

# Generating Questions from OpenLearn study units

Brendan WYSE <sup>a,1</sup> and Paul PIWEK <sup>a</sup>

<sup>a</sup>*Computing Department, Open University, UK*

**Abstract.** OpenLearn is a large online educational repository which can be used as a resource for developing and testing Question Generation (QG) systems. OpenLearn is described in detail and its advantages are discussed. To illustrate the use of OpenLearn for QG, a QG system, Ceist <sup>2</sup>, is presented which generates questions from OpenLearn content. Ceist is also offered as a case study for how existing NLP tools can be used to build a QG system, and should provide useful information to those considering the development of their own QG system.

**Keywords.** Question Generation, Data collection, Question Generation System

## Introduction

Recently, a number of communities with interest in Question Generation (QG) including those from Natural Language Processing (NLP) and Intelligent Tutoring Systems (ITS) have met with the aim of setting up a Shared Task and Evaluation Challenge (STEC) for QG (<http://www.questiongeneration.org>).

The current paper aims to contribute to the development of the QG STEC in two ways: firstly by identifying and describing a specific resource for QG, the OpenLearn study units and secondly by demonstrating how state-of-the-art NLP tools can be used and combined to provide a flexible QG system. We explain the process that is used to develop a pattern to match specific target sentences.

The OpenLearn online educational resource project was initiated by the Open University in 2005 with the aim to 'open access to education for all'. Development on OpenLearn began in 2006 and the OpenLearn website now has over 5,400 hours of learning material contained in over 475 study units in a variety of categories from 'Arts and History' to 'IT and Computing' (<http://openlearn.open.ac.uk>).

A QG system, Ceist, has been created which uses OpenLearn as an input data instance. Newcomers to the QG task and to Natural Language Processing (NLP) will discover that a vast amount of work has already been done to solve many different aspects of NLP [1]. It is hoped that by giving an insight into the methods used by Ceist, others may be inspired to produce their own QG systems.

---

<sup>1</sup> Corresponding Author: Brendan Wyse; Email: [bjwyse@gmail.com](mailto:bjwyse@gmail.com)

<sup>2</sup> Ceist is pronounced 'kesht' and is the word for question in the Irish language.

## 1. OpenLearn: A structured online resource of educational materials

There are significant advantages to be gained by using OpenLearn as a data source for QG system development. Some of these advantages are listed below and then described in more detail.

- Applying QG to educational material showcases an excellent practical application of the QG task;
- The material is available in several downloadable formats;
- The material covers a wide range of categories and varying discourse types and, unlike for example Wikipedia articles, it has been authored exclusively by subject experts and passed through a rigorous process of proofreading and editing (with the study units being based mostly on printed Open University course materials);
- The material is available under a Creative Commons ‘Attribution; Non-commercial; Share Alike’ license.

The ability to enhance the learning process has been identified as a key benefit of QG [2]. QG can be used to automatically create questions for assessment and also provide students with examples of questions as a way to help them learn to formulate their own questions. Research has shown that asking the right questions is a good way to measure how well a subject has been understood following learning. Applying QG to OpenLearn helps to establish a baseline against which future QG systems can be measured and demonstrates the benefits of QG to parties not involved in the area of NLP but in other areas such as education and teaching.

Each OpenLearn study unit is available for download in 8 different formats. The formats include a complete printable article with images, a zip file containing the HTML for the article and its associated images, a RSS XML format, an XML format, a complete zipped set of XML files and images and also the article in formats suitable for use with Moodle<sup>3</sup> and the Common Cartridge<sup>4</sup> specification.

The XML schema used by OpenLearn is well designed. Containing over 1,250 lines it defines a hierarchy within an article that allows an appropriately designed machine to easily retrieve the sections of the article it needs. It permits subsections within sections and different types of content within each section such as the section title, paragraphs, media content and exercise activities as can be seen in Fig. 1. This organization of content facilitates the writing of a script for selecting only the relevant parts from each section, ignoring titles or media content.

---

<sup>3</sup> A web application used to create online learning sites. See <http://moodle.org>

<sup>4</sup> A package interchange format for learning content. See <http://www.imsglobal.org>

```

<SubSection id="SEC001_002_003">
  <Title>Fall of the Bastille, 14 July 1789</Title>
  <Paragraph>In a similar mood of aggrieved self-righteousness and re
  <Paragraph>The event seemed to its supporters literally epoch-makin
  <Paragraph>In France, the anniversary of the taking of the Bastille
  <MediaContent src="Gustav 3" id="PDF001_002" type="pdf" target="new
    <Caption/>
    <!-- optional -->
    <SourceReference/>
    <!-- Optional -->
    <Description>Click on 'View document' to read Gustav III of Swe
    <!-- mandatory -->
  </MediaContent>
  <Activity id="EXE001_002">
    <Heading>Exercise</Heading>
    <Question>
      <Paragraph>Now read the second document (letter from Gustav
  
```

Figure 1. An example of OpenLearn’s XML format.

