Generating Natural Language Questions to Support Learning On-Line

David Lindberg Fred Popowich
School of Computing Science
Simon Fraser University
Burnaby, BC, CANADA
dll4,popowich@sfu.ca

John Nesbit Phil Winne
Faculty of Education
Simon Fraser University
Burnaby, BC, CANADA
nesbit,winne@sfu.ca

Abstract

When instructors prepare learning materials for students, they frequently develop accompanying questions to guide learning. Natural language processing technology can be used to automatically generate such questions but techniques used have not fully leveraged semantic information contained in the learning materials or the full context in which the question generation task occurs. We introduce a sophisticated template-based approach that incorporates semantic role labels into a system that automatically generates natural language questions to support online learning. While we have not yet incorporated the full learning context into our approach, our preliminary evaluation and evaluation methodology indicate our approach is a promising one for supporting learning.

1 Introduction

Ample research (e.g., Callender and McDaniel, 2007) shows that learners learn more, and more deeply, if they are prompted to examine their learning materials while and after they study. Often, these prompts consist of questions related to the learning materials. After reading a given passage or section of text, learners are familiar with learning exercises which consist of questions they need to answer.

Questioning is one of the most common and intensively studied instructional strategies used by teachers (Rus and Graesser, 1989). Questions embedded in text, or presented while learners are studying text, are hypothesized to promote self-explanation which is known to increase comprehension and enhance transfer of learning (e.g., Rittle-Johnson, 2006).

Traditionally, these questions have been constructed by educators. Recent research, though, has investigated how natural language processing techniques can be used to automatically generate these questions (Kalady et al., 2010; Varga and Ha, 2010; Ali et al., 2010; Mannem et al., 2010). While the automated approaches have generally focussed on syntactic features, we propose an approach that also takes semantic features into account, in conjunction with domain dependent and domain independent templates motivated by educational research. After introducing our question generation system, we will provide a preliminary analysis of the performance of the system on educational material, and then outline our future plans to tailor the questions to the needs of specific learners and specific learning outcomes.

2 Question Generation from Text

The task of question generation (QG) from text can be broadly divided into three (not entirely disjoint) categories: syntax-based, semantics-based, and template-based. Systems in the syntactic category often use elements of semantics and vice-versa. A system we would call template-based must to some extent use syntactic and/or semantic information. Regardless of the approach taken, systems must perform at least four tasks:

1. content selection: picking spans of source text (typically single sentences) from which questions can be generated
2. target identification: determining which specific words and/or phrases should be asked about
3. question formulation: determining the appropriate question(s) given the content identified
4. surface form generation: producing the final surface-form realization

Task 2 need not always precede task 3; target identification can drive question formulation and vice-versa. A system constrained to generating specific kinds of questions will select only the targets appropriate for those kinds of questions. Conversely, a system with broader generation capabilities might pick targets more freely and (ideally) generate only the questions that are appropriate for those targets. We consider the methods used in performing tasks 2 and 4 to be the primary discriminators in determining the category into which a given method is best placed. This is not the only way one might classify a QG system. However, we believe this method allows us to best compare and contrast our approach with previous approaches.

Syntax-based methods comprise a large portion of the existing literature. Kalady et al. (2012), Varga and Ha (2010), Wolfe (1976), and Ali et al. (2010) provide a sample of these methods. Although each of these efforts has differed on a few details, they have followed the same basic strategy: parse sentences using a syntactic parser, simplify complex sentences, identify key phrases, and apply syntactic transformation rules and question word replacement.

The methods we have labeled "semantics-based" use method(s) of target identification (task 2) that are primarily semantic, using techniques such as semantic role labeling (SRL). Given a sentence, a semantic role labeler identifies the predicates (relations and actions) along with the semantic entities associated with each predicate. Semantic roles, as defined in PropBank (Palmer et al., 2005), include Arg0, Arg1, ..., Arg5, and ArgA. A set of modifiers is also defined and includes ArgM-LOC (location), ArgM-EXT (extent), ArgM-DIS (discourse), ArgM-ADV (adverbial), ArgM-NEG (negation), ArgM-MOD (modal verb), ArgM-CAU (cause), ArgM-TMP (time), ArgM-PNC (purpose), ArgM-MNR (manner), and ArgM-DIR (direction). We adopt the shorter CoNLL SRL shared task naming conventions (Carreras and Màrquez, 2005) (e.g., A0 and AM-LOC).

