Infusing NLU into Automatic Question Generation

Karen Mazidi and **Paul Tarau**
Department of Computer Science and Engineering
University of North Texas, Denton TX 76207 USA
karenmazidi@my.unt.edu, paul.tarau@unt.edu

Abstract

We present a fresh approach to automatic question generation that significantly increases the percentage of acceptable questions compared to prior state-of-the-art systems. In our evaluation of the top 20 questions, our system generated 71% more acceptable questions by informing the generation process with Natural Language Understanding techniques. The system also introduces our DeconStructure algorithm which creates an intuitive and practical structure for easily accessing sentence functional constituents in NLP applications.

1 Introduction

Question generation has been described as a dialogue and discourse task, drawing on both Natural Language Understanding and Natural Language Generation (Rus et al., 2012). However, current state-of-the-art question generation systems pay scant attention to the NLU aspect, an issue we address in this work. The question generator we present explores means of infusing NLU analysis (Allen, 1995) into the task of automatically generating questions from expository text for educational purposes. Ginzburg's work on Questions Under Discussion (2012) frames discourse as a series of questions to be addressed. Expository text could be viewed from this perspective: it is a monologue from which the author hopes the reader would be able to answer a set of questions. Automatic question generation, then, could be viewed as a process of discovering unasked questions within the monologue.

2 Prior work in question generation

Pioneering work in QG dates back to Wolfe (1976) who not only demonstrated the feasibility of automatically generating questions from text but also that automatically generated questions could be as effective as human-authored questions (Wolfe, 1977). Question generation has received revived interest in recent years, spurred in part by a series of workshops on question generation, the last one of which occurred in 2010 (Boyer and Piwek, 2010).

2.1 Common approaches to QG

Apart from a few outliers in specialized domains with limited results, the majority of question generation systems input a text source, parse the sentences, and transform sentences into questions. Two major design decisions are: (1) selecting a parser, and (2) deciding whether to use external templates or internal rules for sentence-to-question transformation. In a recent survey of question generation approaches for educational applications, Le et al. (2014) observed that template-based approaches tended to perform better than systems that syntactically rearrange the source text. Our observation is that generating any question type is theoretically possible in any approach, but that some approaches make some types easier to generate than others.

One of the most popular QG approaches involves parsing text with a PSG (phrase structure grammar) parser and then forming questions using templates (Rus et al., 2007; Wyse and Piwek, 2009; Liu et al., 2010; Liu et al., 2012) or transformation rules and tree manipulation tools (Gates, 2008; Heilman, 2011; Ali et al., 2010). Heilman notes (2011)

Sentence	ArgN	Dep. Label	Meaning
1. John broke *the window*.	Arg1	dobj	second entity in relation
2. John was *angry*.	Arg1	acomp	property of subject
3. John felt *that everyone always ignored him*.	Arg1	ccomp	proposition of subject
4. John is *an angry man*.	Arg1	attr	definition of subject
5. John wanted *to make his presence heard*.	Arg1	xcomp	purpose
6. John began *bleeding profusely*.	Arg1	xcomp	action

Table 1: Arg1 versus Dependency Labels

that these purely syntactic approaches do not allow higher-level abstractions that may be possible with more semantically informed approaches.

An alternative to the phrase-structure parse is the SRL (semantic role label) parse which identifies for each predicate in a sentence, its associated arguments and modifiers, and specifies their semantic roles. A QG system can then extract arguments and modifiers for question construction (Mannem et al., 2010; Lindberg et al., 2013; Mazidi and Nielsen, 2014; Chali and Hasan, 2015). These systems are able to generate a wider variety of questions than the phrase structure approach and are not as closely bound to the sentence source text.