The fact that the range of topics covered is so varied and the articles are available for research makes OpenLearn an ideal data instance for QG system development and testing. Fig. 2 below shows the process for taking an OpenLearn article and generating the input data for Ceist. In the next section, the process of generating questions from this input with Ceist is described.

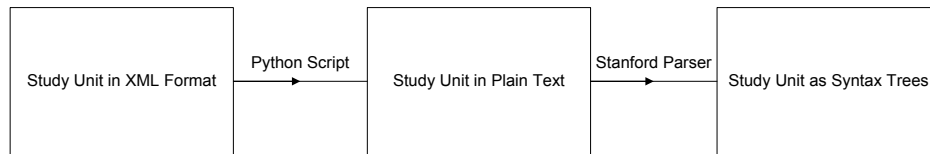


Figure 2. Preparing an OpenLearn study unit for Ceist.

## 2. Ceist v0.1: A Question Generation system

Many of the current documented QG systems use pattern matching to find sentences for which pre-determined questions can be generated based on rules [3][4][5]. This technique was also used with great success in the QA task [6]. Ceist v0.1 follows the same approach.

Ceist takes syntax trees for its input and uses a pattern as part of a rule, to match sentences which are compatible with specific rules. Ceist then uses templates in conjunction with the matched parts of the sentence, to generate a question and the corresponding (short) answer. The main focus for Ceist was to allow the sentence matching patterns in the rules to have maximum flexibility: the patterns can be written to match a very wide range of word sequences or a very specific and narrow range.

As shown in the System Architecture diagram (Fig. 3), Ceist uses the Tregex [7] tree searching code provided by the Stanford NLP group for pattern matching. Ceist achieves flexibility by allowing patterns to be defined to match an individual word or regular expression, up to a POS tag or group of POS tags such as a Noun Phrase. This means all input data must be parsed into a syntax tree during pre-processing. Ceist then has the power to match any individual node in that tree or any specific sub-tree regardless of the node value being text, a POS tag or a phrase grouping tag.

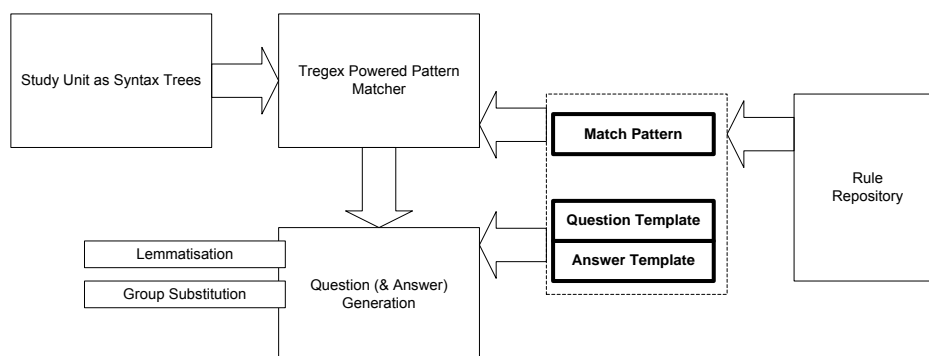


Figure 3. Ceist v0.1 architecture.

The rest of this section will describe the process of customising Ceist such that it can recognise a sentence from an OpenLearn article about the French Revolution and generate a question for that sentence. The sentence which will be the ‘target’ for the rule is *‘Emmanuel-Joseph Sieyès (1748-1836) trained as a priest and became assistant to a bishop’*. The rule will be written so as to generate the question *‘What did Emmanuel-Joseph Sieyès train as?’*.

The steps commonly taken to allow a machine to manipulate or find patterns in words are to tag those words in a sentence and then optionally to group the words into a hierarchy of their sentence parts. Words can be nouns, verbs (in various tenses), and adjectives amongst other types. They can also be grouped into phrases such as noun phrases or verb phrases. Sentences may also contain cardinal numbers and special symbols such as parentheses or brackets, just like the dates in our target sentence, *‘(1748-1836)’*.