Mannem et al. (2010), for example, introduce a semantics-based system that combines SRL with syntactic transformations. In the content selection stage, a single sentence is first parsed with a semantic role labeler to identify potential targets. Targets are selected using simple selection criteria. Any of the predicate-specific semantic arguments (A0-A5), if present, are considered valid targets. Mannem et al. further identify modifiers AM-MNR, AM-PUNC, AM-CAU, AM-TMP, AM-LOC, and AM-DIS as potential targets. These roles are used to generate additional questions that cannot be attained using only the A0-A5 roles. For example, AM-LOC can be used to generate a *where* question, and an AM-TMP can be used to generate a *when* question. After targets have been identified, these, along with the complete SRL parse of the sentence are passed to the question formulation stage. Two heuristics are used to rank the generated questions. Questions are ranked first by the depth of their predicate in the dependency parse of the original question. This is based on the assumption that questions arising from main clauses are more desirable than those generated from deeper predicates. In the second stage, questions with the same rank are re-ranked according to the number of pronouns they contain, with questions with fewer pronouns having higher rank.

One limitation of the syntax and semantics-based methods is that they generate questions by rearranging the surface form of sentences. Question templates offer the ability to ask questions that are not so tightly-coupled to the exact wording of the source text. A question template is any predefined text with placeholder variables to be replaced with content from the source text. Question templates allow question generation systems to leverage human expertise in language generation.

The template-based system of Cai et al. (2006) uses Natural Language Generation Markup Language (NLGML), a language that can be used to generate not only questions but any natural language expression. NLGML uses syntactic pattern matching and semantic features for content selection and question templates to guide question formulation and surface-form realization. Note that a pattern need not specify a complete syntax tree. Additionally, patterns can impose semantic constraints. However, simple "copy and paste" templates are not a panacea for surface-form realization. Mechanisms for changing capitalization of words and changing verb conjugation (when source sentence verbs are to appear in the output text) need to be provided: NLGML provides some such functions.

3 Our Approach

We develop a template-based framework for QG. The primary motivation for this decision is the ability of a template-based approach to generate questions that are not merely declarative to interrogative transformations. We aim to address some of the limitations of the existing approaches outlined in the previous section while leveraging some of their strengths in novel ways. We combine the benefits of a semantics-based approach, the most important of which is not being tightly-constrained by syntax, with the surface-form flexibility of a template-based approach.

The data used to develop our approach was obtained from a collection of 25 documents prepared for educational research purposes within the Faculty of Education at SFU. All hand-coded rules we describe below were motivated by patterns observed in this development data. This collection was modeled after a high-school science curriculum on global warming, with vocabulary and discourse appropriate for learners in that age group. Although the collection included a glossary of key terms and their definitions, this resource was used only for evaluation purposes as described in Section 4.

3.1 Semantic-based templates

Previous template-based methods have used syntactic pattern matching, which does provide a great deal of flexibility in specifying sentences appropriate for generating certain types of questions. However, this flexibility comes at the expense of generality. As seen in Wyse and Piwek (2009), who use Stanford Tregex (Levy and Andrew, 2006) for pattern matching, the specificity of syntactic patterns can make it difficult to specify a syntactic pattern of the desired scope. Furthermore, semantically similar entities can span different syntactic structures, and matching these requires either multiple patterns (in the case of Cai et al., 2006) or a more complicated pattern (in the case of Wyse and Piwek, 2009).

If we want to develop templates that are semantically motivated, more flexible in terms of the content they successfully match, and more approachable for non-technical users, we need to move away from syntactic pattern matching. Instead, we match *semantic patterns*. We define a *semantic pattern* as the SRL parse of a sentence and the named entities (if any) contained within the span of each semantic role. We use Stanford NER (Finkel et al., 2005) for named entity recognition. Figure 1 shows a sentence and its corresponding semantic pattern. Notice this sentence has two predicates, each with its own semantic arguments. Each of these predicate-argument structures is a distinct *predicate frame*.

