Quinductor: a multilingual data-driven method for generating reading-comprehension questions using Universal Dependencies

Dmytro Kalpakchi*
Division of Speech, Music and Hearing,
KTH Royal Institute of Technology,
Stockholm, Sweden

Johan Boye
Division of Speech, Music and Hearing,
KTH Royal Institute of Technology,
Stockholm, Sweden

We propose a multilingual data-driven method for generating reading comprehension questions using dependency trees. Our method provides a strong, mostly deterministic, and inexpensive-to-train baseline for less-resourced languages. While a language-specific corpus is still required, its size is nowhere near those required by modern neural question generation (QG) architectures. Our method surpasses QG baselines previously reported in the literature and shows a good performance in terms of human evaluation.

1 Introduction

We are interested in **question generation** (QG) – the task of automatically generating reading comprehension questions and their correct answers from given declarative sentences. Numerous methods have been proposed for solving this task, most of which have been aimed at the English language. Recent methods are based on neural networks and rely on the availability of large-scale datasets, such as SQuAD (Rajpurkar et al. 2016) – a question-answering dataset repurposed for QG – or large-scale pretrained models, such as GPT-3 (Brown et al. 2020). Early methods, mostly based on context-free grammars, relied on the strict word order and the limited inflectional morphology of English. These traits made it relatively straightforward to craft hand-written templates based on these grammars. The above mentioned idiosyncracies and the unique availability of large-scale resources for English leave a number of open challenges for developing QG methods applicable to languages other than English.

The first challenge is the lack of large-scale training datasets, and a prohibitively high cost of obtaining such resources. State-of-the-art QG methods for English train their models on the previously mentioned SQuAD dataset, which contains more than 100,000 questions. Obtaining a good-quality dataset of a similar size is very expensive, especially for languages with fewer native speakers around the world.

The second challenge is knowing how well available methods developed for English would generalize to other languages, especially synthetic ones with richer inflectional morphology and less strict word order (e.g., Finnish, Turkish or Russian). To the best of our knowledge, not much research has been done on QG for these kinds of languages.

The third challenge is assessing the obtained performance results. Evaluation results in isolation do not provide a comprehensive picture of the method's performance, especially when using only automatic evaluation metrics, such as BLEU (Papineni et al. 2002). Researchers that developed the first statistical QG methods for English could compare their results to baselines

* E-mail: dmytroka@kth.se

that relied on context-free grammars. However, most other languages lack QG baselines, leaving researchers to wonder if the obtained performance is worth the spent computational resources on training the model.

In this article we are addressing all three challenges by proposing a novel, mostly deterministic method, called **Quinductor**[1] (Question inductor), for automatically generating question-answer pairs from data. Quinductor is based on dependency trees and can also be used for languages other than English, due to the Universal Dependencies (UD) framework (Nivre et al. 2020) offering more than 200 treebanks in 100 languages. The method does require a language-specific QA dataset, but its size can be orders of magnitude smaller than SQuAD. Hence we believe that Quinductor can serve as a strong QG baseline for less-resourced languages.

2 Related work

Rus, Cai, and Graesser (2008) broadly defined QG as automatic generation of questions from inputs such as text, raw data or knowledge bases. In this article, we are interested in generating reading comprehension questions from textual data, with their respective correct answers, and we want to do this in multiple languages. We exclude Yes/No-questions and fill-in-the-blank questions, as those can be generated with less sophisticated methods (Gates 2011; Mostow and Jang 2012; Agarwal, Shah, and Mannem 2011). Hence we limit the scope of related works only to articles exploring a similar QG setup.

To the best of our knowledge, no other work has proposed an automatic multilingual QG method relying on dependency parsing. The closest by spirit is the work by Afzal and Mitkov (2014), where sentences are matched against a set of automatically extracted semantic patterns from the GENIA Event Annotation corpus using a Named Entity Recognizer (NER). These patterns are used to extract relevant parts of the dependency tree, which are then transformed into the question by abstracting away the information constituting the correct answer (which should not be a part of the question). The method requires resources that are often lacking for other languages, such as, a NER system and a corpus which would be very expensive to annotate.

Mazidi and Nielsen (2015) also relied on a dependency parser, a semantic role labeler and discourse cues. However, their method only generated questions without the correct answers, requiring the manual creation of question templates, and relying on language-specific information. Similarly, Khullar et al. (2018) proposed a method using a dependency parser and three manually crafted rule sets for transforming statements into questions (without exploring the generation of correct answers).

Other non-neural QG methods utilised hand-written templates based on context-free grammars. One example is the work by Heilman and Smith (2009), which used an overgenerate-and-rank strategy for QG without generating correct answers. Another example is the work by Bernhard et al. (2012), which is based on constituent trees and a NER system to generate questions (and their correct answers) in French. Such methods require linguists to create context-free grammars, which is an expensive process, especially for languages with less strict word order and a richer morphology than English.

The most recent QG methods are based on neural networks, and thus require both large-scale datasets in the language of interest, as well as vast computational resources to train the models. Impressive performance for English have been demonstrated by both Transformer-based masked language models (Chan and Fan 2019; Liao, Jiang, and Liu 2020; Dong et al. 2019) and auto-regressive models based on encoder-decoder architectures (Kim et al. 2019; Liu et al. 2019; Du, Shao, and Cardie 2017; Song et al. 2018; Zhao et al. 2018; Bahuleyan et al. 2017). Note that

1 The code is available at `https://github.com/dkalpakchi/quinductor`

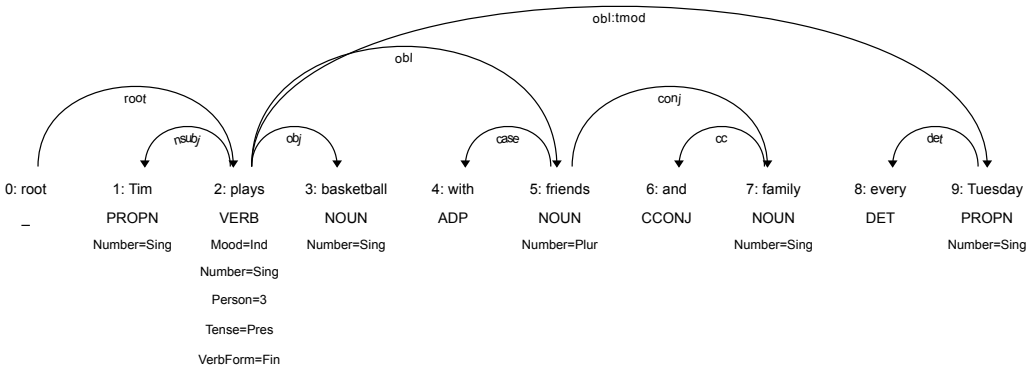


Figure 1
The dependency tree for the sentence "Tim plays basketball with friends and family every Tuesday"

neural models typically do not generate correct answers, but instead use them as an input along with the sentence to generate questions. However, we are not going into more details on the neural methods, as our proposed method is not neural.

To the best of our knowledge, only a very limited number of neural methods explore QG in other languages than English, or multilingual QG. One such example is the work by Kumar et al. (2019) exploring joint cross-lingual training aimed at reusing the large-scale SQuAD dataset for Hindi and Chinese.

3 Methodology

Let D be a dataset consisting of triples (c_i, q_i, a_i), where c_i is a context (a text passage), q_i is a question created based on c_i, and a_i is a contiguous phrase in c_i, answering q_i. A pair of (q_i, a_i) will be referred to as a question-answer pair (QA-pair). The aim is then to be able to generate QA-pairs (q'_j, a'_j) given a previously unseen context c'_j.

Our method, Quinductor, automatically induces QA-templates from the dataset D using dependency parsing based on the UD framework. More formally, let s_i be the sentence from the context c_i in which the answer a_i appears (s_i can be found using a sentence segmenter, a tokenizer, and simple string matching). The QA-pair (q_i, a_i) is recast into a template in a specific formal language (defined in Section 3.1), using parts of the dependency tree for the sentence s_i. For instance, suppose s_i is *"Tim plays basketball with friends and family every Tuesday"* (with its dependency tree shown in Figure 1), and q_i is "When does Tim play basketball with friends and family?". Assuming r represents the root of the dependency tree (i.e. the dependent of the "root" pseudonode; the word "plays" in this example), q_i can now be expressed using the question template (1) and the answer "every Tuesday" could be expressed using the answer template (2). For a formal definition of these expressions we refer to the Section 3.1.

(1) `When does [r.nsubj#1] [r.lemma] [r.obj#3] <r.obl#5>?`

(2) `<r.obl:tmod#9>`

Such a transformation can be applied only if certain conditions are met, and therefore each QA-template has an associated **guard** (described in its own formal language defined in section 3.2). After inducing both QA-templates and associated guards, we can then apply them to any previously unseen context c' by processing its every sentence s' using the following procedure.

Step 1: Perform dependency parsing on s' and get a dependency tree T'.

Step 2: Find all satisfied guards for T' and get a set of corresponding QA-templates $QA_{T'}$.

Step 3: Apply all templates from $QA_{T'}$ to s', in order to get a set of generated question-answer pairs QA'. Note that many QA-pairs will be unsatisfactory, which is why the next step is introduced.

Step 4: Rank QA' so that a QA-pair (q', a') is ranked highly if it is likely to be relevant, grammatical, and where a' is likely to be the correct answer to q'. The ranking is done according to the method presented in Section 3.5.

As an example of this procedure, consider s' to be the sentence "Ericsson pays dividends to the shareholders every first quarter of the year" (with a dependency tree in Figure 2), then using the question template (1) and the answer template (2), the following QA-pair can be generated:

(3) When does Ericsson pay dividends to the shareholders? – Every first quarter of the year

The key to our method is to make templates as generic as possible to allow a certain amount of variation in their dependency structures. For instance, we want to avoid using adverbial clauses word by word, but instead matching adverbial clauses more generally. This generalization is addressed by **a novel shift-reduce algorithm**, described in Section 3.3. Automatic induction of guards is described in 3.4. However, before describing the induction algorithms, let us first explain and motivate the designed template and guard languages. In the sections below we use a bold font in the expression definitions to indicate metalinguistic variables which are not part of the defined languages.