The manner in which sentences are parsed into their parts is a well researched area and the state-of-the-art has reached a level of performance which provides excellent results for most sentences [8]. As shown in Fig. 2, the Stanford parser (<http://nlp.stanford.edu/software/lex-parser.shtml>) is used for parsing. It is applied to the relevant text extracted from the XML formatted version of the article as seen in Fig 4. The parser provides the parsed output as a syntax tree as shown in Fig. 5 using labelled bracketing notation.

```

<SubSection id="SEC001_002_002">
  <Title>The Third Estate as the voice of the nation</Title>
  <Paragraph>
Emmanuel-Joseph Sieyès (1748-1836) trained as a priest and became
assistant to a bishop. He had no religious vocation, however, and his fame arose as
the author of a highly influential pamphlet, <i>What is the Third Estate?</i>,

```

Figure 4. An OpenLearn article in the XML format.

```
(ROOT (S (NP (NP (NNP Emmanuel-Joseph) (NNP Sieyès)) (PRN (-LRB- -LRB-)) (NP (NP (CD 1748)) (: --) (NP (CD 1836))) (-RRB- -RRB-)) (VP (VP (VBN trained)) (PP (IN as) (NP (DT a) (NN priest)))) (CC and) (VP (VBD became) (NP (NN assistant)) (PP (TO to) (NP (DT a) (NN bishop))))) (. .)))
```

Figure 5. Syntax Tree output provided by the Stanford parser.

The syntax tree in Fig. 5 can be displayed in a more discernable view using source code again made available by the Stanford NLP team and implemented in Ceist as a feature. Ceist was designed to assist rule design by providing features such as the syntax tree display shown in Fig. 6. This visual aid provides a reference for rule authors and helps them to derive match patterns using the tree nodes.

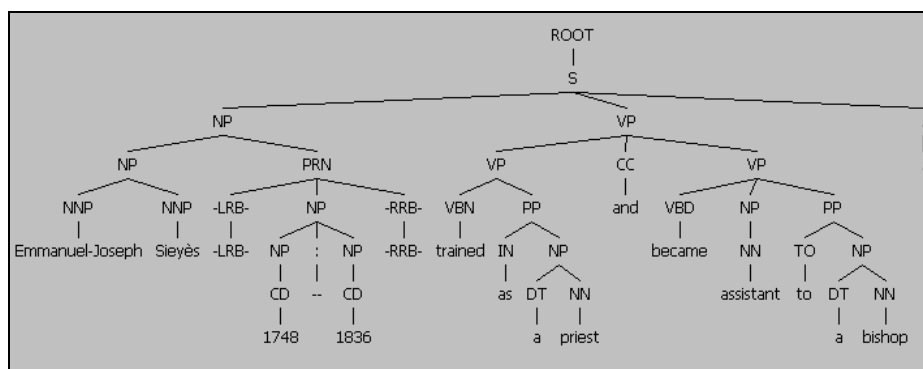


Figure 6. The Syntax Tree as displayed in the QG application.

The tree in Fig. 6 has for its leaves, each word of the sentence. The parent node for each leaf is the Part-Of-Speech (POS) tag. A noun is represented by NN, a proper noun by NNP. The representation used depends on the parser used. The POS tags are all children of a group which can be a noun phrase (NP) or a verb phrase (VP) among others. Given the parsed input sentence, as shown in Fig. 6, we would like to write a pattern which matches this sentence.

The flexibility of Tregex was described earlier. We can write a very narrow focused pattern to only match the exact sentence above using the example shown in Fig. 7.

Match Pattern:	Emmanuel-Joseph .. Sieyès .. trained .. as .. a .. priest	
Question Template:	<input type="text"/>	Answer Template: <input type="text"/>

Figure 7. A match pattern with low coverage due to narrow expression terms.

This pattern contains exact words which must be matched and the double dots indicate that the words must be in sequence (but not necessarily immediately preceding one another). It is a valid pattern and will match our target sentence, but the coverage is low. The pattern will only ever match very specific sentences which contain the words from the pattern in the correct order. It would be better if this rule could cover not just *Emmanuel-Joseph Sieyès* but any priest. It would be even better if the rule was not restricted to just priests!

Referring back to the syntax tree in Fig. 6, we can use the groups and POS tags to give our rule a bit more coverage. The rule is rewritten to match a noun phrase (NP), followed by a verb in the past participle (VBN), the words ‘as a’ and a noun (NN). In Fig. 6 one can see that two noun phrases are nested on the left most branch containing the persons name. To ensure we match the lowest noun phrase in the branch, we use the syntax ‘!< NP’ meaning does not have a child which is a noun phrase.