A third type of parse used in QG systems is the dependency parse, which connects words in a sentence in a graphical structure based on their grammatical and functional relations. Although the SRL parse is sometimes referred to as a shallow semantic parse, certain dependency relations give greater insight into semantics than the SRL parse. The italicized portions of the sentences in Table 1 were all parsed as `Arg1` by the SRL parser. In contrast, the labels provided by the dependency parser are quite varied, and provide opportunities to glean varied meanings from what is simply `Arg1` in the SRL parse. Although the dependency parse had been used as an ancilliary tool and for sentence simplification, Mazidi et al. (2015) was the first to fully exploit dependency relations in question generation.

Another recent innovative approach (Labutov et al., 2015) used crowd sourcing to develop QG templates by leveraging the structure of their source data, Wikipedia. As an example, articles about persons have similar subsections such as Eary Life, Influences, and so forth, so that templates formed for one person should transfer reasonably well to others. It remains to be seen how this innovative but source-specific approach would transfer to other text sources such as textbooks on a wide range of topics. Interestingly, the approach relies on the observation that expository text tends to be rather redundant in structure, an observation that has parallels with the observations we offer in this paper.

2.2 NLU: the missing piece of the puzzle

Most prior work in QG views a sentence as a string of constituents and proceeds to rearrange those constituents into as many questions as possible according to grammar rules. In contrast, the work we present here first classifies what a sentence is communicating by examining the pattern of constituent arrangement. As described below, the identification of this *sentence pattern* is key to determining what type of question should be asked about that sentence, as opposed to generating questions on every possible sentence constituent. This sentence identification process is part of the DeconStructure algorithm.

3 DeconStructure algorithm

The DeconStructure algorithm has one major objective: a sentence is taken apart to be restructured in such a way that reveals what it is trying to communicate. This involves two major phases: deconstruction, then structure formation, In the deconstruction phase, the sentence is parsed with both a dependency parse and an SRL parse. Additionally, word lemmas and parts of speech are gathered, along with named entity information. In the structure formation phase, the algorithm first divides the sentence into independent clauses, then utilizes output from all parses to identify clause components and assigns each a label that represents its function within the clause. Before delving into the specifics of these two phases, we justify the approach with theoretical foundations.

Token	PSG	SRL	Dependency
1 The	(S(NP*	B-A0	det(algorithm-3,the-1)
2 DeconStructure	*	I-A0	compmod(algorithm-3,DeconStructure-2)
3 algorithm	*)	E-A0	nsubj(creates-4,algorithm-3)
4 creates	(VP*	S-V	ROOT(root-0,creates-4)
5 a	(NP(NP*	B-A1	det(representation-7,a-5)
6 functional-semantic	*	I-A1	amod(representation-7,functional-semantic-6)
7 representation	*)	I-A1	dobj(creates-4,representation-7)
8 of	(PP*	I-A1	adpmod(representation-7,of-8)
9 a	(NP*	I-A1	det(sentence-10,a-9)
10 sentence	*)))	E-A1	adpobj(of-8,sentence-10)
11 by	(PP*	B-AM-MNR	adpmod(creates-4,by-11)
12 leveraging	(S(VP*	I-AM-MNR	adpcomp(by-11,leveraging-12)
13 multiple	(NP*	I-AM-MNR	amod(parses-14,multiple-13)
14 parses	*)))))	E-AM-MNR	dobj(leveraging-12,parses-14)

Table 2: Comparing Parser Outputs: Phrase Structure Grammar, Semantic Role Label, Dependency

Constituent	Text	Head	Governor
predicate	creates	4	0
subject	the DeconStructure algorithm	3	4
dobj	a functional-semantic representation of a sentence	7	4
MNR	by leveraging multiple parses	11	4

Table 3: Front End DeconStructure for Sentence in Table 2

3.1 Theoretical Foundations

The Cambridge Grammar of the English Language (Huddleston et al., 2002) identifies three essential concepts in the analysis of sentences: (1) Sentences have parts, which may themselves have parts, (2) The parts of sentences belong to a limited range of types, and (3) The parts have specific roles or functions within the larger parts they belong to. Kroeger (2004) identifies three aspects of sentence structure: (1) argument structure, (2) constituent structure, and (3) functional structure. With these concepts in mind, the DeconStructure algorithm was designed with three desiderata: (1) Identify sentence constituents in a manner that is intuitive yet consistent with linguistic foundations, (2) Classify constituents from a set of types indicating the semantic function of constituents within sentences, and (3) Determine the sentence pattern: a sequence consisting of the root predicate, its complements and adjuncts.