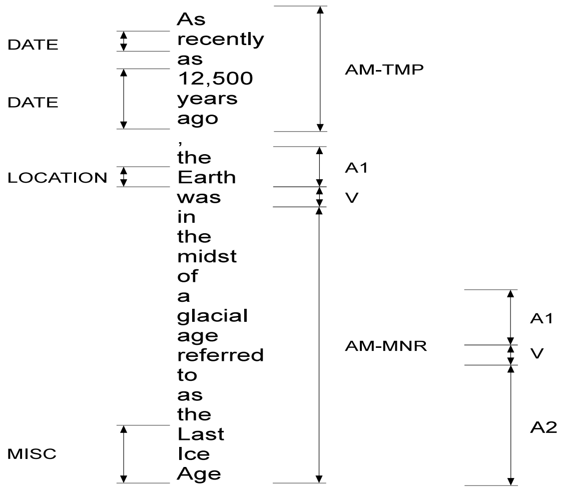
DATE
DATE
LOCATION
MISC
As recently as 12,500 years ago , the Earth was in the midst of a glacial age referred to as the Last Ice Age
AM-TMP
A1
V
AM-MNR
A1
V
A2

Figure 1: A sentence and its semantic pattern

Even the shallow semantics of SRL can identify the semantically interesting portions of a sentence, and these semantically-meaningful substrings can span a range of syntactic patterns. Figure 2 shows a clear example of this phenomenon. In this example, we see two sentences expressing the same semantic relationship between two concepts, namely, the fact that trapped heat causes the Earth's temperature to increase. In one case, this causation is expressed in an adjective phrase, while the other uses a sentence-initial prepositional phrase. The parse trees are generated using the Stanford Parser (Klein and Manning, 2003). The AM-CAU semantic role captures the cause in both sentences. It is impossible to accomplish the same feat with a single NLGML pattern. However, it is possible to capture both with a single Tregex pattern.

The principle advantage of semantic pattern matching is that a single semantic pattern casts a narrow semantic net while casting a large syntactic net. This means fewer patterns need to be defined by the template author, and the patterns are more compact.

Our templates have three components: plaintext, slots, and slot options. Plaintext forms the

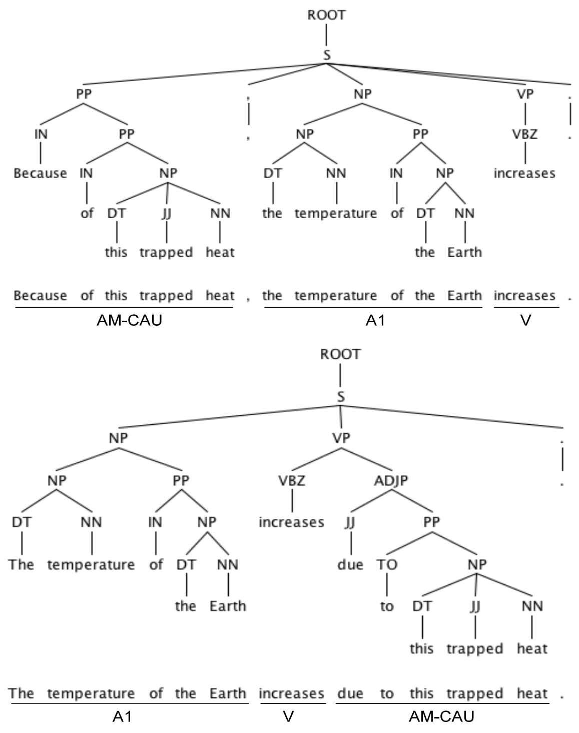
Because of this trapped heat , the temperature of the Earth increases .
AM-CAU
A1
V
The temperature of the Earth increases due to this trapped heat .
A1
V
AM-CAU

Figure 2: Two different syntax subtrees subsumed by a single semantic role

skeleton into which semantically-meaningful substrings of a source sentence are inserted to create a question. The only restrictions on the plaintext is that it cannot contain any text that looks like a slot but is not intended as one, and it cannot contain the character sequence used to delineate the plaintext from the slots appearing outside the plaintext. Aside from these restrictions, any desired text is valid.

Slots facilitate sentence and template matching. They accept specific semantic arguments, and can appear inside or outside the plaintext. These provide the semantic pattern against which a source sentence is matched. A slot inside the plaintext acts as a variable to be replaced by the corresponding semantic role text from a matching sentence, while any slots appearing outside the plaintext serve only to provide additional pattern matching criteria. The template author does not need to specify the complete semantic pattern in each template. Instead, only the portions relevant to the desired question need to be specified. This is an important point of contrast between our template-based approach vs. syntax and semantics-based approaches. We can choose to generate questions that do not include any predicates from the source sentence but instead ask more abstract or general questions about other semantic constituents. We believe these kinds of questions are better able to escape the realm of the factoid, because they are not constrained to the actions and relations described by predicates.

