Sentence Simplification for Question Generation

Feras Al Tarouti and Jugal Kalita
Department of Computer Science
University of Colorado at Colorado Springs
Colorado Springs, Colorado 80918, USA
{faltarou & jkalita}@uccs.edu

Conor McGrory
Department of Computer Science
Princeton University
Princeton, New Jersey 08544
cmcgrory@princeton.edu

***Abstract* - Automatic generation of basic, factual questions from a single sentence is a problem that has received a considerable amount of attention. Some studies have suggested splitting this problem into two parts: first, decomposing the source sentence into a set of smaller, simple sentences, and then transforming each of these sentences into a question. This paper outlines a novel method for the first part, combining two techniques recently developed for related NLP problems. Our method uses a trained classifier to determine which phrases of the source sentence are potential answers to questions, and then creates different compressions of the sentence for each one.**

Index Terms – Sentence Simplification, Question Generation.

I. Introduction

The ability of a speaker to form a grammatical question—to request a specific piece of information from another party—is indispensable in most practical situations involving basic communication. Recently, there has been a significant amount of research towards developing systems that can automatically generate basic questions from input text. This is called the problem of Question Generation (QG). Although some studies in the past have tried to generate questions based on whole blocks of text [1], the majority of recent work done on QG has focused on the problem of generating factual questions from a single sentence.

Early attempts to solve this problem used complicated sets of grammatical rules to transform the input sentence directly into a question [2]. However, Heilman and Smith [3] suggested separating the problem into two steps: first, simplifying the source sentence, and then transforming it into a question. The advantage of this approach is that grammatical rules are much better at transforming simple sentences than they are at transforming complex ones. Our paper outlines a method for performing the first step, which we refer to as the problem of Simplified Statement Extraction (SSE).

II. Prior Work

Two problems in NLP that are related to QG are cloze question generation and sentence compression. In a cloze question, the student is asked, after reading the text, to complete a given sentence by filling in a blank with the correct word. One example could be the question

A ___________ is a conceptual device used in computer science as a universal model of computing processes.

In this case, the answer would be Turing machine. However, selecting which phrase(s) in the sentence to delete is somewhat difficult. A question like

A Turing Machine ____ a conceptual device used in computer science as a universal model of computing processes.

with the verb *is* as the answer would be completely useless to a student interested in testing knowledge of computer science. An automatic cloze question generator needs to distinguish informative questions from extraneous ones. Because the quality of a cloze question can depend on relationships between a large number of factors, to generate high-quality questions, Becker et al. [4] train a logistic regression classifier on a corpus of questions paired with human judgments of their quality.

Sentence compression is the problem of transforming an input sentence into a shorter version that is grammatical and retains the most important semantic elements of the original. Knight and Marcu [5] used a statistical language model where the input sentence is treated as a noisy channel and the compression is the signal, while Clarke and Lapata [6] used a large set of constituency parse tree manipulation rules to generate compressions.

Heilman and Smith [7] developed a rule-based algorithm, which is called Simplified Factual Statement Extractor (SFSE), that extracts multiple simple sentences from a source sentence. While traditional sentence compression algorithms usually compress a long sentence into a single short sentence, SFSE extracts one or more simple sentences from a long sentence. By doing so, the algorithm ensures that important information, which can be used to generate questions, is reserved. Each simple sentence produced by the algorithm can be easily converted into a question. The SFSE algorithm uses textual entailment recognition to split the complex sentences into a set of true simple sentences given the original sentence. There are two linguistic phenomena that the SFSE algorithm works on: semantic entailment and presupposition. By extracting multiple simplified statements from the source sentence, they increased the number of possible questions that could be generated.

Kalady et al. [8] presented a rule-based algorithm for generating definitional and factoid questions from a multi-sentence source. Here, to generate definitional questions, keywords from the source document are selected using a summarization system [9]. These keywords are called Up-Keys. Then, the Up-Keys are mapped to simple question templates. For instance, if the word “Ebola” is selected as a keyword, then, it would be mapped to the template: <Question-word>is <Up-Key>? to generate the question “What is Ebola?”. To generate factoid questions, the source sentence is preprocessed to produce simple clauses by splitting the independent clauses within the sentence and replacing pronouns. Then, using a tree regular expression language, the algorithm tries to identify named entities, subject-auxiliaries, appositives, subject verb object structures, prepositional phrases and adverbials. Finally, for each case of these patterns, a procedure is applied to generate a question. The authors evaluated the system by comparing the questions generated by the system with manually generated questions. The system scored an average precision of 0.46 and an average recall of 0.68. The authors reported that the overall quality of the generated questions decreases as the length of the source sentence increases.