3.2 Parser Comparisons

In prior work, we determined that no one parse tells us everything we would like to know about a sentence, as each of the three parser types gives its own particular viewpoint. Table 2 compares parser outputs. The PSG (phrase structure grammar) parse identifies sentence constituents and labels phrases with the appropriate phrase label such as VP, NP, and so forth. The SRL parse (semantic role label parse, also called predicate-argument parse) identifies numbered arguments of the predicate as well as modifiers. The dependency parse provides a representation of the grammatical relations between individual words in a sentence. Table 3 shows the front end of the DeconStructure created by the algorithm. The DeconStructure algorithm gleans the most important aspects from each of the parsers and combines them in to a structure that is both intuitive and practical, thus making sentence elements readily available for downstream NLP applications, such as the question generation system presented in this paper. Although Table 3 shows the front end of the DeconStructure, it is important to note that all of the parsing information from Table 2, as well as generated information such as sentence type, is available in the DeconStructure sentence object.

Pattern	Meaning	Frequency
S-V-acomp	adjectival complement that describes the subject	8%
S-V-attr	nominal predicative complement defining the subject	14%
S-V-ccomp	clausal complement indicating a proposition of subject	7%
S-V-dobj	indicates the relation between two entities	28%
S-V-iobj-dobj	indicates the relation between three entities	$< 1\%$
S-V-parg	phrase describing the how/what/where of the action	17%
S-V-xcomp	non-finite clause-like complement	8%
S-V	indicates an action of the entity	14%
other	combinations of constituents	4%

Table 4: Typical Sentence Pattern Distribution in Expository Text

3.3 Advantages of Multiple Parsers

The DeconStructure algorithm is encoded in a Python program that first parses sentences with Microsoft Research's SPLAT [1] (Quirk et al., 2012), which provides constituency parsing, dependency parsing using universal dependency labels (McDonald et al., 2013), semantic role labeling, tokenizing, POS tagging, lemmatization, and other NLP functions through a JSON (JavaScript Object Notation) request. It should be noted that the DeconStructure algorithm can be implemented with any parser that provides an SRL and dependency parse. Hence it does not require a custom parser as do other representations such as AMR (Banarescu et al., 2012).

The DeconStructure algorithm (see Algorithm 1) exploits synergies between the SRL and dependency parses. For example, a prepositional phrase that is dependent on the verb can be an argument or an adjunct. Knowing what role the PP is playing is crucial for NLP applications but the dependency parse does not identify this information. However, the SRL will label PPs with numbered arguments if they are arguments of the verb. By checking if a PP dependent on a root verb is also a numbered argument in the SRL parse, the PP can be identified as an argument; otherwise it will be considered to be an adjunct.

Complements are words, phrases and clauses that complete the meaning of the verb, including the objects of traditional grammar (Carnie, 2013; Huddleston et al., 2002). The universal dependency label set has six distinct labels that may be internal complements of the VP: direct object, indirect object, attr (attribute), acomp (adjectival complement), ccomp (clausal complement) and xcomp (non-finite clause-like complement) (McDonald et al., 2013). Including the PP-argument and the case in which there are no internal VP arguments, this gives eight distinct patterns for major constituents in clauses. Table 4 provides pattern distribution data observed from collections of expository text. Table 7 provides sample sentences for each structure, along with generated questions. Note that all modifiers and PP that are not core arguments are available in the DeconStructure for placement in generated questions.