Slot options function much like NLGML functions and are of two types: modifiers and filters. Modifiers apply transformations to the role text inserted into a slot, and filters enforce finer-grained matching criteria. Predicate slots have their own distinct set of options, while the other semantic roles share a common set of options. A template's slots and filters describe the necessary conditions for the template to be matched with a source sentence semantic pattern.

3.2 Predicate slot options

The predicate filter options restrict the predicates that can match a predicate slot. With no filter options specified, any predicate is considered a match. Table 1 shows the complete list of filters.

Filter	Description
be	predicate lemma must not be "be"
!be	predicate lemma must be "be"
!have	predicate lemma must not be "have"

Table 1: Predicate filters

The selection of predicate filters might at first seem oddly limited. Failing to consider the functional differences between various types of verbs (particularly auxiliary and copula) would indeed produce low-quality questions and should in fact be ignored in most cases. For example, consider the sentence "Dinosaurs, along with many other animals, became extinct approximately 65 million years ago." A question such as "What did dinosaurs, along with many other animals, become?" is not particularly useful. We can recognize copula predicates by their surrounding semantic pattern, so in the broad sense, we do not need to adopt any copula-specific rules.

The one exception to the above rule is any copula whose lemma is *be*. The `be` and `!be` filters allow the presence or absence of such a predicate to be detected. This capability is useful for two reasons. First, the presence of such a predicate gives us an inexpensive way to generate definition questions, even if the source text is not written in the form of a definition. Although this will over-generate definition questions, non-predicate filters can be used to add additional mitigating constraints. Second, requiring the absence of such a predicate allows us to actively avoid generating certain kinds of ungrammatical or meaningless questions. Whether using one of these predicates results in ungrammatical questions depends on the wording of the underlying template, so we provide the `!be` filter for the template author to use as needed. Consider the sentence "El Nino is caused when the westerly winds are unusually weak." Without the `!be` filter, one of our templates would generate the question "When can El Nino be?" Applying the `!be` filter prevents this question from being generated.

Like copula, auxiliary verbs are often not suitable for question generation. Fortunately, many auxiliary verbs are also modal and are assigned the label AM-MOD and so do not form predicate frames of their own. Instead, they are included in the frame of the predicate they modify. In other cases auxiliary verbs are not modal, such as in the sentence "So far, scientists have not been able to predict the long term effects of this wobble." In this case, the auxiliary *have* is treated as a separate predicate, but importantly, the span of its A1 includes the predicate *been*. We provide a non-predicate filter to prevent generation when this overlap is present.

The `!have` filter is motivated by the observation that the predicate *have* can appear as a full, non-copula predicate (with an A0 and A1) but often does not yield high-quality questions. For example, consider the sentence "This effect can have a large impact on the Earth's climate." Without the `!have` filter, one of our templates would generate the question "What can this effect have?" With the `!have` filter, that template does not yield any questions from the given sentence.

Predicate modifiers allow the template author to explicitly force a change in conjugation. See Table 2 for the complete set of predicate modifiers, where fps is an abbreviation for first person singular, sps for second person singular, and so on. The `lemma` modifier can appear on its own. However, all other conjugation changes must specify both a *tense* and a *person*. If no modifiers are used, the predicate is copied as-is from the source sentence. Although *perfect* is an aspect rather than a tense, MorphAdorner[1], which we use to conjugate predicates, defines it as a tense, so we have implemented it as a tense filter.

[1]http://morphadorner.northwestern.edu

Modifier	Tense	Modifier
lemma	lemma (dictionary form)	fps
pres	present	sps
prespart	present participle	tps
past	past	fpp
pastpart	past participle	spp
perf	perfect	tpp
pastperf	past perfect	
pastperfpart	past perfect participle	

Table 2: Predicate modifiers

3.3 Non-predicate slot options

The filters for non-predicate slots impose additional syntactic and named entity restrictions on any matching role text. As with predicates, the absence of any non-predicate filters results in the mere presence of the corresponding semantic role being sufficient for matching. See Table 3 for the complete list of non-predicate filters describing restrictions on the role text (RT), role span (RS), and predicate frame (PF) in terms of the semantic type of named entities (and in some cases in terms of non-semantic features).