3.1 Template language

Let T be an arbitrary dependency tree and `r` denote *the root of the dependency tree*, i.e. the dependent of the "root" pseudonode of T. In the example sentence in Figure 1 `r` corresponds to the word "play" (all further examples will also be given for this sentence). Let `n` be an arbitrary node of T, then the following definitions are introduced.

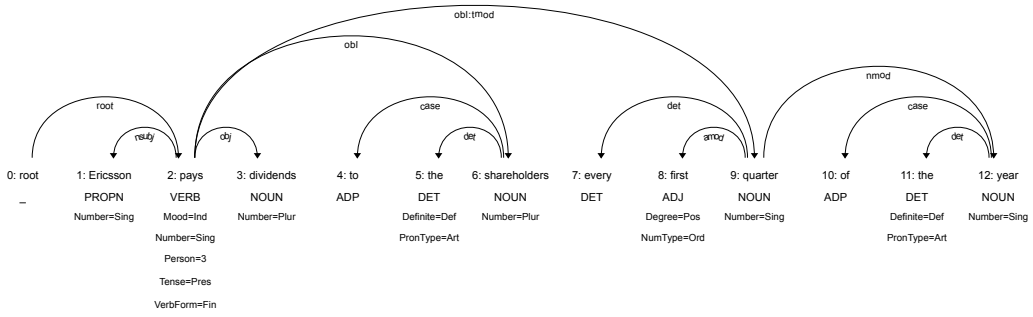


Figure 2
The dependency tree for the sentence "Ericsson pays dividends to the shareholders every first quarter of the year"

- `n.rel#id` denotes a dependent of `n` with a dependency relation **rel** and index **id** of this dependent of `n` (starting from 0 for the "root" pseudonode). For instance, `r.obj#3` denotes the node for the word "basketball".
- `n.rel1#id1.rel2#id2` ... .**relN#idN** denotes a node `n'` such that there exists a directed path between nodes `n` and `n'` with each edge having a corresponding dependency relation from a relation chain **rel1#id1**.**rel2#id2** ... .**relN#idN**. The node `n'` will be referred to as *a node at the end of the chain* and a whole relation chain will be shortened to **relchain**. For instance, `r.obl#5.case#4` denotes the node for the word "with". This node can then be referred as the node at the end of the chain `obl#5.case#4`. The ID of the last element of relchain is later referred to as the ID of a template expression.

The IDs are included in the template expressions above to be able to distinguish between different dependents having the same dependency relation. To illustrate when this could be necessary, imagine the dependency relation between the words "plays" and "Tuesday" is `obl` instead of `obl:tmod` (an inaccuracy that could be produced by the dependency parsers in practice, especially for languages other than English). Then the question "When does Tim play basketball with friends and family?" with the answer "every Tuesday" would result into the following QA-template (assuming IDs are excluded):

(4) `When does [r.nsubj] [w.lemma] <r.obl>?`

(5) `<r.obl>`

It is impossible to distinguish `<r.obl>` in the question template (4) from the one in the answer template (5). However if the IDs are introduced, then one immediately understands that those expressions correspond to different nodes. Note that differentiating between nodes with the same dependency relations is the only purpose of IDs, i.e. we do NOT require the new sentences using QA-templates to have exactly the same IDs, as it would obviously hinder generalization.

Let us define the following operators for selecting substructures from a dependency tree T:

- `[n]` extracts the token at the node `n` (for instance, `[r]` extracts the token "plays");
- `[n.relchain1]` extracts the token of the node at the end of **relchain1** (for instance, `[r.obl#5.conj#7]` extracts the token "family");
- `[n.lemma]` (`[n.relchain.lemma]`) extracts the lemma of the token at the node `n` (the node at the end of **relchain1**). Either of these will be referred to as a **lemma-expression**. For instance, `[r.lemma]` extracts the string "play" and `[r.obl#5.lemma]` extracts "friend".
- `<n>` extracts the text string of the subtree rooted at the node `n` (for instance, `<r>` extracts the whole sentence);
- `<n.relchain1>` extracts the text string of the subtree rooted at the node at the end of **relchain1** *preserving the linear order* (for instance, `<r.obl#5.conj#7>` extracts the string "and family");
- `<n.relchain1 - relchain2>` extracts the text string of the subtree rooted at the node at the end of **relchain1** *except the text string of the subtree rooted at the node at the end of* **relchain2** (if such a subtree exists). Relchains that are subtracted in any template expression will be referred to as **negatives**. For instance,

`<r.obl#5.conj#7 - cc#6>` extracts the string "family". However, non-existing negatives do not influence the result, hence `<r.obl#5.conj#7 - case#6>` extracts the string "and family", since there is no child of `r.obl#5.conj#7` with a dependency relation `case`.

- `<n.` **relchain1 - relchain2*** `>` extracts the text string of a subtree rooted at the node at the end of **relchain1** *except the contents of the node at the end of* **relchain2** (if it exists). For instance, `<r.obl#5 - conj#7*>` extracts the string "with friends and". Note that the extracted string is not guaranteed to be a contiguous substring of the sentence.

Template expressions surrounded by square brackets (`[]`) will be referred to as **node-level expressions**, and those surrounded by angle brackets (<>) as **subtree-level expressions**.

To distinguish the answer from the question, we use an additional binary infix operator `q => a`, denoting that the first operand is the question template, and the second one is the answer template. For instance, the question "When does Tim play basketball with friends and family?" with the answer "every Tuesday" could be represented as the following QA-template.

(6) `When does [r.nsubj#1] [r.lemma] [r.obj#3] <r.obl#5>? => <r.obl:tmod#9>`

Note that words from the question that do not appear in the original sentence will not form any template expressions. Instead, they will be considered **constant** and rendered as a plain text, e.g., "When" and "does" in the template (6).

3.2 Guard language

Recall that a **guard** is an expression specifying conditions for using a specific template. Formally, let T be an arbitrary dependency tree, and `n` denote a node in T. Then:

- `n.pos` denotes the part-of-speech (POS) tag assigned to the word associated with `n` (below referred to as a **pos-property**);
- `n.morph` denotes a set of morphological features, as defined by UD, associated with `n` (below referred to as a **morph-property**).

Each guard consists of clauses separated by a comma operator (`,`) denoting logical AND. Let us introduce operators defining the conditions for the guard clause to be satisfied:

- the unary operator `exists` can be applied exclusively to relchains in order to only accept sentences having a specified relchain;
- a binary operator `is` (`is_not`) can be applied merely to pos-properties to only accept sentences with a specific node having (lacking) a specified POS-tag;
- a binary operator `has` (`has_not`) can be applied to morph-properties exclusively to only accept sentences with a specific node having (lacking) specified morphological properties (in the UD format).

To specify which template should be used if all guard clauses are satisfied, we use an additional infix operator `guard -> t` denoting that if the first operand (guard) is satisfied, the template found by the unique identifier `t` can be used.

To exemplify, the guard for the template (6) could look as follows.

(7) `n.pos is VERB, n.nsubj exists, n.obj exists, n.obl exists, n.obl:tmod exists -> template3`

Note that *no requirement* on the exact IDs of the nodes is present in the guards, since the IDs are only used during the template induction phase.

After having described both template and guard languages, we are now ready to explain the algorithms for automatically inducing templates (Section 3.3) and guards (Section 3.4).

3.3 Template induction

Recall that a datapoint is a triple (c_i, q_i, a_i), where c_i is a context, q_i is a question asked on the basis of c_i, and a_i is a contiguous phrase in c_i constituting the correct answer to q_i. The goal is to induce templates for every $(q_i, a_i) \in D$, allowing to generalize to syntactically similar QA-pairs $(q'_j, a'_j) \notin D$. This is achieved by merging template expressions into subtree-level expressions as much as possible, using the novel shift-reduce algorithm described below.

The preprocessing step is to find all triples (s_i, q_i, a_i) such that a_i is a contiguous phrase in $s_i \in S(c_i)$, where $S(c_i)$ is the set of sentences of the context c_i. Recall that this step is trivially performed using a sentence segmenter, a tokenizer, and simple string matching.

The next step is to select only **satisfactory triples**, where s_i and q_i have at least one word in common (if not, then generalization is impossible). After obtaining a number of satisfactory triples (s_i, q_i, a_i), the induction of a template for transforming s_i into a pair of (q_i, a_i) can be described as the following 3-step process applied twice (once for q_i and once for a_i).

1. **Sentence transformation**. Describe every word of q_i (a_i) in terms of dependency structures present in s_i using the formal template language presented in Section 3.1. When finished, proceed to step 2.
2. **Shift-reduce**. Simplify the template obtained at the previous step using the novel shift-reduce algorithm described in Section 3.3.2. In the rare case when the resulting template consists only of a single template expression (and would therefore generalize poorly), return the sentence transformation from step 1 as the final template, otherwise, proceed to step 3.
3. **Merging negatives**. If possible, merge negatives (the subtracted relchains) in every template expression using the algorithm described in Section 3.3.3. Return the template with merged negatives as final.

Recall that words from the question that do not appear in the original sentence will be considered **constant** and rendered as a plain text. Templates containing only constants will not generalize and are thus removed after all templates have been induced. The remaining templates (i.e., with at least one non-constant template expression) are post-filtered to exclude templates with rare words (since those will not generalize well). We define a word as rare if it appeared in less than 25% of the documents and detect it based on the inverse document frequency (IDF), i.e. we exclude all templates with a maximal IDF among their constants exceeding $log(\frac{N}{\frac{N}{4}}) = log(4)$, where N is the number of documents in the corpus.