Filippova and Strube [10] developed a method where a compressed sentence is generated by pruning the dependency parse tree of the input sentence. Using the Tipster corpus, they calculated the conditional probabilities of specific dependencies occurring after a given head word. These were used, in combination with data on the frequencies of the words themselves, to calculate a score for each dependency in the tree. They then formulated the problem of compressing the sentence as an integer linear program. Each variable corresponded to a dependency in the tree. A value of 1 meant the dependent word of that dependency would be preserved in the compression, and a value of 0 meant that it would be deleted. Constraints were added to restrict the structure and length of the compression, and the objective function set to be maximized was the sum of the scores of the preserved dependencies. The central assumption made by Filippova and Strube’s method is that the frequency with which a particular dependency occurs after a given word is a good indicator of its grammatical necessity.

III. Simplified Sentence Extraction

A. Problem Statement

We divide the process of QG into three major steps: answer selection, sentence simplification and question generation. Figure 1 shows the QG process applied on the sentence *John performed Yoga, which is a Hindu spiritual discipline, to reduce his stress*. In this work we focus on the answer selection and sentence simplification steps, which we refer to as simplified statement extraction (SSE).We define the problem of (SSE) as follows.

For a source sentence S, create a set of simplified statements $\{s_i \ldots s_n\}$ that are semantic entailments of S. A sentence is considered a *simplified statement* if it is a declarative sentence (a statement) that can be directly transformed into a question-answer pair without any compression.

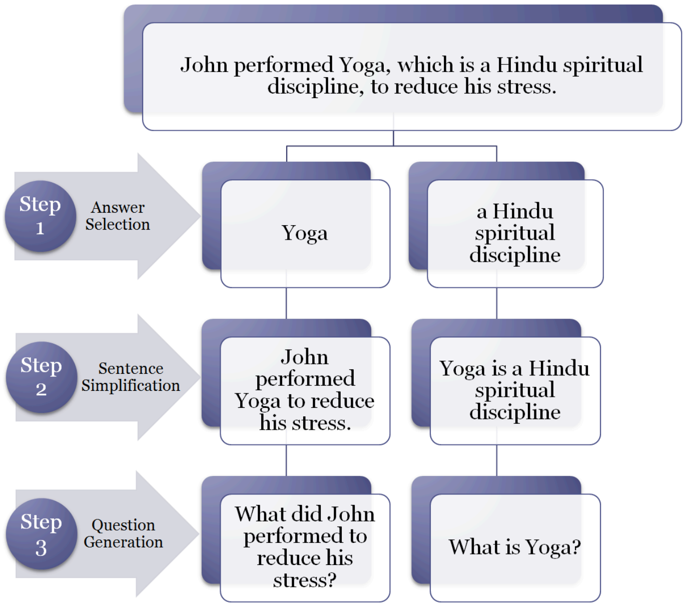
John performed Yoga, which is a Hindu spiritual discipline, to reduce his stress.
Step 1
Answer Selection
Yoga
a Hindu spiritual discipline
Step 2
Sentence Simplification
John performed Yoga to reduce his stress.
Yoga is a Hindu spiritual discipline
Step 3
Question Generation
What did John performed to reduce his stress?
What is Yoga?

Fig. 1 The process of question generation applied to the sentence *John performed Yoga, which is a Hindu spiritual discipline, to reduce his stress.*

B. Solution Steps

As Becker et al. [4] showed, there are certain phrases in S that make sense as answers to questions and others that do not. The idea behind our SSE system is that knowledge of which phrases in S are good answers can inform the compression process, preventing us from missing important information and thereby maximizing coverage. We solve the SSE problem in two parts: first identifying potential answers, and then generating for each of these answers a compression of S where that answer is preserved. These compressions form the set $\{s_i\}$ of simplified statements. Our goal when compressing for a particular answer is to find the shortest grammatical compression of S that contains the given answer. To select potential answers from the input sentence, we use a slightly modified version of Becker et al.’s cloze question generation system. Once we have the set of possible answers, we use a more substantially modified version of Filippova and Strube’s [10] dependency tree pruning method to generate the set of shortest grammatical compressions of S that contain each of the answers.