Filter	Description
null	PF must not contain this semantic role.
!nv	RS must not contain a predicate
dur	RT must contain DURATION
date	RT must contain DATE
!date	RT must not contain a DATE
loc	RT must contain a LOCATION.
ne	RT must contain a named entity
misc	RT must contain a MISC
comp	RT must contain a comparison
!comma	RT must not contain a comma
singular	RT must be singular
plural	RT must be plural

Table 3: Non-predicate filters

The choice of filters again requires some explanation. The `null` and `!nv` filters were foreshadowed above. For slots appearing outside the template's plaintext, the `null` filter explicitly requires that the corresponding semantic role not be present in a source sentence semantic pattern. An A0 slot paired with the `null` filter is the mechanism alluded to earlier that allows for the recognition of copula predicates without the need to examine the predicate itself. The `!nv` filter can be used to prevent ungrammatical questions. We observe that if a role span does include a predicate, resulting questions are often ungrammatical due to the conjugation of that predicate. Applying this filter to the A1 of a predicate prevents generation from a predicate frame whose predicate is a non-modal auxiliary verb.

The named entity filters (`dur`, `!dur`, `date`, `loc`, `ne`, and `misc`) are those most relevant to the corpus we have used to evaluate our approach and thus the easiest to experiment with effectively. Because named entities are used only for filtering, expanding the set of named entity filters is a trivial task.

The filters `comp`, `!comma`, `singular`, and `plural` are syntax-based filters. With the exception of `!comma`, these filters force the examination of the part-of-speech (POS) tags to detect the desired features. The `singular` and `plural` filters let templates be tailored to singular and plural arguments in any desired way, beyond simply selecting appropriate auxiliary verbs. The type of comparison we search for when the `comp` filter is used is quite specific. We search for phrases that describe conditions that are atypical. These can be seen in phrases such "unusually weak," "unseasonably warm," "strangely absent," and so on. These phrases are present when a word whose POS tag is RB (adverb) is followed by a word whose tag is JJ (adjective). Consider a sentence such as "El Nino is caused when the westerly winds are unusually weak." The `comp` filter allows us to generate questions such as "What data would indicate El Nino?" or "How do the conditions that cause El Nino differ from normal conditions?" Although this heuristic does produce both false positives and false negatives, other syntactic features such as comparative adverbs and comparative adjectives are less semantically constrained. Further investigation is needed to determine more flexible ways to recognize descriptions of atypical conditions.

We see two situations in which a comma appears within the span of a single semantic role. The first situation occurs when a list of nouns is serving the role, such as in "Climate change includes changes in precipitation, temperature, and pressure." Here, "changes in precipitation, temperature, and pressure" is the A1 of the predicate *includes*. In cases where a question is only appropriate for single concept (e.g. temperature) rather than a set of concepts, the `!comma` filter prevents such a question from being generated from the sentence above. This has implications for role text containing appositives, the second situation in which a comma appears within a single role span. Such roles are rejected when `!comma` is used. This is not ideal, as removing appositives does not cause semantic roles to be lost from a semantic pattern. Future work will address this problem.

The non-predicate modifiers (Table 4) serve two purposes: to create more fluent questions and to remove non-essential text. Note that the `-tpp`, which forces the removal of trailing prepositional phrases, can have undesired results when applied to certain modifier roles, such as AM-LOC, AM-MNR, and AM-TMP, when they appear in the template plaintext. These modifiers often contain only a prepositional phrase, and in such cases, `-tpp` will result in an empty string being placed into the template.

Modifier	Effect
-lp	If initial token is prep, remove it
-tpp	If RT ends with PP, remove PP
-ldt	If initial token is determiner, remove it

Table 4: Non-predicate modifiers

3.4 Our QG system

Figure 3 shows the architecture and data flow of our QG system. One of the most important things to observe about this architecture is that the templates are an external input. They are in no way coupled to the system and can be modified as needed without any system modifications.

Compared to most other approaches, we perform very little pre-processing. Syntax-based methods in particular have been motivated to perform sentence simplification, because their methods are more likely to generate meaningful questions from short, succinct sentences. We have chosen not to perform any sentence simplification. This decision was motivated by the observation that common methods of sentence simplification can eliminate useful semantic content. For example, Kalady et al. (2010) claim that prepositional phrases are often not fundamental to the meaning of a sentence, so they remove them when simplifying a sentence. However, as Figure 4 shows, a prepositional phrase can contain important semantic information. In that example, removing the prepositional phrase causes temporal information to be lost.