At this point we will also mark those parts of the match which we are interested in for generating our question, the initial noun phrase and the verb. We need to use those to generate the question and answer.

Match Pattern:	NP=g1 !< NP . (VBN=g2 . (as . (a . NN=g3 )))	
Question Template:	What did /1 /2->VPAST as ?	Answer Template: /3

Figure 8. A pattern with high coverage through use of broader expression terms.

The pattern in Fig. 8 has a much wider coverage. It also marks the matched parts using the syntax ‘=gX’ where the value X is then referenced in the question template. The question template uses the syntax ‘/X’ to indicate where the matched parts are inserted to generate the question.

One NLP tool which can be used to change verb tenses is a morphology database. The advantage of such a tool is that each word has been manually entered into the database and therefore it is very accurate. The disadvantage is that new words must be added to the database over time. The Xtag morphology database [9] was ported to Java to provide this feature in Ceist. A sample line from the flat file version of the morphology database contains a keyword followed by other forms of that keyword:

```
trained          train  V PAST WK#train      V PPART WK
```

The database is queried by sending a keyword and a command. The XTAG program then finds the entry in the database for that keyword and looks up the word for which the inflected form is the given command. For example, the VPAST WK of ‘train’ is ‘trained’. If we send the query ‘trained’ and the command ‘VPAST’, the database returns ‘train’. This allows us to change the form of the verb when we generate the question by using the ‘->VPAST’ modifier.

Higher coverage can, however, also introduce problems: whenever the coverage of a pattern is widened the risk of matching unwanted sentences increases. This can be seen when the rule above is applied to the entire article on the French Revolution; the output for this is shown in Fig. 9. Another feature to aid rule editing with Ceist is the rule output. This updates instantly when the match pattern is edited; note how Ceist also marks those parts of the target sentence which will be used in the question and answer templates with superscripted integers and colouring.

Full Sentence	Question
<sup>1</sup> Emmanuel-Joseph Sieyès -LRB- 1748 -- 1836 -RRB- <sup>2</sup> trained as a <sup>3</sup> priest and became assistant to a bishop .	What did Emmanuel-Joseph Sieyès train as?
<sup>1</sup> Its commander was the liberal-minded Marquis de Lafayette -LRB- 1757 -- 1834 -RRB- , who had <sup>2</sup> fought as a	What did Its commander fight as?
<sup>1</sup> Plate 1 -LRB- see page 11 -RRB- shows an actor <sup>2</sup> dressed as a <sup>3</sup> sans-culotte , carrying the tricolor banner	What did Plate 1 dress as?

Figure 9. Results display showing all matched sentences and generated questions.

The pattern matches three sentences from the OpenLearn article. The three sentences which were matched cannot be seen completely in Fig. 9, but they are in full:

1. Emmanuel-Joseph Sieyès (1748–1836) trained as a priest and became assistant to a bishop.
2. Its commander was the liberal-minded Marquis de Lafayette (1757–1834), who had fought as a volunteer with the American revolutionaries.
3. Plate 1 (see page 11) shows an actor dressed as a *sans-culotte*, carrying the tricolour banner (on which is emblazoned the slogan liberty or death’) at the ‘festival of liberty’ in Savoy in October 1792.

Now that the coverage has been broadened, the pattern author must decide which of these sentences they are attempting to target. The matches might produce valid questions, but if a pattern is intended to produce other questions such as ‘What was the occupation of Emmanuel-Joseph Sieyès’, then it might be better to refine the pattern further. To achieve this the rule author must refine the rule.

One distinguishing factor evident from the results is that the first matching part is a person’s name for our target sentence, but not for the others. Another is the fact that the subject of the target sentence is the noun phrase immediately preceding the verb. This is important because it is quite possible that a sentence very similar to sentence 3 would be incorrectly matched if the first noun phrase were a persons name.

Currently, rules are manually refined by modifying the pattern to exclude irrelevant sentences. This is a slow process but does result in improved rules over time.

The expression ‘NP <- NNP !<, DT’ matches a noun phrase which does not begin with a determiner and ends in a proper noun. This expression matches person names and is useful for matching sentences containing person names. Ceist provides the capability to use a definition for a person name which is then substituted for this expression whenever a pattern seeks to match a noun phrase which is a person name.