3.3.1 Sentence transformation

The goal of this step is to describe every word in q_i and a_i in terms of dependency structures of s_i. For instance, consider the QA-pair "When does Tim play basketball with friends? – Every Tuesday", created based on the example sentence (see Figure 1). Sentence transformation applied to the question would then look as follows:

(8) `When does [r.nsubj#1] [r.lemma] [r.obj#3] [r.obl#5.case#4] [r.obl#5]`

Whereas sentence transformation applied to the answer would take the following form:

(9) `[r.obl:tmod#9.det#8] [r.obl:tmod#9]`

To perform sentence transformation, first, both s_i and q_i should be parsed to get the dependency trees T_{s_i} and T_{q_i} respectively. T_{q_i} is then traversed in linear order $L_{T_{q_i}}$, skipping the question word, which is assumed to be the first word from the beginning or the end of the sentence, depending on the language of interest. For each node n_q in $L_{T_{q_i}}$, the algorithm attempts to find a matching node in T_{s_i} with the same token. If no matching nodes are found, n_q is replaced by its token. If matching nodes are found, n_q is replaced by the list of template expressions corresponding to those nodes in T_{s_i} (using the template language presented in Section 3.1). This list is sorted in the ascending order by the distance from the root node of s_i in edges. The resulting list of lists of template expression will be referred to as **LLTE**. Note that generation of lemma-expressions (e.g., `[r.lemma]`) is subject to the availability of a lemmatizer, in the absence of which the algorithm will simply insert a constant expression (i.e., the token itself).

A template will generalize if many syntactic structures can be merged into subtree-level expressions, which is the goal of the shift-reduce step of Quinductor. Hence, the result of sentence transformation should contain as many long contiguous phrases as possible. With this goal in mind, after all nodes in $L_{T_{q_i}}$ have been processed, the combination of template expressions with longest contiguous spans is selected. This can be achieved by finding template(s) with the smallest sum of absolute ID differences between every two neighboring template expressions.

For instance, consider the sentence "The longest river in Brazil is the Amazon river" with a dependency tree shown in Figure 4. Assume that the question in the dataset which is based on this sentence is "What is the longest river in Brazil?". The LLTE for this question is shown in Figure 3, assuming `w` refers to the root of the original sentence, namely the word "river".

What	is	the	longest	river	in	Brazil
What	`w.co#6`	`w.de#7` `w.ns.de#1`	`w.ns.am#2`	`w#9` `w.ns#3`	`w.ns.nm.ca#4`	`w.ns.nm#5`

Figure 3
LLTE for the question "What is the longest river in Brazil?". Each column represents all available alternatives for the given word. For the sake of brevity, the `[·]` operator is omitted, since all template expressions are node-level, the first two letters of dependency relations are used, and only the IDs of the last dependency relations in relchains are specified.

As we can see, only the words "the" and "river" have multiple possible representations (i.e., `w.det#7` and `w.nsubj#3.det#1` for "the" and `w#9` and `w.nsubj#3` for "river"). The sums of absolute ID differences for different combinations of representations for the words "the" and "river" are presented in Table 1. The first representation from LLTE with the minimal sum of absolute ID differences is `[w.det#7]` for "the" and `[w.nsubj#3]` for "river", resulting in the following sentence transformation

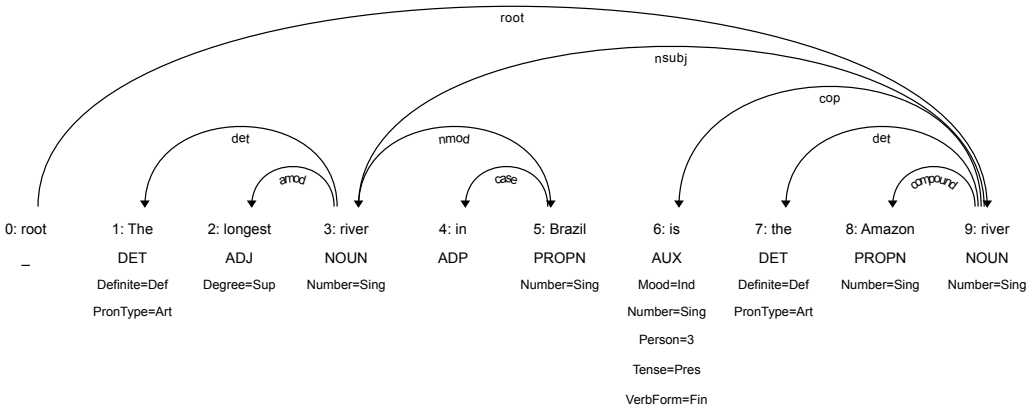


Figure 4
The dependency tree for the sentence “The longest river in Brazil is the Amazon river”

(10) What `[w.cop#6] [w.det#7] [w.nsubj#3.amod#2] [w.nsubj#3] [w.nsubj#3.nmod#5.case#4] [w.nsubj#3.nmod#5]`

As can be seen, the algorithm chose the right expression for the word “river” and the wrong one for the word “the”. Such errors depend on the order of lists in LLTE and there’s no universal order that will result in choosing the right expressions all the time for all the languages.

3.3.2 Shift-reduce

The goal of this step is to make templates generalizable, which is achieved by merging template expressions into subtree-level expressions as much as possible using a novel shift-reduce algorithm. At every algorithm step, a current template (starting with the template obtained after the sentence transformation) is divided into *a LIFO stack*, where all seen items reside, and *a FIFO buffer*, containing the remainder. Depending on the stack-buffer configuration, one of the following two actions can be chosen:

- **SHIFT**, that removes the top expression from the buffer and adds it to the stack.
- **REDUCE**, that checks the topmost and the second topmost expressions on the stack and merges them into a subtree-level expression.

Table 1
Sums of absolute ID differences for alternative representations of the words “the” and “river” for LLTE in Figure 3

Representation of “the”	Representation of “river”	Sum of absolute ID differences
`[w.det#7]`	`[w#9]`	19
`[w.det#7]`	`[w.nsubj#3]`	9
`[w.nsubj#3.det#1]`	`[w#9]`	19
`[w.nsubj#3.det#1]`	`[w.nsubj#3]`	9

While SHIFT action is self-explanatory, REDUCE can be described as a 3-step procedure, operating on the topmost (`stackTop`) and the second topmost (`stackTop2`) template expressions on the stack.

Step 1: Extract relchains from `stackTop` and `stackTop2` and then find the longest common prefix for them, later referred to as the **common relchain**. The node at the end of the common relchain is the closest common ancestor of the nodes corresponding to `stackTop` and `stackTop2`.

Step 2: The second step is to ensure that the two **merging conditions** are satisfied:

1. the common relchain is not empty;
2. the common relchain differs from either relchain of `stackTop` or `stackTop2` by at most one dependency relation.

We have empirically found that these merging conditions increase chances of generalization.

Step 3: If the aforementioned conditions are met, `stackTop` and `stackTop2` can be replaced by the template expression of their common relchain with a number of subtracted negatives corresponding to all tokens of the induced subtree except those necessary to keep: `stackTop`, `stackTop2`, any node from the sentence transformation, and any whole subtree containing any of these nodes.

To illustrate the algorithm, consider turning the phrase "friends and family" from the sentence in Figure 1 into a template. Initially the stack is empty and the buffer contains the sentence transformation of the phrase, resulting in the configuration shown in Figure 5.

Stack	Buffer
	[r.obl#5] [r.obl.#5.conj#7.cc#6] [r.obl#5.conj#7]

Figure 5
Initial stack-buffer configuration for the shift-reduce algorithm applied on the sentence transformation of the phrase "friends and family" from the sentence in Figure 1.

First, two SHIFTs are required to ensure that the stack has at least two expressions, leading to the configuration in Figure 6.

Stack	Buffer
[r.obl.#5.conj#7.cc#6] [r.obl#5]	[r.obl#5.conj#7]

Figure 6
Stack-buffer configuration after 2 SHIFT actions.

The top two template expressions on the stack are neither constants nor lemma-expressions, so the REDUCE action can be invoked. The common prefix for relchains is `obl#5`, meaning both merging conditions are satisfied and the expressions can be merged into `<r.obl#5 - conj#7*>`. This new expression replaces the top two expressions on the stack, resulting in the configuration in Figure 7.

Stack	Buffer
<r.obl#5 - conj#7*>	[r.obl#5.conj#7]

Figure 7
Stack-buffer configuration after 2 SHIFT and 1 REDUCE actions

The next step is SHIFTing the last expression of the buffer into the stack, resulting in the configuration in Figure 8.

Stack	Buffer
[r.obl#5.conj#7] <r.obl#5 - conj#7*>	

Figure 8
Stack-buffer configuration after 3 SHIFT and 2 REDUCE actions

The top two template expressions on the stack are neither constants nor lemma-expressions, so the REDUCE action can be invoked again. Following the same logic as before, the expressions can be merged into <r.obl#5>, resulting in the configuration in Figure 9.

Stack	Buffer
<r.obl#5>	

Figure 9
Stack-buffer configuration after 3 SHIFTs and REDUCE

The buffer is empty, which means shift-reduce is finished. The final template is on the stack.

3.3.3 Merging negatives

Recall that negatives are relchains subtracted in any template expression. The goal of this step is to merge negatives in the resulting template after the shift-reduce step, in order to make templates even more generic and generalizable. For instance, sentence transformation (8) would be converted to the following template after shift-reduce:

(11) `When does [r.nsubj#1] [r.lemma] [r.obj#3] <r.obl#5 - conj#7.cc#6 - conj#7*>`

The downside of this template is that it presupposes that by subtracting `conj#7.cc#6` and `conj#7*` it effectively removes the whole subtree corresponding to `conj#7`. However, conjuncts vary in structure and thus this template will generalize poorly to sentences with syntactically similar structures. To avoid this, the negatives of each template expression should be merged as much as possible to their common parent.

To perform this step, create a mapping between each node and its direct children. Then, for every template expression, check if any subset of negatives matches any set of children from the mapping. In case of a match, swap all matched negatives for the corresponding subtree root. After these steps expression (11) transforms into:

(12) `When does [r.nsubj#1] [r.lemma] [r.obj#3] <r.obl#5 - conj#7>`

3.4 Guard induction

A template can have multiple guards. Consider two sentences "John is playing basketball" and "John has played basketball". Both questions "What is John playing?" and "What has John played?" would result in the same template (supported by the previously mentioned sentences). However, the morphological properties of the root of each sentence ("playing" and "played" respectively) are different. Hence, there are 2 different cases when this template could be applied, and thus 2 guards.