4 Question generation

As seen in Table 4, these *sentence patterns* fall into a surprisingly small number of categories. For each sentence, the QG system classifies its sentence pattern prior to the question generation phase. The sentence pattern is key to determining what type of question should be asked about that sentence. This analysis was based on text extracted from open source textbooks as well as Wikipedia passages, where each text passage consisted of the text of one chapter section, or Wikipedia text of equivalent length. In order to identify patterns to be included in the QG system, the following criteria was used: (1) Does the sentence pattern occur frequently across passages in different domains? (2) Is the semantic information conveyed by the sentence pattern consistent across different instances? and (3) Does the sentence pattern identify important content in source sentences so that generated questions will be meaningful and not trivial?

An independent clause can be viewed as a proposition, and the predicate identifies the relationship, property or state of the entities participating in the

[1] http://research.microsoft.com/en-us/projects/msrsplat/

proposition. The predicate determines the number of participants, or arguments, that are allowed (Kroeger, 2005). In the S-V-iobj-dobj pattern, for example, there must be 3 entities identified in the sentence. The predicate is often the main verb but there are other constructions in which the predicate can be found in other syntactic categories. The `acomp` constituent follows a copula verb which has negligible semantic content in this construction. The meaning is carried by the `acomp`, which may be an adjective or a noun. Linguists often used the term `xcomp` to denote predicate complements of various syntactic categories (Kroeger, 2005). In contrast, the universal dependency relations divide the complements into `acomp` for AP, `attr` for NP, `ccomp` for subordinate clauses, leaving `xcomp` for VP. It matters what syntactic category a complement belongs to because this provides important semantic indications of what the clause is saying. Take for instance a `ccomp` compared to a `dobj`. They differ syntactically in that the `ccomp` is a clause whereas the `dobj` is a phrase. Semantically, the `dobj` identifies the second entity in the predicate relation whereas the `ccomp` can be viewed as an independent proposition either indicated by or about the subject.

Algorithm 1 DeconStructure Algorithm

```
S ← set of parsed sentences
for each sentence s ∈ S do
    DIVIDEINDEPCLAUSES(s)
    for each indepClause ic ∈ s: do
        Step 1: Add predicate complex
        ic[pred.label] ← predicate
        icRoot = pred.index
        Step 2: Add constituents
        for each dep ∈ dependencies do
            if dep.gov == icRoot then
                ic[const.label] ← dp
        Step 3: Add ArgMs to IC
        for each AM in ArgMs for icRoot do
            ic[AM.label] ← ArgM
        Step 4: Determine pp type
        for each pp in PPs do
            if pp == ArgN then
                pp.label = ppArg
            else
                pp.label = ppMod
        Step 5: Determine ic structure
        Determine ic type (passive, active, ...)
        Classify ic pattern
        Flag sentences with questionable parse
```

4.1 Templates and question generation

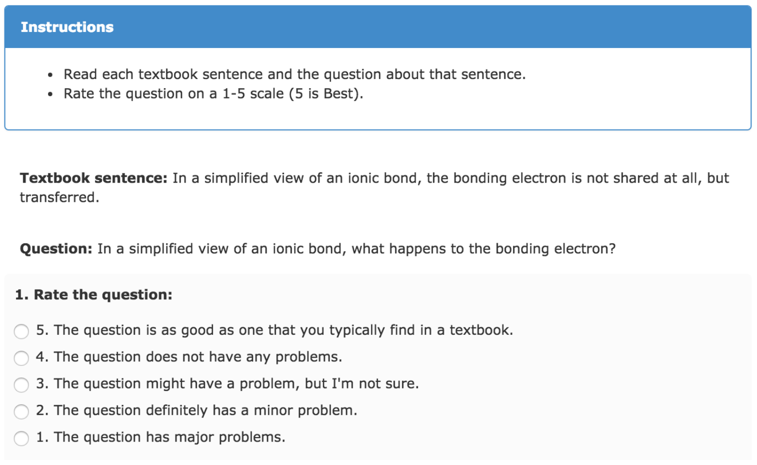
After a sentence object is created for each independent clause of each sentence via the DeconStructure algorithm, the sentence pattern is compared against approximately 60 templates. If a template matches the pattern, a question can be generated. Templates are designed to ask questions related to the major point of the sentence as identified in the pattern (see Table 7). Templates also contain filter conditions which are checked. Filter conditions may check for the presence or absence of particular verbs (particularly be, do and have), whether the sentence is in active or passive voice, and other conditions that are documented in the template file. More information is available[2] for those interested in implementation details.