IV. Answer Selection

We implemented the answer selection system using the Stanford NLP Toolkit [11] and the Weka machine learning software [12]. It uses the corpus of sentences, QA pairs, and human judgments from Becker et al. [4] to train a classifier to find the nodes in the parse tree of the input sentence that are most likely viable answers to questions.

A. Feature Set

The dependency relations identified by the Stanford NLP Toolkit are a set of grammatical relations between governor

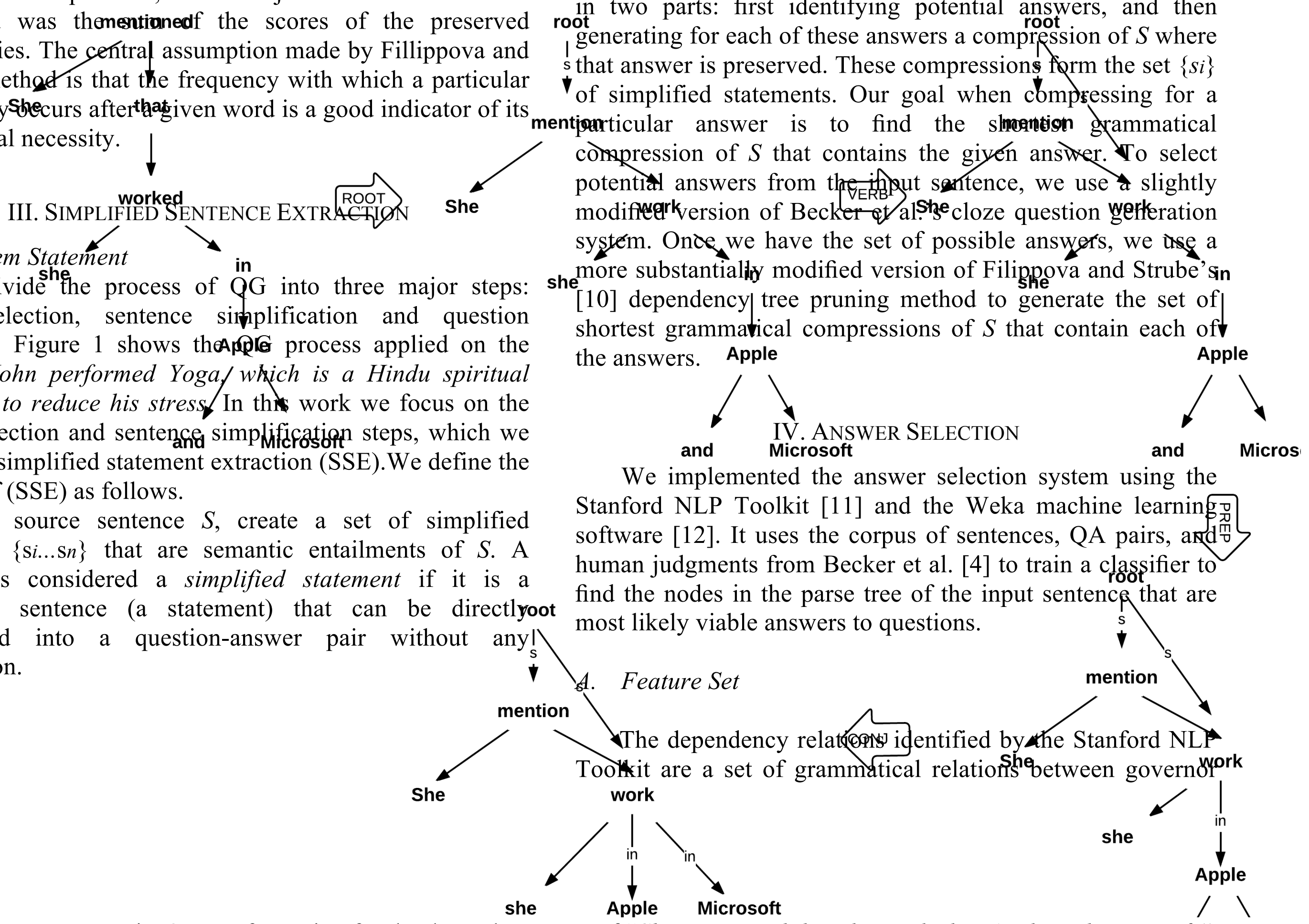

Fig. 2 Transformation for the dependency tree of “*She mentioned that she worked in Apple and Microsoft.*”

and dependent words in a sentence [11]. Some examples include verb-subject, verb-indirect object, noun-modifier, and noun-determiner. For our purposes, we used the 56 basic relations defined in the Stanford library to categorize all of our dependencies.

Our features can be divided into three basic categories: token count features, syntactic features, and semantic features. The token count features contained 5 features which had to do with the length of the answer in comparison to the length of the sentence, like the raw lengths of both and the length of the answer as a percentage of the length of the question. Examples of syntactic features we use are the Penn POS tag of the word [13] that comes immediately before the answer, the tag of the word that comes immediately after, and the set of tags of words contained in the answer phrase. The semantic features use the Stanford dependencies system and are completely different than the semantic features used by [4]. These include the dependency relation between the head of the answer phrase and its governor in the sentence, the set of relations between governors in the answer and dependents not in the answer, the set of relations with both governors and dependents in the answer, and the distance in the constituency tree between the answer node and its maximal projection.

B. *Classifier*

The classifier used in our system is the Weka Logistic classifier [14]. This is a binary logistic regression classifier, similar to the one used by Becker et al [4].

C. *Human Judgments*