Motivated by the example above, guards consist of a base guard and complementary guards. **A base guard** contains the requirements for using templates and its creation involves the following 3 steps:

1. Create an `exists`-clause for the relchain of every template expression present in the question or answer (excluding the negatives).
2. If a template for the answer contains a nominal subject (`nsubj`) as a non-negative expression, add the clause `n.nsubj.morph has_not PronType=Rel` ensuring that the subject is not a relative pronoun (e.g., "which"). This is motivated by the fact that no reading comprehension question would ask about a relative pronoun.
3. If a root is involved in the creation of a template, and it is a verb without an auxiliary verb, add a clause `n.aux not_exists`. The rationale behind this step is to separate templates for questions with either copula or tenses requiring a modal verb, from those questions that do not exhibit these features.

Complementary guards contain requirements specific to the sentences supporting the generated template. Complementary guards are induced by creating an `is`-clause for the pos-property and `has`-clause for the morph-property of the root of every sentence from the corpus supporting the current template.

To get a final set of guards for the template, add the base guard to each complementary guard and use an infix operator `->` to point each guard from the induced set to the template of interest.

For instance, the guard for the template (12) would look as (13), where `props` equals to `Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin`, and `n.morph has props, n.pos is VERB` is the only auxiliary guard.

(13) `n.morph has props, n.pos is VERB, n.nsubj exists, n.obj exists, n.obl exists, n.nsubj.morph has_not PronType=Rel, n.aux not_exists -> template7`

3.5 Ranking and filtering

After all templates and guards have been generated, they can be applied to unseen data to produce a number of QA-pairs. With the purpose of down-voting undesirable QA-pairs we use the following two models, that serve as a proxy to grammaticality of the questions for ranking and filtering.

1. An n**-gram model** based on any pos-morph-tagged UD-compliant corpus. For instance, we use a 3-gram model, which could give the following probability $P(\texttt{NOUN/Number=Sing}|\texttt{DET/Definite=Def}, \texttt{ADJ/Number=Sing})$.

2. A **question-word model** calculating the count $c(qw, r)$ for each pair of question word qw and pos-morph expression of the root r of the corresponding answer, e.g., `(when,NOUN/Number=Sing)`.

Both models operate on *pos-morph expressions* (i.e. words are substituted by their POS-tag together with UD morphological features, if applicable). For instance, the pos-morph expression for the word "basketball" is `NOUN/Number=Sing` and for the word "on" is `ADP`.

The first step is to filter out QA pairs with a single-word answer, whose pos-morph expression has never occurred in the training corpus as an answer. This step prevents generating single-word answers containing only function words (e.g., "the", "himself", "to").

The second step is to rank every remaining QA pair j according to the score $r_{qa}^{(j)}$ using equations (1) - (3). Equation (1) is a convex combination using the n-gram model, where p^* denotes the backoff probability, in which each word w_i is substituted by its pos-morph expression (or by a POS-tag only if the former does not exist). N_j is the number of trigrams in the question q_j, since we use a 3-gram model.

$$r_{ng}^{(j)} = \frac{1}{N_j} \sum_{i=1}^{N_j} \lambda_1 p^*(w_i|w_{i-2}, w_{i-1}) + \lambda_2 p^*(w_i|w_{i-1}) + \lambda_3 p^*(w_i) + \lambda_4 \tag{1}$$

Equation (2) uses the question-word model to get the frequency of the pair of the question word qw and the root token of the answer r_{a_j}.

$$r_{qw}^{(j)} = \frac{c(qw, r_{a_j})}{\sum_i c(qw, r_i)} \tag{2}$$

Equation (3) provides the final score, which is a linear combination between the scores calculated in Equations (1) and (2). α is a constant, which we set to 0.8 in our experiments.

$$r_{qa}^{(j)} = \alpha \cdot r_{ng}^{(j)} + (1 - \alpha) \cdot r_{qw}^{(j)}; \tag{3}$$

Using the aforementioned n-gram and question-word models, a generated QA-pair is given a high score r_{qa} if the question is made of a likely sequence of pos-morph expressions, and the question word (e.g., "when") matches the answer well.

The final step is referred to as **mean filtering**. This step ensures that only questions scoring higher than the mean of the scores for all generated questions for all sentences will be returned. Such filtering allows Quinductor to drop generated questions of potentially poor quality, and thus sometimes choose not to generate any QA-pairs for a given sentence.

4 Data

To evaluate Quinductor in a multilingual setting we have utilized a dataset called TyDi QA (Clark et al. 2020). The dataset is a question-answering benchmark based on Wikipedia articles for 11 typologically diverse languages. 8 of these languages have available UD treebanks and trained dependency parsers in Stanza package (Qi et al. 2020), which we have utilized for inducing templates in all languages. For both training and evaluation we have excluded Yes/No-questions resulting in the training/development sets of the sizes reported in Table 2. Due to limited resources, we have performed a human evaluation only on a subset of languages, while reporting automatic evaluation metrics for all languages with the available UD treebanks.

Table 2
Language-specific information along with the sizes of training and development splits (in the number of QA-pairs along with a proportion of the original training set) and the associated UD treebanks (UDT size, in tokens) used by the pre-trained Stanza parsers for the languages in the TyDi QA dataset. Question phrase positions are either obligatorily initial (OI), or not OI, or mixed, as defined by Dryer (2005).

Language	QP position	Training set	Dev. set	UDT size	Human eval.
Finnish (fi)	OI	7132 (47%)	1129 (52%)	397K	✔
Russian (ru)	OI	6425 (50%)	902 (56%)	1289K	✔
English (en)	OI	3837 (42%)	644 (62%)	648K	✔
Japanese (ja)	Not OI	4506 (28%)	705 (41%)	1676K	✘
Telugu (te)	Not OI	5680 (23%)	724 (29%)	6K	✘
Arabic (ar)	Not OI[2]	14771 (64%)	1016 (74%)	1042K	✘
Indonesian (id)	Mixed	5587 (37%)	728 (40%)	169K	✘
Korean (ko)	Not OI	1638 (15%)	427 (25%)	446K	✘
Bengali (bn)	Not OI	2506 (23%)	129 (39%)	NA	NA
Thai (th)	Not OI	4150 (37%)	1161 (52%)	NA	NA
Swahili (sw)	Not OI	2372 (16%)	661 (29%)	NA	NA

To compare Quinductor to previous work we have also used the SQuAD dataset (Rajpurkar et al. 2016) for English, specifically the training/validation/test split provided by Du, Shao, and Cardie (2017).

5 Evaluation

Essentially, automatic evaluation metrics, such as BLEU (Papineni et al. 2002), ROUGE (Lin 2004), METEOR (Agarwal and Lavie 2008), CIDEr (Vedantam, Lawrence Zitnick, and Parikh 2015), rely on comparing word overlap between a generated question and a reference question. Such metrics can yield a low score even if the generated question is valid but just happens to be different from the reference question, or a high score even though the question is ungrammatical but happens to have a high word overlap with the reference question (see the article by Callison-Burch, Osborne, and Koehn (2006) for a further discussion). Nonetheless, Amidei, Piwek, and Willis (2018) report that 32% of the surveyed papers on automatic QG used only automatic evaluation metrics, although Nema and Khapra (2018) found that there is only a weak correlation between the automatic evaluation metrics and human judgements on answerability of the generated questions. For a broader discussion on the relationship between automatic evaluation metrics and human judgements an interested reader is referred to (Gatt and Krahmer 2018, Section 7.4.1). In this article, we report automatic evaluation metrics for the following two reasons: Firstly, for the sake of comparability to other results reported in the literature, and giving a point of reference to researchers lacking resources to conduct human evaluations. Secondly, to assess the degree of the word overlap of the questions generated by Quinductor and the reference questions, as well as the quality of this overlap (e.g., if it contains mostly stop words).

The conducted human evaluation aims at providing insights about strengths and weaknesses of Quinductor as well as directions for future research. Unfortunately, there exist no standardized

2 The exception is Syrian Arabic, in which the interrogative phrase is obligatorily initial.

questionnaires and/or guidelines for human evaluation of automatically generated questions and answers. Amidei, Piwek, and Willis (2018) report 22 different criteria used by researchers to evaluate QG systems. Evaluations differ both on the number of criteria used and on the granularity of the rating scales for human judgements, see (Amidei, Piwek, and Willis 2018, Table 7) for more details. The number of human judges ranges from 1 to 364 (with an average of 4 and a mode of 2) and the number of sampled questions to be evaluated ranges from 60 to 2186 (with an average of 493). Amidei, Piwek, and Willis (2018) note that often the papers provide only little information about the evaluation guidelines as well.

For this article, we have tried to combine best practices from the reported evaluation guidelines for QG, notably (Heilman and Smith 2009; Rus et al. 2010), and more generally, best practices in human evaluation for NLG, as consolidated by van der Lee et al. (2020). On that basis, we propose to conduct human evaluation using a 9-item questionnaire. Each questionnaire item is rated on a 4-point Likert-type scale (see more information and design motivation in Appendix A).

A subset of automatic evaluation metrics (BLEU-N, ROUGE-L and CIDEr) were calculated using the nlg-eval package (Sharma et al. 2017), and METEOR using METEOR-1.5 package[3] (Denkowski and Lavie 2014). For all experiments we have used dependency parsers trained on UD treebanks as a part of Stanza package (Qi et al. 2020). All templates were induced and then processed using UDon2 (Kalpakchi and Boye 2020) – an efficient package for manipulating dependency trees, written in C++ with Python bindings.

5.1 Multilingual setting

In order to support the claim about Quinductor's applicability to multiple languages, we have performed an evaluation on the TyDi QA dataset (see more information about the dataset in Section 4). Different languages required somewhat different pre-processing steps, which are documented in Appendix B.