One pre-processing step we do perform is pronominal anaphora resolution (Charniak and Elsner, 2009). Even though we do not split com-

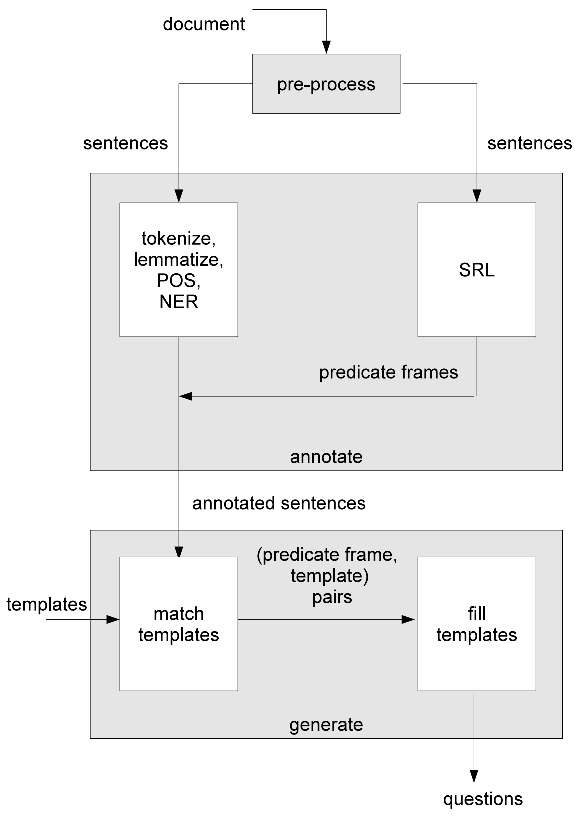

Figure 3: System architecture and data flow

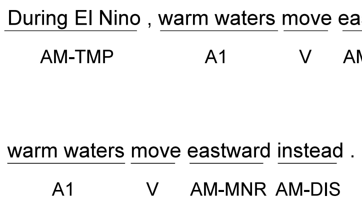
During El Nino , warm waters move eastward instead .
AM-TMP A1 V AM-MNR AM-DIS
warm waters move eastward instead .
A1 V AM-MNR AM-DIS

Figure 4: Semantic information can be lost during sentence simplification. Removing the prepositional phrase from the first sentence leaves the simpler second sentence, but the AM-TMP modifier is lost.

plex sentences and therefore do not create new sentences in which pronouns are separated from their antecedents, this kind of anaphora resolution remains an important step in limiting the number of vague questions.

Each source sentence is tokenized and annotated with POS tags, named entities, lemmata, and its SRL parse. SRL is the cornerstone of our approach. We generate the SRL parse (Collobert et al., 2011) in order to extract a set of predicate frames. Questions are generated from individual predicate frames rather than entire sentences (unless the sentence contains only one predicate frame). Given a sentence, the semantic pattern of each of its predicate frames is compared against that of each template. Algorithm 1 describes the process of matching a single predicate frame (*pf*) to a single template (*t*). Although it is not stated in Algorithm 1, the sentence-level tokenization, lemmata, named entities and POS tags are checked as needed according to the template's slot filters. If a predicate frame and template are matched, they are passed to Algorithm 2, which fills template slots with role text to produce a question. Even in the absence of modifiers, all role text receives some additional processing before being inserted into its corresponding slot. These additional steps include the removal of colons and the things they introduce and the removal of text contained in parentheses. We observe that these extra steps lead to questions that are more meaningful.

Algorithm 1 patternsMatch(pf,t)

```
for all $slot \in t$ do
    if $pf$ does not have $slot.role$ then
        if $null \notin slot.filters$ then
            return $false$
        end if
    else
        for all $filter \in slot.filters$ do
            if $pf.role$ does not match $filter$ then
                return $false$
            end if
        end for
    end if
end for
return $true$
```

Because we generate questions from predicate frames rather than entire sentences, two sentences describing the same semantic entities might result in duplicate questions. To avoid duplicates we keep only the first occurrence of a question.

Using slots and filters, we can now create some interesting templates and see the questions they

Algorithm 2 fillTemplate(t,pf)

```
$question\_text \leftarrow t.plaintext$
for all $slot \in t.plaintext\_slots$ do
    $role\_text \leftarrow pf.role(slot.role).text$
    for all $modifier \in slot.modifiers$ do
        applyModifier($role\_text$,$modifier$)
    end for
    In $question\_text$, replace $slot$ with $role\_text$
end for
return $question\_text$
```

yield. Table 5 shows some templates (**T**) that match the sentence in Figure 1 and the questions (**Q**) that result. Although the questions that are generated are not answerable from the original sentence, they were judged answerable from the source document in our evaluation. The full set of templates is provided in (Lindberg, 2013).