4.2 Ranking question importance

A question generation system can increase its utility by ranking the output questions in order to identify which questions are more likely to be acceptable. Heilman and Smith used a logistic regression question ranker which focused on linguistic quality. The ranker more than doubled the percentage of acceptable questions in the top 20% of generated questions, from 23% to 49% (2011). The logistic regression approach has also attempted by others, but with less success. One system (Lindberg et al., 2013) was able to identify with 86% precision that a question was not acceptable; however, their annotator considered 83% of the questions to be unacceptable questions so the utility of the classifer is unclear.

Given that our system typically outputs questions that are grammatically correct, we decided to evaluate the question importance, an often overlooked criterion (Vanderwende, 2008). To that end we employed the TextRank algorithm (Mihalcea and Tarau, 2004) for keyword extraction. For a given input passage, the top 25 nouns were identified by TextRank. Then each generated question was given

[2]http://www.karenmazidi.com/


Instructions

- Read each textbook sentence and the question about that sentence.
- Rate the question on a 1-5 scale (5 is Best).

Textbook sentence: In a simplified view of an ionic bond, the bonding electron is not shared at all, but transferred.

Question: In a simplified view of an ionic bond, what happens to the bonding electron?

1. Rate the question:

- ◯ 5. The question is as good as one that you typically find in a textbook.
- ◯ 4. The question does not have any problems.
- ◯ 3. The question might have a problem, but I'm not sure.
- ◯ 2. The question definitely has a minor problem.
- ◯ 1. The question has major problems.

Figure 1: Sample Amazon Mechanical Turk HIT (Human Intelligence Task).

a score based on the percentage of top TextRank words it contained, with a penalty for very short questions such as *What is* `keyword?` Our evaluation demonstrated that outputting *important* questions also increases their acceptability scores.

5 Evaluation

There is no standard way to evaluate automatically generated questions. Recent work in QG and other NLP applications favors evaluation by crowdsourcing which has proven to be both cost and time efficient and to achieve results comparable to human evaluators (Snow et al., 2008; Heilman and Smith, 2010). We compared our system performance to the most-frequently cited prior question generation system by Heilman and Smith (2011). The evaluation was conducted using Amazon's Mechanical Turk Service. Workers were selected with at least 90% approval rating and who were located in the US and proficient in US English. To monitor quality, work was submitted in small batches, manually inspected, and run through software to detect workers whose ratings did not correspond well with fellow workers. Each question was rated on a 1-5 scale by 4 workers. The four scores were averaged. Figure 1 shows a sample HIT. Agreement between each set of workers and the average had a Pearson's correlation $r = .71$, showing high agreement.

5.1 Test data

Test data consists of 10 science and humanities passages, one each from 10 open source textbooks from OpenStax and Saylor. All text sources are written at an early college reading level with an average of 83 sentences per passage. Each passage represents the text of one textbook chapter section, chosen at random. Table 5 lists the topics in the test data set, along with the number of sentences in each file and the number of questions generated by the Heilman & Smith system and our system. The H&S system takes an overgenerate-and-rank approach, generating almost 5 questions per input sentence. In contrast, our system generates an average closer to one question per every 2 input sentences by focusing on the important content in each sentence but not generating questions when conditions are not favorable for generating a good question.