5.1.1 Automatic evaluation

We have evaluated Quinductor on all languages present in the TyDi QA dataset with available UD treebanks and pre-trained dependency parsers in Stanza. The templates were induced using the training sets and the questions were generated on the development sets of the TyDi QA dataset. Only the top-ranked generated question (if any) was considered for automatic evaluation with the respective automatic evaluation metrics reported in Table 3.

Quinductor was able to induce templates for all of these languages, but failed to generate any questions on the development sets for Telugu and Korean. The main reason is that QA-pairs for these languages contain answers that use smaller parts of some words (dubbed *subwords*) in the original sentence. As can be seen in Table 4, such cases constitute 53% and 60% of the training QA-pairs for Telugu and Korean respectively, whereas the corresponding proportions for other languages are much lower. For instance, the word "이스라엘" ("Israel") is used as the answer for one of the QA-pairs in Korean, whereas the original sentence contains the word "이스라엘의" ("Israeli"). Given that the number of possible questions not using subwords in the provided answers is only 19%, and the dataset for Korean is the smallest (only 1638 QA-pairs), it is no surprise that Quinductor managed to generate only 9 templates. The same proportion for Telugu is 35%, resulting in a larger number of templates, but only 1 generated question. This can most

3 METEOR-1.5 does not fully support all languages used in TyDi QA dataset, so we set language to "other" for all languages other than English)

Table 3
Automatic evaluation on the filtered TyDi QA development sets only for generated questions ranked first.

Metric	fi	ja	te	ar	id	ko	ru	en
BLEU-1	18.25	25.12	0	14.23	17.55	0	30.23	20.23
BLEU-2	10.04	12.03	0	8.35	10.06	0	19.99	12.16
BLEU-3	5.81	5.25	0	4.87	6.12	0	14.47	7.57
BLEU-4	3.42	2.30	0	2.90	3.74	0	11.23	4.72
METEOR	11.75	12.03	0	13.12	11.67	0	19.02	12.46
ROUGE-L	21.69	32.54	0	24.69	22.43	0	32.61	27.55
CIDEr	7.0	22.29	0	22.70	26.51	0	63.69	21.35

probably be attributed to the small size of Telugu's treebank (only 6K tokens), which might result in a less generalizable dependency parser.

While it is no surprise that subwords are used in agglutinative languages (e.g., Telugu, Korean, Japanese, Finnish, Indonesian) or fusional languages (e.g., Arabic), such cases are more surprising for English and Russian. For these languages, the cases are due to differences in tokenization between the original sentences and the provided answers (which can happen, since Stanza's tokenizers are based on neural networks). For example, the answer "$102 million" was tokenized as "$102", "million" for the answer and as "$", "102", "million" in the original sentence.

Russian and Japanese are two best performing languages in terms of BLEU-1 scores, meaning the induced questions have the highest word overlap with the reference questions. However, while Russian performs the best in terms of BLEU-4 (4-grams overlap), Japanese performs the worst (the other agglutinative languages, Finnish and Indonesian, perform similarly to Japanese in terms of BLEU-4).

Performance in METEOR scores (which have been shown by Agarwal and Lavie (2008) to correlate with human judgements better than BLEU scores) is roughly similar between all languages except a considerably higher score for Russian. This shows that while 4-gram precision is lower for some languages, the number of aligned matches is comparable.

Performance in ROUGE-L scores varies significantly with Japanese and Russian performing on-par at the top of the list, while Indonesian and Finnish are at the bottom of the list. While

Table 4
Descriptive statistics of the TyDi QA training data for different languages, as well as templates and questions produced using this data. Recall that "satisfactory questions" have at least one word in common with the original sentence.

	fi	ja	te	ar	id	ko	ru	en
(1) Satisfactory questions	74%	96%	68%	90%	83%	44%	62%	91%
(2) Answer uses subwords	9%	10%	53%	14%	5%	60%	10%	5%
(1) but not (2)	68%	86%	35%	78%	79%	19%	56%	87%
Number of induced templates	496	25	48	1104	340	9	85	254
Number of generated questions	611	97	4	462	558	0	93	409

this clearly indicates that the length of the longest common matched subsequence varies across languages, the reasons behind this variation are unclear.

The final metric, CIDEr takes into account if the matched words are frequent or rare (and thus more informative) using inverse document frequency (IDF). The more rare the matched words, the higher the score. The CIDEr score for Russian is significantly higher than for all other languages meaning that word overlap with reference questions contains more rare words. By contrast, the score CIDEr for Finnish is significantly lower than for all the other languages, meaning that most of the matched words are frequent ones (such as, question words, prepositions or common verbs). This makes both Russian and Finnish interesting candidates for human evaluation to see whether such significant difference in CIDEr scores results in significant difference in the quality of the questions according to human judges.

The only available fusional language, Arabic, exhibits similar performance to the agglutinative languages (except CIDEr in Finnish). The notable difference is the significantly higher number of induced templates. It is also interesting that no templates could be generated without the prior removal of punctuation as a pre-processing step. This calls for additional investigation of the quality of the output of Arabic's dependency parser and potentially further tweaks of Quinductor to suit fusional languages better.

Finally, the performance for Indonesian is on-par with Arabic, which is interesting, given that Indonesian is the only language in TyDi QA with a mixed question phrase position (meaning that some question phrases are obligatorily initial and some are not). However, the templates for Indonesian have been induced assuming that the first word of the reference question is a question word. Hence the obtained performance might be due to specific properties of the dataset and requires further investigation on other datasets.

5.1.2 Human evaluation

As mentioned previously, Finnish and Russian were interesting candidates for human evaluation, and were chosen along with English. For evaluation, we randomly sampled 50 sentences for each language, and generated QA-pairs for them using the induced templates. 50 generated QA-pairs were combined with 50 original QA-pairs from the corpus (later referred to as *gold* QA-pairs), corresponding to the same sampled sentences, and presented for evaluation in a random order to 5 human judges via the Prolific platform[4]. Each triple of a sentence and a QA-pair was judged using a questionnaire comprising 9 criteria (formulated as statements) to be evaluated on a 4-point Likert-type scale (from "Disagree" to "Agree"). Further details about the questionnaires, guidelines and evaluation process in general are provided in Appendix A.

The score of each judge per criterion is treated as a judgement on an ordinal scale, instead of treating all criteria together as an interval scale. The rationale behind such treatment is that a single aggregated quality score of questions and/or answers (over judgement criteria) is not very informative and will not help in pinpointing the exact problems observed in generated QA-pairs.

Following the work of Amidei, Piwek, and Willis (2019) we assess inter-annotator agreement (IAA) using Fleiss' κ (Fleiss 1971) and Goodman-Kruskall's γ (Goodman and Kruskal 1979). However, keeping in mind that we deal with ordinal data, the following two slight differences from (Amidei, Piwek, and Willis 2019) are introduced in our approach.

Firstly, Fleiss' κ (Fleiss 1971) measures the level of agreement compared to agreement by chance, originally defined by Fleiss through the marginal distribution of scores over categories (hence another name of this statistics – fixed-marginal κ). However, using Fleiss' κ is appropriate only if judges know a priori how many cases should be distributed into each category (see (Randolph 2005) for an extensive discussion on the matter). In our case, it would not make sense

4 https://www.prolific.co/

Table 5
Inter-annotator agreement per criterion. Q stands for "Question" and SA – for "Suggested answer"

Criterion	IAA Metric	en		fi		ru	
		gold	gen	gold	gen	gold	gen
Q is grammatically correct	Randolph's κ	0.34	0.12	0.75	0.22	0.76	0.58
	GK γ_N	0.53	0.63	0.83	0.83	0.71	0.87
Q makes sense	Randolph's κ	0.25	0.17	0.67	0.29	0.76	0.61
	GK γ_N	0.55	0.72	0.78	0.82	0.79	0.93
Q would be clearer if more information were provided	Randolph's κ	0.15	0.09	0.49	0.35	0.40	0.42
	GK γ_N	0.44	0.47	0.53	0.62	0.46	0.56
Q would be clearer if less information were provided	Randolph's κ	0.41	0.36	0.85	0.78	0.59	0.81
	GK γ_N	0.47	0.54	0.86	0.88	0.65	0.93
Q is relevant to the given sentence	Randolph's κ	0.21	0.19	0.38	0.18	0.32	0.54
	GK γ_N	0.64	0.55	0.67	0.70	0.79	0.81
SA correctly answers the question	Randolph's κ	0.25	0.23	0.54	0.32	0.30	0.57
	GK γ_N	0.75	0.73	0.83	0.82	0.61	0.86
SA would be clearer if phrased differently	Randolph's κ	0.03	0.05	0.59	0.35	0.29	0.31
	GK γ_N	0.27	0.42	0.62	0.47	0.56	0.49
SA would be clearer if more information were provided	Randolph's κ	0.16	0.06	0.55	0.33	0.31	0.35
	GK γ_N	0.38	0.42	0.59	0.62	0.58	0.61
SA would be clearer if less information were provided	Randolph's κ	0.53	0.55	0.87	0.96	0.83	0.86
	GK γ_N	0.67	0.66	0.92	0.90	0.74	0.82

to require judges to rate in this way, making the original Fleiss' κ inappropriate for our purposes. Instead the free-marginal alternative κ, introduced by Randolph (2005) and later referred to as Randolph's κ, should be used. In Randolph's κ the probability of agreement by chance is assumed to be uniform and thus suitable in our case.

Secondly, Goodman-Kruskall's γ (GK γ) was designed to measure rank correlation between ordinal judgements of two judges. Amidei, Piwek, and Willis (2019) averaged GK γ over pairs of judges, which is not interpretable from a statistical perspective, given that correlation coefficients are not additive (see Appendix C for a discussion on the matter). Instead of computing the mean, we propose a generalization of GK γ to multiple raters, dubbed γ_N (derived in Appendix C).

$$\Pi_N = \{(i,j) | i \in U, j \in U, i < j\} \tag{4}$$

$$C_N = \sum_{(i,j)\in\Pi_N} C_{ij}; \;\; D_N = \sum_{(i,j)\in\Pi_N} D_{ij}; \;\; \gamma_N = \frac{C_N - D_N}{C_N + D_N} \tag{5}$$

where U is the set of indices corresponding to human judges, C_{ij} (D_{ij}) is the number of concordant pairs (i.e., ranked in the same order) or discordant pairs (ranked in the reversed order), between judges i and j.