As recently as 12,500 years ago, the Earth was in the midst of a glacial age referred to as the Last Ice Age.
T: How would you describe [A2 -lp misc]? **Q**: How would you describe the Last Ice Age?
T: Summarize the influence of [A1 -lp !comma !nv] on the environment. **Q**: Summarize the influence of a glacial age on the environment.
T: What caused [A2 -lp !nv misc]? ## [A0 null] **Q**: What caused the Last Ice Age?

Table 5: A few sample templates and questions

4 Evaluation

There remains no standard set of evaluation metrics for assessing the quality of question generation output. Some present no evaluation at all (Wyse and Piwek, 2009; Stanescu et al., 2008). Among those who do perform an evaluation, there does appear to be a consensus that some form of human evaluation is necessary. Despite this agreement in principle, approaches tend to diverge thereafter. There are differences in the evaluation criteria and the evaluation procedure.

Most previous efforts in QG have not gone beyond manual evaluation. While some have gone a step further and built models for ranking based on the probability of a question being *acceptable* (Heilman and Smith, 2010), these models have not had a strong basis in pedagogy. While a question that is both syntactically and semantically well-formed is considered acceptable in some evaluation schemes, such questions can greatly outnumber the questions that we can reasonably expect a student would want or have time to answer. We implement a classifier that attempts to identify the questions that are the most pedagogically useful.

For our initial evaluation of the performance of our QG system, we selected a subset of 10 documents from the collection described in the previous section. On average, each document contained 25 sentences. From the 10 documents, our system generated 1472 questions in total, an average of 5.9 questions per sentence. Due to the educational nature of this material, we needed evaluators with educational training rather than naive ones. Accordingly, the questions we generated were evaluated by a graduate student from the Faculty of Education. She was asked to give binary judgements for grammaticality, semantic validity, vagueness, answerability, and learning value. For each question, two aspects of answerability were evaluated. The first aspect was whether the question was answerable from the source sentence from which it was generated. The second was whether the question was answerable given the source document as a whole. The evaluator was given no predetermined guidelines regarding the relationships among the evaluation criteria (e.g., the influence of vagueness and answerability on learning value). This aspect of the evaluation was left to her discretion as an educator. She found that 85% of the questions were grammatical, with 66% of them actually *making sense*. It was determined that 14% of the questions were answerable from the sentence used to generate them, while 20% of them were answerable from the document. Finally, she determined that 17% of the questions had *learning value* according to the prescribed learning outcomes for the curriculum being modeled. Aside from performing this evaluation, the evaluator was not involved in this research.

Given this evaluation, we then built a classifier which used logistic regression (L2 regularized log-likelihood) to classify on learning value. We used length, language model, SRL, named entity, glossary, and syntax features. Length and language model features measure the token count and grammaticality of a question and the sentence from which it was generated. SRL features include the token count of each semantic role in the generating predicate frame, whether each role is required by the matching template, and whether each role's text is used. Named entity features indicate the presence of each of nine named entity types in both the source sentence and generated question. Glossary features note the number of glossary terms that appear in a sentence and question and a measure of the average importance of each term, which we calculated from a simple *in-terms-of* graph (Winne and Hadwin, 2013) we constructed from the glossary. This graph has directed edges between each glossary term and the terms that appear in its gloss. Syntax features identify the depth of the generating predicate frame in the source sentence and the POS tag of its predicate. Without adding noise, the training set had 217 questions with learning value and 1101 questions without learning value. The classifier obtained precision and recall scores of 0.47 and 0.22 respectively for questions with learning value, along with scores of 0.79 and 0.92 for questions with no learning value. We then added *noise* to the training set by relabelling any grammatical question that *made sense* as having *learning value*. This relabelling resulted in a training set of 778 questions with learning value and only 540 questions without learning value. The classifier trained on this noisy set showed a precision score on learning value questions decreased to 0.29 but a dramatic increase in recall to 0.81. For questions with no learning value, the precision increased slightly to 0.86 which was offset by a dramatic decrease in recall to 0.38. So when the system generates a poor quality question, we have a high probability of knowing that it is a poor question which allows us to then filter or discard it.