The corpus provided by Becker et al. [4] consists of slightly over 2,000 sentences, each with a selected answer phrase and four human judgments of the quality of the answer. Our program used the four judgments to calculate a score for each answer, which we then used to determine how to classify it in the data set. This score is then compared to the threshold value (a pre-set constant in the program). If the score is greater than or equal to this value, the answer is classified in the data set as “Good”. Otherwise, it is classified as “Bad”.

D. *Results*

We used the program to produce a data set from the Becker et al. corpus. This data set was created using a threshold value of 1.0 (all four human judges have to rate the sentence as “Good”). A random sample of the sentences was drawn from this data to produce a subset with a comparable amount of “Good” and “Bad” sentences. This set contained a total of 582 instances, 278 of which were “Good” and 304 of which were “Bad”. We tested both the Weka Logistic classifier and the Weka Simple Logistic classifier on the data using 10-fold cross-validation.

For the Logistic classifier, the correct classification rate was 72.3%, the true positive rate was 78.4%, and the false positive rate was 33.2%.

V. Sentence Compression

Filippova and Strub [10] developed an unsupervised sentence compression approach that compresses sentences by pruning unnecessary subtrees from the dependency tree of the sentence. Three processes are applied to the dependency tree to compress a sentence: transformation, compression and linearization.

The tree transformation process is carried out in four steps: ROOT, VERB, PREP and CONJ. In the ROOT step, a root node is inserted to the tree. Then, in the VERB step, the root node is connected to all the inflected verbs in the tree with edges labeled as *s*. After that, all auxiliary verbs edges are deleted from the tree and grammatical properties of the verbs are stored to be recovered later. In the next step (PREP), all prepositional nodes are replaced with labels on the edges which connect a head to the respective noun. Finally, in the CONJ step, for every chain of conjoined non-verb words, the chain is split and each conjunction on it is connected directly to the head of the first of the chain using edges that have labels similar to the edge connecting the first conjunction to the head. Figure 2 shows the transformation for the dependency tree of the sentence *She mentioned that she worked in Apple and Microsoft*.