The ‘NP’ in the original pattern can be replaced with ‘personNP’ and the rule will then only match a noun phrase which is a person’s name. Ceist also allows this technique to be used for groups such as colours or the days of the week. One advantage of this technique is that the definition for personNP can be refined in one place, without needing to rewrite all rules containing personNP.

Tregex allows us to specify that a noun phrase must immediately precede the verb by using a single dot instead of the double dots we were using.

The refined pattern now looks like that in Fig. 10. When using groups such as ‘personNP’, the reference number does not use the syntax ‘=gX’, but instead the reference number is appended to the group name.

Match Pattern:	NP < personNP1 . (VBN=g2 . (as . (a . NN=g3 )))	
Question Template:	What did /1 /2->VPAST as?	Answer Template: a /3

Figure 10. Modified pattern which matches a persons name.

This new pattern eliminates the undesired results returning a single match and generating the question we originally targeted. Fig. 11 shows the final result.

Full Sentence	Question	Answer
<sup>1</sup> Emmanuel-Joseph Sieyès -LRB- 1748 -- 1836 -RRB- <sup>2</sup> trained as a <sup>3</sup> priest and became assistant to a bishop .	What did Emmanuel-Joseph Sieyès train as?	a priest

Figure 11. Result showing the desired match and generated question.

The approach used by Ceist is similar to that used by many existing Natural Language Generation (NLG) systems. Rules consisting of patterns and templates are defined, with the template reusing some of the words from the input that matched the pattern. The manner in which the rules are represented does vary however.

Ceist stores its rules as XML and uses a format similar to the QuALiM Question Answering system [6]. Matched parts of the sentence are marked as integers and the templates reference these integers.

Cai et al. also use a format which they present as a mark up language, NLGML [5]. NLGML marks matched parts of the original sentence as variable names for use in the templates and where Ceist replaces semantic features such as a persons name with a new match type, NLGML uses attributes within the noun phrase's XML tags.

### 3. Conclusions and Further Work

This paper introduced OpenLearn as a suitable data resource for Question Generation and described the Ceist Question Generation system. Ceist uses a rule-based approach based on pattern-template combinations and provides several facilities for making the rule authoring process more user-friendly.

We would like to conclude by proposing that one way to advance the state-of-the-art in QG is to initiate an effort to arrive at a standard format for the representation of rules. Ideally rules should be shareable and possibly even come with information on their coverage and error rates. The main problem to overcome will be to make sure that a common format does not commit the QG community to a restricted set of NLP tools.

### References

- [1] D. Jurafsky and J.H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics*, Prentice-Hall, New Jersey, 2008.
- [2] A. Graesser, J. Otero, A. Corbett, D. Flickinger, A. Joshi and L. Vanderwende, Chapter 1: Guidelines For Question Generation Shared Task Evaluation Campaigns, In V. Rus and A.C. Graesser (Eds.) *The Question Generation Shared Task and Evaluation Challenge* (2009), <http://www.questiongeneration.org>.
- [3] D. Gates, Generating Look-Back Strategy Questions from Expository Texts, *1<sup>st</sup> Workshop on the Question Generation Shared Task and Evaluation Challenge*, NSF, Arlington, VA (2008).
- [4] W. Wang, T. Hao and W. Liu, Automatic Question Generation for Learning Evaluation in Medicine, *Advances in Web Based Learning – ICWL 2007* (2008), 242-251.
- [5] Z. Cai, V. Rus, H.J. Kim, S.C. Susarla, P. Karnam and A.C. Graesser, NLGML: A Markup Language for Question Generation, *Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education* (2006), 2747-2752.
- [6] M. Kaisser and T. Becker, Question Answering by Searching Large Corpora with Linguistic Methods, *Proceedings of the 2004 Edition of the Text REtrieval Conference*, Gaithersburg, Maryland (2004).
- [7] R. Levy and G. Andrew, Tregex and Tsurgeon: tools for querying and manipulating tree data structures, *Proceedings of the 5<sup>th</sup> International Conference on Language Resources and Evaluation (LREC 2006)*, Genoa, Italy, (2006).
- [8] D. Klein and C.D. Manning, Accurate Unlexicalized Parsing, *Proceedings of the 41<sup>st</sup> Meeting of the Association for Computational Linguistics*, Sapporo, Japan, (2003), 423-430.
- [9] C. Doran, D. Egedi, B.A. Hockey, B. Srinivas and M. Zaidel, XTAG System – A Wide Coverage Grammar for English, *Proceedings of the 15<sup>th</sup> International Conference on Computational Linguistics*, Kyoto, Japan, (1994), 922-928.