Table 5: Test Data and Questions Generated

Topic	Sents	H&S	M&T
Epithelial Tissue	148	600	77
Protists	118	545	76
Bankruptcy	37	159	23
Network Layers	79	267	55
Monetary Policy	90	431	37
Uzbekistan	71	351	52
Legislature	73	375	55
Jackson Era	46	279	24
Stages of Sleep	72	339	44
Education	103	715	50
Average	83	406	50
Generation Percents		488%	60%

5.2 Results

The evaluation looked at the top 20 questions output from each system for each input file, with each system performing its own internal ranking. Table 6 compares the average MTurk worker ratings for each file for the two systems. Our system had a higher rating for every topic. When averaging all 200 questions, the Heilman & Smith system had an average rating of 2.9. Our system had an average rating of 3.7. The results are statistically significant, $p < 0.001$, as determined by the Student's t-Test. Figure 2 shows a side-by-side histogram of the score distributions between the two systems. The histogram demonstrates that the majority of the Heilman and Smith system questions are below the mid-point of 3.0 and that the majority of our questions are above this mid-point. Using > 3.0 as the acceptability threshold, 72% of our questions are acceptable whereas only 42% of the Heilman and Smith questions pass this threshold. This is an increase in the acceptablity percentage of the top questions of 71%. Interestingly, the Heilman and Smith percentage of 42% found in our evaluation of their top 20 questions is close to the 49% acceptable percentage they found in their analysis of the top 20 *percent* of their generated questions.

Table 6: Average Scores for Top 20 Questions

Topic	H&S	M&T
Epithelial Tissue	2.6	3.9
Protists	2.6	4.1
Bankruptcy	2.7	3.5
Network Layers	3.0	3.9
Monetary Policy	2.8	3.8
Uzbekistan	3.3	3.6
Legislature	3.0	3.1
Jackson Era	3.4	3.7
Stages of Sleep	3.0	4.0
Education	2.6	3.1
Average	2.9	3.7

5.3 Error analysis

Analysis of unacceptable questions revealed both sources of errors and areas for future work. Some errors are caused by idiomatic langauge. For example the sentence: *Few members spend time in the chamber other than when they are speaking or voting,* resulted in the generated question: *What do few members spend?* In this case *time* grammatically is the direct object which is why this question was generated, but *spend time* is an idiom. One way to avoid generating this question would be to look for specific idiomatic phrases and rephrase them, essentially translating the idiomatic language into more direct language.

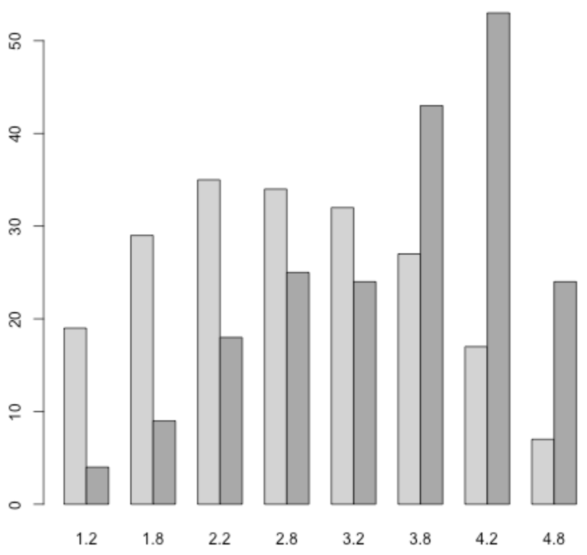


Figure 2: Score Distributions. Light:H&S, Dark:M&T

Another issue is that some templates work better with some topics other than others. For example, a template that matches the S-V-attr pattern is *How would you describe* `subject`? which generated the question: *How would you describe a gland?* with the answer: *a structure made up of one or more cells modified to synthesize and secrete chemical substances*. However in another passage it generates the question: *How would you describe the sea?* from the sentence: *The sea was once the fourth-largest body of water in the world.* Techniques need to be employed to identify noun phrases that are suitable for definition questions, a task to be explored in future work.