The tree compression process is performed by removing edges from the dependency graph produced by the transformation process. To select which edge should be removed from the graph, a score is computed for the subtree

Table I
SAMPLES OF SIMPLIFIED SENTENCES ALONG WITH THEIR MFQ VALUES AND EVALUATIONS

Source Sentence	Selected Answer	Simplified Sentence	MFQ	Evaluation
John walks to his shop every morning.	John	John walks every morning.	1.096	Correct
John performed Yoga, which is a Hindu spiritual discipline, to reduce his stress.	Yoga	John performed Yoga to reduce his stress.	1.044	Correct
	a Hindu spiritual discipline	Yoga is a Hindu spiritual discipline.	1.085	Correct
Apothecaries cared about plague victims when physicians left in 1665.	Apothecaries	Apothecaries cared about physicians.	1.098	Incorrect
A major concern of the Indian government is the trafficking of wildlife products such as tiger and leopard skins.	leopard skins	A major concern of the government is the trafficking of wildlife products such as tiger and leopard skins.	1.088	No Result

connected by each edge. We first calculate probabilities of dependencies occurring after head words and use this as an estimate of the grammatical necessity of different dependencies given the presence of a head word. Along with all of the constraints placed on the ILP in the original model by Filippova and Strube [10], we add an extra constraint that ensures the preservation of the answer phrase in the compression. We then use a linear program solver to solve the ILP for all length values between 0 and the length of *S*, generating a set of compressions of *S* with all possible lengths. From these compressions, we use a 3-gram model to calculate the Mean First Quartile (MFQ) grammaticality metric described by Clark et al. [15]. Compressions with an MFQ value lower than a threshold are deemed grammatical, and the shortest of these is selected as the final compression of *S* for the given answer.

Finally, in the tree linearization process, the selected words are presented in the order they appear in the original sentence.

A. Dependency Probabilities

In addition to the feature set used in the selection part of the system, we included additional ones such as collapsed dependencies [11], which are created when closed-class words like and, of, or by are made part of the grammatical relations. To calculate the frequencies of dependencies after certain head words, we use a pre-parsed section of the Open American National Corpus [16]. To prevent rounding errors, we used a smoothing function when calculating the probabilities from the frequency data. Finally, to avoid problems that come with probability values of zero, our system linearly maps the smoothed probability $P(\ell|h)$ values from [0,1] to $[10^{-4},1]$.

B. Integer Linear Program

We formulate the compression problem as an ILP. For each dependency with the Stanford type ℓ, holding between head word *h* and dependent word *w*, we create variable $x^{\ell}_{h,w}$. These variables must each take on a value of 0 or 1 in the solution, where dependencies whose variables are equal to 1 are preserved in the resulting compression and dependencies whose variables are equal to 0 are deleted, along with their dependent words. The ILP maximizes the objective function

$$f(x) = \sum_{x} x^{\ell}_{h,w} \cdot \mathbf{P}_{(\ell,h)} \cdot t(\ell, \mathbf{P}_{(\ell,h)})$$

where t is the *tweak function*, which corrects discrepancies between frequency and grammatical necessity that occur with some specific types of dependencies. Filippova and Strube used two constraints in their model to preserve tree structure and connectedness in the compression. To ensure that all of the words in the pre-selected answer *A* are also preserved, we include in our model the extra constraint

$$\forall w \in A, \sum_{h,\ell} x^{\ell}_{h,w} \geq 1.$$

We solved these integer linear programs using lp solve[1], an open-source LP and ILP solver.

C. Shortest Grammatical Compression

In order to find the shortest grammatical compression of *S*, our system first finds a solution to the ILP for *S* and *A* for every value of α (the maximum length constraint parameter) between the length of *S* and the length of *A*. Because the constraints also specify that every word in *A* is preserved in the compression, any model where α is less than the length of *A* would have no solution.

To determine the grammaticality of the compressions, we use the MFQ metric [15], which is created using the Berkeley Language Model Toolkit [17] and trained on the OANC text. It considers the log-probabilities of all of the n-grams in the given sentence, selects the first quartile (25% with the lowest values), and calculates the mean of the ratios of each n-gram log-probability over the unigram log-probability of that n-gram's head word. The larger the MFQ value is, the less likely the sentence is to be grammatical.