Inter-annotator agreement (IAA) for the conducted human evaluations are reported in Table 5 per criterion for gold and generated QA-pairs separately. On the scale for Fleiss' kappa

proposed by Landis and Koch (1977), there is a slight agreement between judges for most of the criteria for English, and a moderate agreement for Finnish and Russian. Following Amidei, Piwek, and Willis (2019) we use the scale for GK γ_N proposed by Rosenthal (1996). On this scale, there is a large correlation between the judgements on most of the criteria for English, and a very large correlation for Finnish and Russian. A notable observation is that IAA for English is substantially lower on all criteria, no matter the IAA metric, or whether the QA-pairs were generated or gold. Another observation of interest is that the generated QA-pairs get lower Randolph's κ, but higher GK γ_N compared to the gold ones in the vast majority of cases. This means that the exact scores for generated questions differ more than for gold ones, but the ranking order is more consistent.

To break down the results even further, we report the aggregated scores per each criterion using bi-variate histograms in Figure 11. Recall that we treat human judgements as ordinal data. Valid measures of central tendency for ordinal data are *median* (the value separating a higher half from the lower half of a sample), and *mode* (the most frequent value of a sample), whereas *mean* is not a valid measure for ordinal scales (see (Blaikie 2003, Chapter 3) for more information). Hence the scores have been aggregated by median (on x-axis) and mode (on y-axis) over all 5 judges. If there are multiple modes, the worst one was taken, meaning if the ideal value for a criterion is "Agree" (corresponding to the numeric value of 4), then the smallest mode was considered, otherwise the largest. The rationale behind this handling of multi-modal distributions is to penalize cases where the human judges could not come to a definite agreement. To aid reader in understanding this presentation format, we provide an annotated histogram in Figure 10.

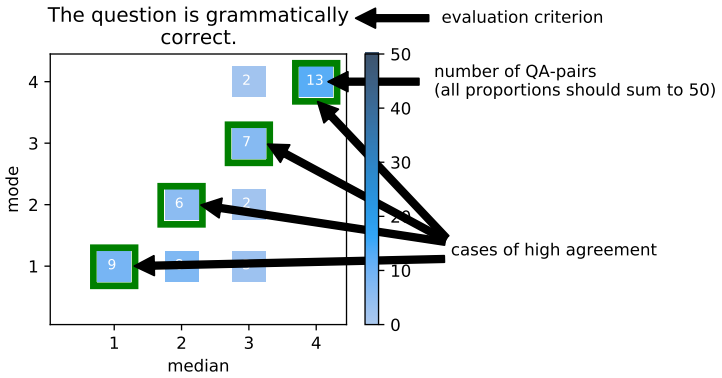
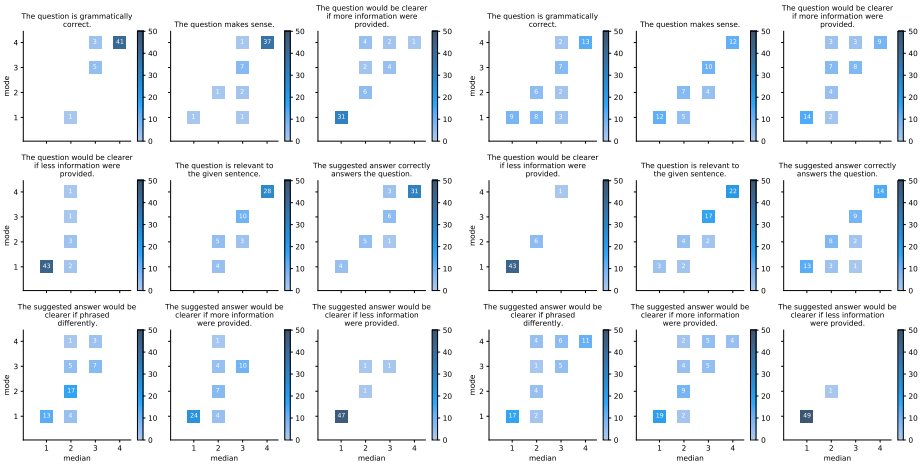


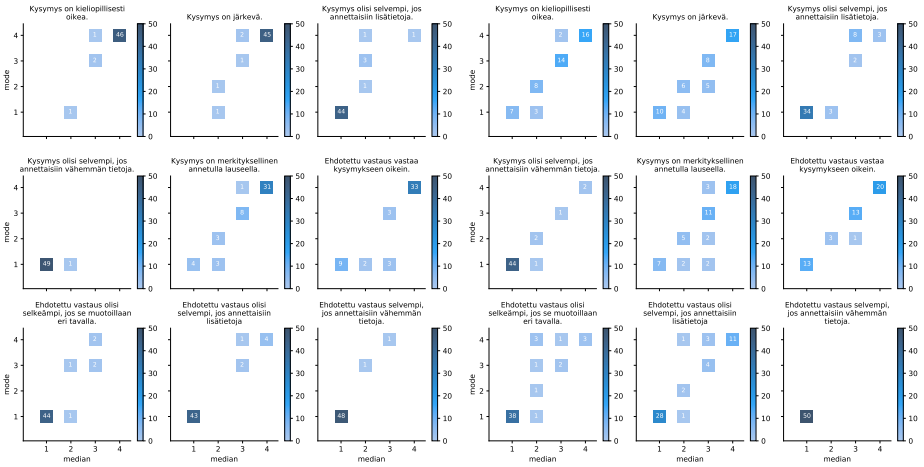
Figure 10
An annotated example of a bi-variate histogram

As established before, GK γ_N is higher than Randolph's κ, meaning we should trust the exact ratings less and the relative rankings more. Then we are interested in QA-pairs ranked better than all other pairs by most of the judges, preferably that both median and mode for these QA-pairs are either equal to 4 if the best rating for the given criterion is 4, or to 1 if the best rating is 1. The proportion of such cases per criterion per language is presented in Table 6.

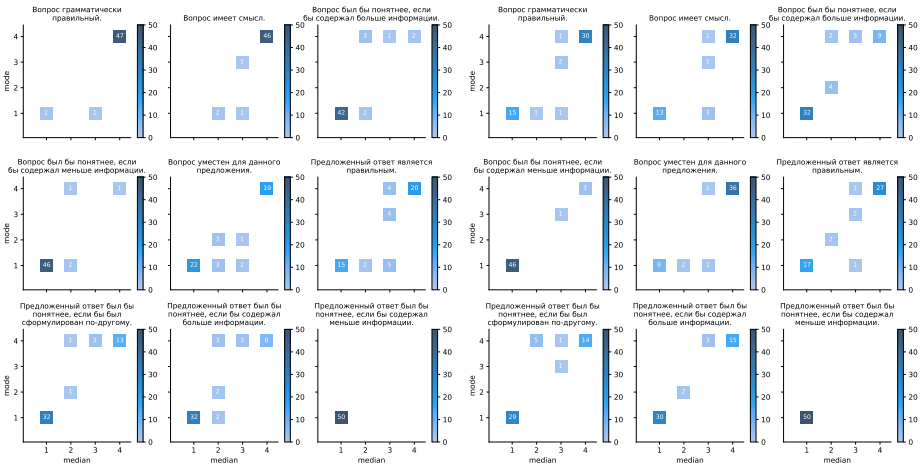
The majority of generated questions for English and Finnish are borderline (given 2 or 3) in terms of grammaticality, whereas the majority of questions in Russian were given a 4. It should be noted, though, that a considerable number of questions were evaluated as being grammatically incorrect (see Section 5.1.3 for error analysis). A similar pattern holds as to whether the question makes sense. The vast majority of the questions are not over-informative across all languages, and would not benefit from more information (except for English). Most of the cases for English



English (gold) English (generated)



Finnish (gold) Finnish (generated)



Russian (gold) Russian (generated)

Figure 11
Bi-variate histograms of human judgements (the order of criteria is the same for all languages)

Table 6
Proportion of generated QA pairs where both median and mode are the same

Criterion	Best if	en		fi		ru	
		1	4	1	4	1	4
Q is grammatically correct	4	18%	26%	14%	32%	30%	60%
Q makes sense	4	24%	24%	20%	34%	26%	64%
Q would be clearer if more information were provided	1	28%	18%	68%	6%	64%	18%
Q would be clearer if less information were provided	1	86%	0%	88%	4%	91%	6%
Q is relevant to the given sentence	4	6%	44%	14%	36%	18%	72%
SA correctly answers the question	4	26%	28%	26%	40%	34%	54%
SA would be clearer if phrased differently	1	34%	22%	76%	6%	58%	28%
SA would be clearer if more information were provided	1	38%	8%	56%	22%	60%	30%
SA would be clearer if less information were provided	1	98%	0%	100%	0%	100%	0%

and Finnish were borderline, whereas a substantial majority of questions were judged as relevant to the given sentence for Russian.

Focusing on the suggested answers, the majority of them have been reported to answer the question correctly for Finnish and Russian, whereas most of the cases were borderline for English. It should be noted that a substantial number of the suggested answers did not answer the question correctly. A breakdown of such cases is presented in Section 5.1.3. A considerable number of answers would benefit from rephrasing for English and Russian, whereas the majority of answers for Finnish would not (which is surprising given that Finnish is an agglutinative language with rich inflectional morphology). Almost none of the suggested answers are over-informative, and a majority of them would not benefit from more information either (except for English).

5.1.3 Error analysis

Exploring the questions that obtained a mode of 1 or 2 for the grammaticality criterion, we have identified three most frequent types of errors, which are summarized in Table 7. As can be seen, the types of errors are different across languages, suggesting that Quinductor's performance might be boosted for each individual language by applying language-specific post-processing (e.g., determiner correction for English). Some of the errors are connected to the errors in dependency parsing, such as split proper names in Russian, calling for a principled error analysis for these parsers beyond the provided development treebanks.