Another problem is insufficient preprocessing to remove sentences such as: *Different episodes of monetary policy are indicated in the figure*, which generated the question: *Where are different episodes of monetary policy indicated?* Our system prepro-

Pattern and Sample
1. `S-V-acomp` Adjectival complement that describes the subject. S: Brain waves during REM sleep appear similar to brain waves during wakefulness. Q: Indicate characteristics of brain waves during REM sleep.
2. `S-V-attr` Nominal predicative complement following copula, often defining the subject. S: The entire eastern portion of the Aral sea has become a sand desert, complete with the deteriorating hulls of abandoned fishing vessels. Q: How would you describe the entire eastern portion of the Aral sea?
3. `S-V-ccomp` Clausal complement indicates a proposition of or about the subject. S: Monetary policy should be countercyclical to counterbalance the business cycles of economic downturns and upswings. Q: What evidence could support the notion that monetary policy should be countercyclical?
4. `S-V-dobj` Indicates the relation between two entities. S: The early portion of stage 1 sleep produces alpha waves. Q: What does the early portion of stage 1 sleep produce?
5. `S-V-iobj-dobj` Indicates the relation between three entities. S: The Bill of Rights gave the new federal government greater legitimacy. Q: What gave the new federal government greater legitimacy?
6. `S-V-pparg` Prepositional phrase that is required to complete the meaning. S: REM sleep is characterized by darting movement of closed eyes. Q: What is REM sleep characterized by?
7. `S-V-xcomp` Non-finite clause-like complement. S: Irrigation systems have been updated to reduce the loss of water. A: For what purpose have the irrigation systems been updated?
8. `S-V` May contain phrases that are not considered arguments such as ArgMs. S: The 1828 campaign was unique because of the party organization that promoted Jackson. Q: Why was the 1828 campaign unique?

Table 7: Sample Questions by Sentence Type

cessing unit removes most but not all references to figures and tables.

Yet another issue is with text that conveys a sequence of events, in which case a given sentence in isolation may be vague. For example the sentence: *Political authority appeared to rest with the majority as never before*, generated the question: *What did political authority appear to do?* This question is vague out of context. This problem suggests that certain topics require features not available in general-purpose question generators. And indeed, there is an inherent conflict in designing a general-purpose question generation system as opposed to one targeted for a specific topic or source text.

6 Discussion

The question generation system presented here introduced a fresh approach to question generation by analyzing intrasentential structure and meaning with the DeconStructure algorithm. The pattern of the constituent structure indicates what meaning can be inferred from the sentence. This enables generation of questions relevant to the central point of a sentence and avoids the overgeneration problem of prior work. The approach can be implemented with off-the-shelf parsers that provide both a dependency and an SRL parse. The QG system achieved a 71% increase in the percentage of acceptable questions from among the top system-ranked questions compared to the most cited prior state-of-the-art system. This improvement is due in part to the internal NLU analysis of what the sentence is communicating and to the application of the TextRank algorithm to identify the most important questions.

References

Husam Ali, Yllias Chali, and Sadid A Hasan. 2010. Automation of question generation from sentences. In *Proceedings of QG2010: The Third Workshop on Question Generation*, pages 58–67.

James Allen. 1995. *Natural language understanding*. The Benjamin/Cummings Publishing Company.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2012. Abstract meaning representation (amr) 1.0 specification. In *Parsing on Freebase from Question-Answer Pairs. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. Seattle: ACL*, pages 1533–1544.

Kristy Elizabeth Boyer and Paul Piwek. 2010. Proceedings. In *Proceedings of QG2010: The Third Workshop on Question Generation. Pittsburgh: questiongeneration. org*.

Andrew Carnie. 2013. *Syntax: A generative introduction*. John Wiley & Sons.

Yllias Chali and Sadid A Hasan. 2015. Towards topic-to-question generation. *Computational Linguistics*.

D Gates. 2008. Generating look-back strategy questions from expository texts. In *The Workshop on the Question Generation Shared Task and Evaluation Challenge, NSF, Arlington, VA. http://www. cs. memphis. edu/~ vrus/questiongeneration//1-Gates-QG08. pdf*.