5 Conclusions

We have shown how a template-based method, using predominately semantic information, can be used to generate natural language questions for use in an on-line learning system. Our templates are based on semantic patterns, which cast a wide syntactic net and a narrow semantic net. The template mechanism supports rich selectional and generational capabilities, generating a large pool from which questions for learners can be selected. A simple automated technique for selecting questions with learning value was introduced. Although this automated technique shows promise for some applications, future investigation into what constitutes a *useful question* in the context of a specific task and an individual learner is needed. Some might argue that it is risky to generate questions that cannot be answered from the source sentence from which they were generated. Although some questions are generated that are not answered elsewhere in a document, there is a benefit in learners being able to recognize that a particular question is not answerable. Our future work will expand both on the types of potential questions generated, and on the selection from the set of potential questions based on the information an individual learner (a) knows, (b) has available in a "library" of saved sources, (c) has operated on while studying online (e.g., tagged), and (d) might find in the Internet. To facilitate this further research, we will be integrating question generation into the nStudy system (Hadwin et al., 2010; Winne and Hadwin, 2013). We will also be performing thorough user studies which will evaluate the generated questions from the learner's perspective in addition to the educator's perspective.

Acknowledgments

This research was supported by an Insight Development Grant (#430-2012-044) from the Social Sciences and Humanities Research Council of Canada and a Discovery Grant from the Natural Sciences and Engineering Research Council of Canada. The authors are extremely grateful to Kiran Bisra from the Faculty of Education for providing information for the evaluation. Finally, special thanks to the reviewers for their comments and suggestions.

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.

Zhiqiang Cai, Vasile Rus, Hyun-Jeong Joyce Kim, Suresh C. Susarla, Pavan Karnam, and Arthur C. Graesser. 2006. Nlgml: A markup language for question generation. In Thomas Reeves and Shirley Yamashita, editors, *Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education 2006*, pages 2747–2752, Honolulu, Hawaii, USA, October. AACE.

Aimee A. Callender and Mark A. McDaniel. 2007. The benefits of embedded question adjuncts for low and high structure builders. *Journal Of Educational Psychology (2007)*, pages 339–348.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 152–164. Association for Computational Linguistics.

Eugene Charniak and Micha Elsner. 2009. Em works for pronoun anaphora resolution. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 148–156. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics.

A.F. Hadwin, M. Oshige, C.L.Z. Gress, and P.H. Winne. 2010. Innovative ways for using nstudy to orchestrate and research social aspects of self-regulated learning. *Computers in Human Behaviour (2010)*, pages 794–805.

Michael Heilman and Noah A Smith. 2010. 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*, pages 609–617. Association for Computational Linguistics.

Saidalavi Kalady, Ajeesh Elikkottil, and Rajarshi Das. 2010. Natural language question generation using syntax and keywords. In *Proceedings of QG2010: The Third Workshop on Question Generation*, pages 1–10.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics.

Roger Levy and Galen Andrew. 2006. Tregex and tsurgeon: tools for querying and manipulating tree data structures. In *LREC 2006*.

David Lindberg. 2013. Automatic question generation from text for self-directed learning. Master's thesis, Simon Fraser University, Canada.

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.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Bethany Rittle-Johnson. 2006. Promoting transfer: Effects of self-explanation and direct instruction. *Child Development (2006)*, pages 1–15.

Vasile Rus and Arthur C Graesser. 1989. Classroom questioning. In *School improvement research series*.

Liana Stanescu, Cosmin Stoica Spahiu, Anca Ion, and Andrei Spahiu. 2008. Question generation for learning evaluation. In *Computer Science and Information Technology, 2008. IMCSIT 2008. International Multiconference on*, pages 509–513. IEEE.

Andrea Varga and Le An Ha. 2010. Wlv: A question generation system for the qgstec 2010 task b. In *Proceedings of QG2010: The Third Workshop on Question Generation*, pages 80–83.

Philip H Winne and Allyson F Hadwin. 2013. nstudy: Tracing and supporting self-regulated learning in the internet. In *International handbook of metacognition and learning technologies*, pages 293–308. Springer.

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

Brendan Wyse and Paul Piwek. 2009. Generating questions from openlearn study units. In *AIED 2009 Workshop Proceedings Volume 1: The 2nd Workshop on Question Generation, 6-9 July 2009, Brighton, UK.*