticed that the similarity metric seems to work not as efficiently with short phrases than with the longer phrases. It would be best to utilize this technique when working with subjects where you must be exact, e.g. in Legal where you must memorize the laws word-by-word, or Medical where you have to know the exact name of the muscle, organ, etc. One possible improvement of this system will be to incorporate a method of machine learning into it, as to better assess students' answers, as described in (Keisuke et al. 2015), but that is in the scope of future work.

References

Xavier Carreras and Lluis Marquez. 2005. Introduction to the CoNLL-2004 Shared Task: Semantic Role Labeling. *Proceedings of the 9th Conference on Computational Natural Language Learning*, pages 152-164. Ann Arbor, MI.

Karen Mazidi and Paul Tarau. 2016. Infusing NLU into Automatic Question Generation. *Proceedings of The 9th International Natural Language Generation conference,* pages 51-60. Edinburgh, UK.

Matteo Pagliardini, Prakhar Gupta, Martin Jaggi, Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features. NAACL 2018.

Naveed Afzal, Ruslan Mitkov, and Atefeh Farzindar. 2011. Unsupervised Relation Extraction Using Dependency Trees for Automatic Generation of Multiple-Choice Questions. *Canadian AI 2011. LNAI 6657,* pages 32-43. Denver, CO.

Keisuke Sakaguch, Michael Heilman, and Nitin Madnani. 2015. Effective Feature Integration for Automated Short Answer Scoring. *Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL,* pages 1049-1054. Montreal (QC), Canada.