Jonathan Ginzburg. 2012. *The interactive stance*. Oxford University Press.

Michael Heilman and Noah A Smith. 2010. Rating computer-generated questions with mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 35–40. Association for Computational Linguistics.

Michael Heilman. 2011. *Automatic factual question generation from text*. Ph.D. thesis, Carnegie Mellon University.

Rodney Huddleston, Geoffrey K Pullum, et al. 2002. The cambridge grammar of english. *Language. Cambridge: Cambridge University Press*, pages 1–23.

Paul Kroeger. 2004. *Analyzing syntax: a lexical-functional approach*. Cambridge University Press.

Paul R Kroeger. 2005. *Analyzing grammar: An introduction*. Cambridge University Press.

Igor Labutov, Sumit Basu, and Lucy Vanderwende. 2015. Deep questions without deep understanding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*.

Nguyen-Thinh Le, Tomoko Kojiri, and Niels Pinkwart. 2014. Automatic question generation for educational applications–the state of art. In *Advanced Computational Methods for Knowledge Engineering*, pages 325–338. Springer.

David Lindberg, Fred Popowich, John Nesbit, and Phil Winne. 2013. Generating natural language questions to support learning on-line. In *Proceedings of the 14th European Workshop on Natural Language Generation.* Association for Computational Linguistics.

Ming Liu, Rafael A Calvo, and Vasile Rus. 2010. Automatic question generation for literature review writing support. In *Intelligent Tutoring Systems*, pages 45–54. Springer.

Ming Liu, Rafael A Calvo, and Vasile Rus. 2012. G-asks: An intelligent automatic question generation system for academic writing support. *Dialogue and Discourse: Special Issue on Question Generation*, 3(2):101–124.

Prashanth Mannem, Rashmi Prasad, and Aravind Joshi. 2010. Question generation from paragraphs at upenn: Qgstec system description. In *Proceedings of QG2010: The Third Workshop on Question Generation*, pages 84–91.

Karen Mazidi and Rodney D Nielsen. 2014. Linguistic considerations in automatic question generation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics. Baltimore, Maryland: Association for Computational Linguistics*.

Karen Mazidi and Rodney D Nielsen. 2015. Leveraging multiple views of text for automatic question generation. In *Artificial Intelligence in Education, Springer LNCS*.

Ryan T McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith B Hall, Slav Petrov, Hao Zhang, Oscar Täckström, et al. 2013. Universal dependency annotation for multilingual parsing. In *ACL (2)*, pages 92–97. Citeseer.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. Association for Computational Linguistics.

Chris Quirk, Pallavi Choudhury, Jianfeng Gao, Hisami Suzuki, Kristina Toutanova, Michael Gamon, Wen-tau Yih, Lucy Vanderwende, and Colin Cherry. 2012. Msr splat, a language analysis toolkit. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstration Session*, pages 21–24. Association for Computational Linguistics.

Vasile Rus, Zhiqiang Cai, and Arthur C Graesser. 2007. Experiments on generating questions about facts. In *Computational Linguistics and Intelligent Text Processing*, pages 444–455. Springer.

Vasile Rus, Brendan Wyse, Paul Piwek, Mihai Lintean, Svetlana Stoyanchev, and Cristian Moldovan. 2012. A detailed account of the first question generation shared task evaluation challenge. *Dialogue and Discourse*, 3(2):177–204.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. 2008. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 254–263. Association for Computational Linguistics.

Lucy Vanderwende. 2008. The importance of being important: Question generation. In *Proceedings of the 1st Workshop on the Question Generation Shared Task Evaluation Challenge, Arlington, VA*.

John H Wolfe. 1976. Automatic question generation from text-an aid to independent study. In *ACM SIGCUE Outlook*, volume 10, pages 104–112. ACM.

John H Wolfe. 1977. Reading retention as a function of method for generating interspersed questions. Technical report, DTIC Document.

Brendan Wyse and Paul Piwek. 2009. Generating questions from openlearn study units.