Our system looks through the list of different length compressions and selects the shortest compression with an MFQ value less than a specified threshold (we used a threshold of 1.14). This compression is returned as the simplified statement extracted from *S* for the answer *A*. Table I shows MFQ values of some simplified sentences along with their evaluation.

D. Results

The functionality of the compression system can be demonstrated with sample outputs from the compressor. For example, given the sentence *She mentioned that she worked in Apple and Microsoft*, the simplified sentence extractor can determine that *she*, *Apple*, *Microsoft* are potential answers for which a question generator can ask questions. For the answers *Apple* and *Microsoft*, the system generates as the compression *She worked in Apple and Microsoft*, which happens to be a compression of the original sentence with the pre-identified answer preserved in it. This statement can now be passed to a question generator as a simple sentence that can potentially generate the question *Where did she work in?* or something similar.

VI. Evaluation and Discussion

To evaluate our algorithm for (SSE), we compare it with the (SFSE) algorithm presented by Heilman and Smith [7]. The source sentences we use are complex sentences from the Simple-Complex Sentence Pairs produced by [18]. The Simple-Complex Sentence Pairs were collected from the English Wikipedia[2] and Simple English Wikipedia[3]. Simple Wikipedia targets children and non-native English speakers.

[1] http://sourceforge.net/projects/lpsolve
[2] http://en.wikipedia.org
[3] http://simple.wikipedia.org

Authors of Simple Wikipedia use short sentences composed of easy words to write articles. The collected dataset include 65,133 articles paired from Simple Wikipedia and Wikipedia using dump files downloaded from Wikimedia[4].

We randomly selected a sample of 85 complex senetences from the corpus. Our algorithm was able to produce 215 compressed sentences, while the SFSE algorithm was able to produce 119 compressed sentences.

To measure the performance of the algorithms, we compute the percentage of the correct compressed sentences produced by both methods. We asked independent human evaluators to evaluate the compressed sentences through a web application. The evaluators were asked if the algorithm produced a new shorter sentence and whether the new sentence is correct or not.

As Figure 3 shows, our (SSE) algorithm was able to produced new compressed sentences in 84.4% of the cases while the (SFSE) algorithm was able to produces new compressed sentences in 73.38% of the cases. Moreover, our (SSE) algorithm generated 43.3% correct sentences and 41.1% of incorrect sentences, whereas the (SFSE) algorithm generated 46.77% correct sentences and 26.77% of incorrect sentences.

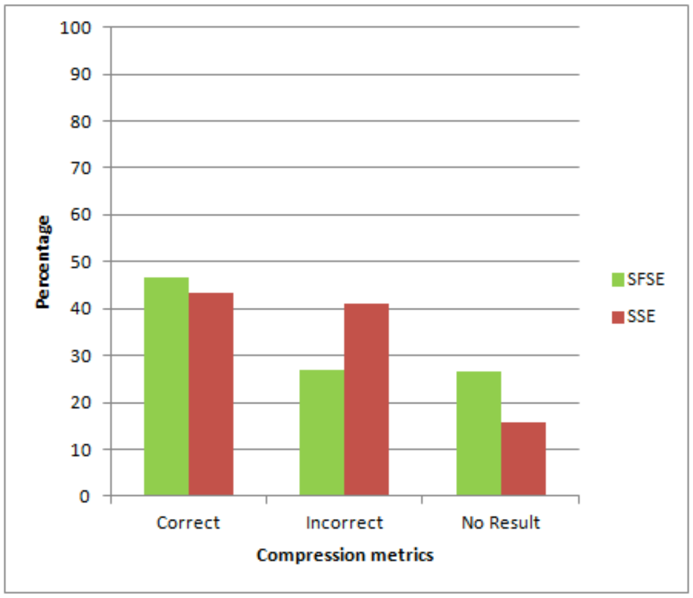

Figure 3 The ratings of sentences produced by our (SSE) algorithm and the (SFSE) algorithm presented by [7].

4 http://download.wikimedia.org

We notice here that our method produced more compressed sentences but with lower grammatical accuracy compared with the rule-based approach presented by [7]. We believe that this is normal since we are using a statistical method for shortening the source sentences. When using rule-based methods, one has the advantage of controlling the output. However, one major disadvantages of using rule-based methods is that it is limited to the implemented set of rules. Our results clearly show that the rule-based method produced fewer sentences compared with the statistical method we use. Another disadvantage of using a rule-based method is that it is also limited to a single language whereas statistical methods can be adapted to use with additional languages.