Another interesting issue pertains to questions that were judged grammatically correct (mode and median of 4 on the grammaticality criterion), but exhibited problems with respect to other criteria. Such cases are presented in Tables 8 and 9.

Table 7
Three most frequent types of grammatical mistakes for questions that received a mode of 1 or 2 for the criterion "The question is grammatically correct"

Lang.	Problem	Freq.	Example
en	Wrong question word	17.8%	Who is the poorest state in the United States of America?
	Underspecified	17.8%	Who finished career?
	Wrong article	14.3%	Which is a largest hub?
ru	A transitive verb lacks object	47.1%	Когда архиепископ признал на ландтаге в городе?
	A split of a proper name	11.8%	Когда Кеи» исключена компания «Мэри?
	Unresolved coreference	11.8%	Когда состоялся третий шаг?
fi	Wrong question word	61.1%	Mikä oli genovalainen tutkimusmatkailija?
	Question is nonsensical	16.7%	Milloin määrä olisi euroa?
	Missing parts of question	16.7%	Minä vuonna ensimmäinen elokuva Spring of Birth sai?

As can be seen in Table 9, most of the questions also make sense, but would benefit from including more information. The suggested answers exhibit much more variation in human judgements, both in terms of them being correct, requiring rephrasing or more information. Most of the suggested answers are not over-informative, as also illustrated by the provided samples in Table 9.

Table 8
Examples of QA-pairs judged grammatically correct (median and mode of 4), but exhibiting problems in other criteria.

Lang.	ID	Question	Suggested answer
en	EN1	Who served 18 months?	Susan McDougal
	EN2	Where was the Nobel Peace Prize awarded?	Frédéric in 1901
ru	RU1	Когда была основана компания?	тремя
	RU2	Когда скончался Эдуард?	5 января 1066
	RU3	Когда был закрыт монастырь?	1924
fi	FI1	Minä vuonna erä päättyi?	2.58
	FI2	Mitä sijamuodot ovat?	nominatiivi
	FI3	Mikä on tartuntatauti eli infektiotauti (morbus contagiosus)?	infektiosairaus

Table 9
Human judgements of the examples in Table 8. If only one number is specified, then mode and median are equal, otherwise the format is `median/mode`

Criterion	EN1	EN2	RU1	RU2	RU3	FI1	FI2	FI3
Q is grammatically correct	4	4	4	4	4	4	4	4
Q makes sense	3	4	4	4	4	2/1	4	4
Q would be clearer if more information were provided	4	4	1	4	2	2/1	1	1
Q would be clearer if less information were provided	1	1	1	1	1	1	1	1
Q is relevant to the given sentence	4	4	4	4	4	1	3/4	4
SA correctly answers the question	4	1	1	4	2	1	1	3
SA would be clearer if phrased differently	1	4	1	1	4	1	1	4
SA would be clearer if more information were provided	2	4	1	1	4	1	4	4
SA would be clearer if less information were provided	1	1	1	1	1	1	1	1

5.2 Comparison to other methods

To support the claim of Quinductor being a strong baseline we compare our method to previously reported results for both state-of-the-art and baseline methods. Most of the previous work is done for the SQuAD dataset (Rajpurkar et al. 2016), although the training/development/test split varies among articles, since the original SQuAD test set is hidden. We have found a number of articles relying on the SQuAD split[5] made by Du, Shao, and Cardie (2017) and others relying on the split made by Zhou et al. (2017). In this article use the former split and hence compare only to the articles that have explicitly reported to use of the same split to ensure a fair comparison between the methods. CIDEr is not provided in all other publications and is thus not reported. We induce templates based on the provided training set, and evaluate on the test set using automatic evaluation metrics only. The rationale for this is that some articles did not perform human evaluation at all (Kim et al. 2019; Song et al. 2018; Dong et al. 2019; Zhao et al. 2018), and others (Du, Shao, and Cardie 2017; Bahuleyan et al. 2017) used different criteria and evaluation guidelines making a fair comparison impossible.

As can be seen our method performs better than all reported baselines in terms of METEOR and ROUGE-L, and substantially better on BLEU-4 compared to the vanilla seq2seq model reported by Du, Shao, and Cardie (2017).

5 The SQuAD split is available at `https://github.com/xinyadu/nqg`
6 The result fot this model is taken from (Du, Shao, and Cardie 2017)

Table 10
Comparison to state-of-the-art QG methods and other reported baselines (shown in italics) on the test set of the SQuAD split made by Du, Shao, and Cardie (2017)

Article	BLEU-1	BLEU-4	METEOR	ROUGE-L
(Dong et al. 2019)	NA	22.12	25.06	51.07
(Kim et al. 2019)	NA	16.2	19.92	43.96
(Zhao et al. 2018)	45.07	16.38	20.25	44.48
(Song et al. 2018)	NA	13.98	18.77	42.72
(Du, Shao, and Cardie 2017)	43.09	12.28	16.62	39.75
(Bahuleyan et al. 2017)	30.87	5.08	NA	NA
Vanilla seq2seq[6]	31.34	4.26	9.88	29.75
H&S[6]	38.50	11.18	15.95	30.98
Ours	30.56	9.71	16.70	31.71

5.3 Cross-dataset evaluation

In this final part of the evaluation, we explore how the induced templates for English are generalizing across datasets. We use 4889 templates induced from the SQuAD training set (from the split by Du, Shao, and Cardie (2017)), and 254 templates induced from the TyDi QA training set for English, to generate QA-pairs on the SQuAD test set (from the split by Du, Shao, and Cardie (2017)) and the TyDi QA development set. The results of this cross-dataset evaluation using automatic metrics are presented in Table 11.

Table 11
Automatic cross-dataset evaluation for first-ranked generated questions in English.

Training - test	BLEU-1	BLEU-4	METEOR	ROUGE-L	CIDEr
SQuAD - SQuAD	30.56	9.71	16.70	31.71	7.69
SQuAD - TyDi QA	13.47	2.22	10.79	23.09	11.33
TyDi QA - TyDi QA	20.23	4.72	12.46	27.55	21.35
TyDi QA - SQuAD	34.31	11.12	14.83	30.61	8.84

As can be observed, QA-pairs generated based on TyDi QA templates generally perform better than those based on SQuAD (except METEOR and ROUGE-L for the TyDi QA-SQuAD setup). This means that the word overlap is larger and of higher quality (i.e., consists of less common words) for the TyDi QA dataset. One reason for such performance difference is that mean filtering during the ranking step was designed for less-resourced languages, when only a few questions are generated. However, mean filtering is substantially weaker if many QA-pairs are generated, especially if most of them have low ranks (which is likely for SQuAD).

6 Discussion

We have shown that the Quinductor method is a strong baseline method, outperforming baselines for English reported previously in the literature in terms of METEOR and ROUGE-L scores and

performing better (or not far behind) some of the previously proposed QG methods. In addition, our method is inexpensive to train both in terms of time and textual resources, and thus applicable to languages other than English.

Quinductor has been successfully applied to 5 *typologically diverse* less-resourced languages with limited training data. Most agglutinative languages (with a rich morphology and a free word order) performed similarly in terms of automatic evaluation metrics. Agglutinative languages with datasets relying on subwords in either questions or answers are proven to not work well with Quinductor (e.g., Korean and Telugu in TyDi QA dataset). Generated questions for Finnish performed better than English in terms of human judgements. Russian performed substantially better than all other languages both in terms of automatic evaluation metrics and human judgements, which might be a merit of a specific dataset and requires further investigation.

However, Quinductor has a number of limitations. Our method relies on the correctness of the dependency parser's output, or rather on the consistency of its errors. This assumption, although weaker than correctness, is still a limitation and does not always hold. We have noticed that some language-specific preprocessing techniques make the output of dependency parsers more consistent, but this requires further investigation.

Our method also incorporates a number of heuristics, such as mean filtering, selecting of contiguous template expressions in sentence transformation and ranking models, which seems to result in a comparable performance across languages. While only empirical evidence supports the applicability of these heuristics, we believe it is enough to make Quinductor a strong multilingual baseline, and set the lower bar for neural methods.

Another limiting property of Quinductor is that it lacks knowledge about semantics, since encoding such knowledge requires a large enough corpus that might not be available for all languages. While lack of semantic knowledge degrades the quality of questions, a surprisingly large number of them remain grammatically correct and make sense according to human evaluation.

It might be said that Quinductor still requires the use of a dependency parser to be trained on a sizeable dataset, and thereby moving the problem rather than solving it. However, firstly, the Universal Dependencies framework includes 200 treebanks for over 100 languages (and counting). Secondly, we have shown that Quinductor could induce templates even for Telugu, whose dependency parser is trained on a treebank with only 6K tokens. Thirdly, less-resourced languages have much smaller corpora of raw text, making pretraining of large-scale neural language models challenging (let alone fine-tuning them for QG). Finally, Quinductor method is a yet another use case for a dependency treebank, adding to the motivation of expanding UD to other languages.

Appendix A: Human evaluation details

Human evaluation has been conducted on the Prolific platform[7]. We used Prolific's pre-screening feature and required each human judge to have the language of interest as the first language and hold at least a high school diploma (A-levels).

The exact guidelines for human evaluation are presented in Figures 1, 2, 3. The instructions and 9 evaluation criteria are the same, but are translated into every language. Each criterion is evaluated on a 4-point Likert-type scale with the ends labeled as "Disagree" and "Agree". A neutral option is excluded, to force judges to make a decision. We opted out of a more typical "Strongly disagree" – "Strongly agree" scale to give judges some alternatives in the middle, such as, "Somewhat (dis)agree". Otherwise, the scale would be interpreted as "Strongly disagree" – "Disagree" – "Agree" – "Strongly agree", which effectively collapses it to a binary scale.