VII. Conclusion

The key principle on which our system is built is that selecting the answer at the beginning of the QG process and using it to guide SSE can improve the coverage of the system. We implemented a machine learning-based approach for answer selection and developed a way to compress a sentence while leaving a specified answer phrase intact. Although we have not yet been able to perform large scale tests on this system where the output is rated by human judges, we have generated some good output sentences. In the near future, this system will be integrated with a direct declarative-to-interrogative transformation system to produce a full, functional, QG system.

References

[1] H. Kunichika, T. Katayama, T. Hirashima, and A. Takeuchi, “Automated question generation methods for intelligent English learning systems and its evaluation,” in Proceedings of ICCE2004, 2003, pp. 2–5.
[2] J. H. Wolfe, “Automatic question generation from text-an aid to independent study,” in ACM SIGCUE Outlook, vol. 10. ACM, 1976, pp. 104–112. [Online]. Available: http://dl.acm.org/citation.cfm?id=803459
[3] M. Heilman and N. A. 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. Association for Computational Linguistics, 2010, pp. 609–617.
[4] L. Becker, S. Basu, and L. Vanderwende, “Mind the gap: learning to choose gaps for question generation,” in Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Association for Computational Linguistics, 2012, pp. 742–751.
[5] K. Knight and D. Marcu, “Statistics-based summarization-step one: Sentence compression,” in AAAI/IAAI, 2000, pp. 703–710.
[6] J. Clarke and M. Lapata, “Modelling Compression with Discourse Constraints.” in EMNLP-CoNLL, 2007, pp. 1–11.
[7] M. Heilman and N. A. Smith, “Extracting simplified statements for factual question generation,” in Proceedings of QG2010: The Third Workshop on Ques-tion Generation, 2010.
[8] S. Kalady, A. Elikkottil, and R. Das, “Natural language question generation using syntax and keywords,” in Proceedings of QG2010: The Third Workshop on Question Generation, 2010, pp. 1–10.
[9] R. Das and A. Elikkottil, “Automatic Summarizer to aid a Q/A system,” International Journal of Computer Applications, vol. 1, no. 1, pp. 108–112, 2010.
[10]K. Filippova and M. Strube, “Dependency tree based sentence compression,” in Proceedings of the Fifth International Natural Language

Generation Conference. Association for Computational Linguistics, 2008, pp. 25–32.

[11]M.-C. De Marneffe and C. D. Manning, “The Stanford typed dependencies representation,” in Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation. Association for Computational Linguistics, 2008, pp. 1–8.

[12]G. Holmes, A. Donkin, and I. H. Witten, “WEKA: a machine learning workbench,” in Proceedings of the 1994 Second Australian and New Zealand Conference on Intelligent Information Systems,1994, Nov. 1994, pp. 357–361.

[13]M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini, “Building a Large Annotated Corpus of English: The Penn Treebank,” Comput. Linguist., vol. 19, no. 2, pp. 313–330, Jun. 1993.

[14]S. L. Cessie and J. C. V. Houwelingen, “Ridge Estimators in Logistic Regression,” Journal of the Royal Statistical Society. Series C (Applied Statistics), vol. 41, no. 1, pp. 191–201, Jan. 1992.

[15]A. Clark, G. Giorgolo, and S. Lappin, “Statistical representation of grammaticality judgements: the limits of n-gram models,” CMCL 2013, p. 28, 2013.

[16]N. Ide and C. Macleod, “The american national corpus: A standardized resource of american english,” in Proceedings of Corpus Linguistics 2001, vol. 3, 2001.

[17]A. Pauls and D. Klein, “Faster and Smaller N-gram Language Models,” in Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1, ser. HLT ’11. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011, pp. 258–267.

[18]Z. Zhu, D. Bernhard, and I. Gurevych, “A monolingual tree-based translation model for sentence simplification,” in Proceedings of the 23rd international conference on computational linguistics. Association for Computational Linguistics, 2010, pp. 1353–1361.