7 https://www.prolific.co/

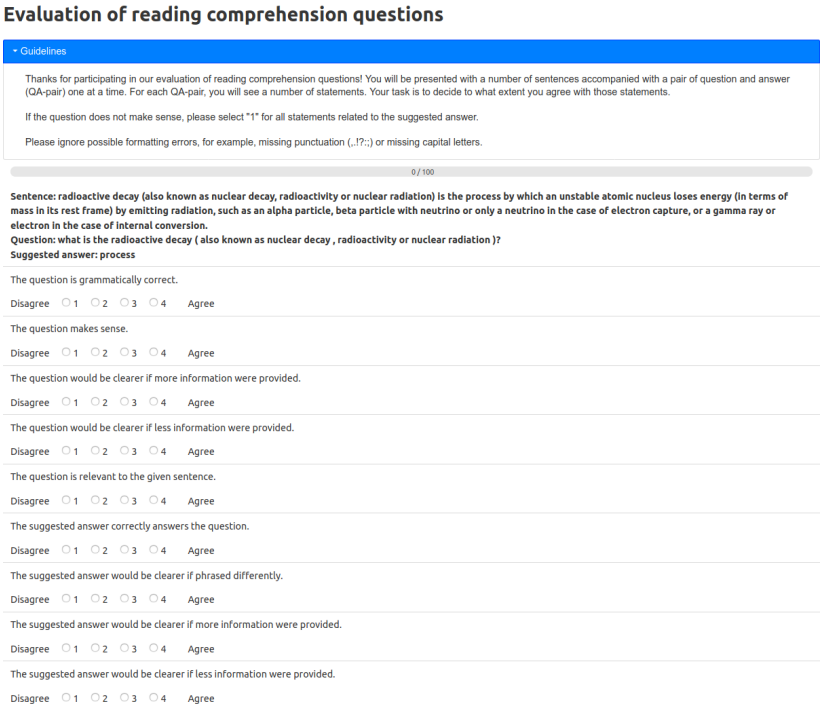


Figure 1
Evaluation guidelines and questionnaire for English

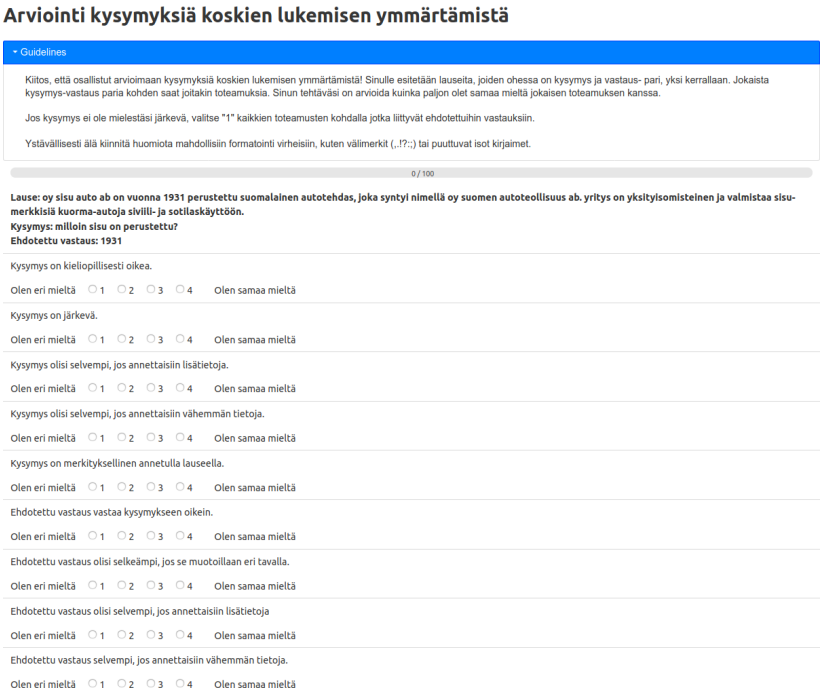


Figure 2
Evaluation guidelines and questionnaire for Finnish

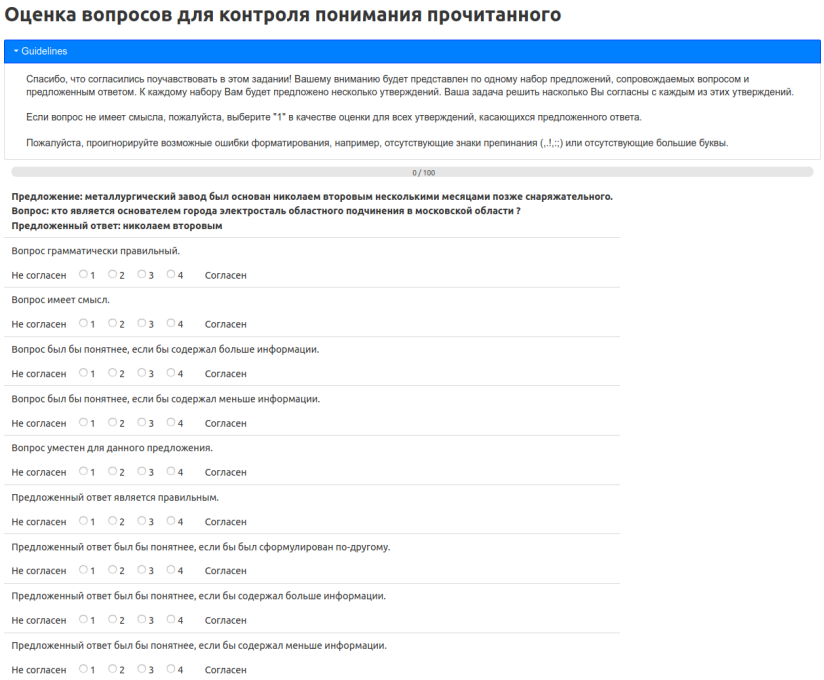


Figure 3
Evaluation guidelines and questionnaire for Russian

Appendix B: Data pre-processing steps

The applied pre-processing steps for every language are specified in Table 1. Note that Arabic is written from right to left, whereas Quinductor processes sentences from left to right. Hence the assumed question word position for Arabic is the first processed word, which is effectively the end of the sentence for Arabic.

Table 1
Language-specific pre-processing steps before template induction. S denotes start of the sentence, E – end of the sentence.

Preprocessing step	fi	ja	te	ar	id	ko	ru	en
Lowercase	✔	NA	NA	NA	✔	NA	✔	✔
Remove punctuation	✘	✔	✔	✔	✘	✘	✘	✘
Remove diacritics	✘	✘	✘	✘	✘	✘	✔	✘
Assumed question word position	S	E	E	S	S	E	S	S

Appendix C: Derivation of multi-rater Goodman-Kruskall's γ_N

Let us start by presenting the original derivation of GK γ, proposed by Goodman and Kruskal (1979), applied to the case of human evaluation. Assume we have two judges independently assigning scores from 1 to α to the same set of M items. Let s_{aj} denote a random variable associated with scores of judge a to the item j. Let $c_{1,2}$ denote the event that a randomly selected pair of items will be ordered in the same way by judges 1 and 2 (such pair is called

concordant). The probability of such event can then be calculated using Equation (C.1), given that the judgements are independent.

$$P(c_{1,2}) = \sum_{(i,j)\in\Pi_M} P(s_{1i} < s_{1j})P(s_{2i} < s_{2j}) + P(s_{1i} > s_{1j})P(s_{2i} > s_{2j}) \tag{C.1}$$

Let $d_{1,2}$ denote the event that a randomly selected pair of items will be ordered differently by judges 1 and 2 (such pair is called **discordant**). The probability of such event can then be calculated using Equation (C.2), given that the judgements are independent.

$$P(d_{1,2}) = \sum_{(i,j)\in\Pi_M} P(s_{1i} < s_{1j})P(s_{2i} > s_{2j}) + P(s_{1i} > s_{1j})P(s_{2i} < s_{2j}) \tag{C.2}$$

Let $t_{1,2}$ denote the event that a randomly selected pair of items will be get the same scores with each other (be **tied**) by both judges. Then the probability of ties is calculated using Equation (C.3) given that the judgements are independent.

$$P(t_{1,2}) = \sum_{(i,j)\in\Pi_M} P(s_{1i} = s_{1j}) + P(s_{2i} = s_{2j}) \tag{C.3}$$

In all equations above Π_M is a 2-combination of the set of item indices between 1 and M. The conditional probability of concordant orders given no ties ($\widetilde{t_{12}}$) then equals to:

$$P(c_{1,2}|\widetilde{t_{1,2}}) = \frac{P(\widetilde{t_{1,2}}|c_{1,2})P(c_{1,2})}{P(\widetilde{t_{1,2}})} = \frac{1 \cdot P(c_{1,2})}{1 - P(t_{1,2})} = \frac{P(c_{1,2})}{1 - P(t_{1,2})} \tag{C.4}$$

Similarly, the conditional probability of discordant orders given no ties equals to $P(d_{1,2}|\widetilde{t_{1,2}}) = \frac{P(d_{1,2})}{1-P(t_{1,2})}$. GK γ was then proposed by Goodman and Kruskal (1979) to be computed as

$$\gamma_{1,2} = \frac{P(c_{1,2}) - P(d_{1,2})}{1 - P(t_{1,2})} \tag{C.5}$$

Observe that $P(c_{1,2}|\widetilde{t_{1,2}}) + P(d_{1,2}|\widetilde{t_{1,2}}) = 1$, since if there are no ties, there can be either concordant or discordant orders, then the following derivation holds:

$$P(c_{1,2}|\widetilde{t_{1,2}}) + P(d_{1,2}|\widetilde{t_{1,2}}) = 1 \tag{C.6}$$

$$\frac{P(c_{1,2})}{1 - P(t_{1,2})} + \frac{P(d_{1,2})}{1 - P(t_{1,2})} = 1 \tag{C.7}$$

$$\frac{P(c_{1,2}) + P(d_{1,2})}{1 - P(t_{1,2})} = 1 \tag{C.8}$$

$$P(c_{1,2}) + P(d_{1,2}) = 1 - P(t_{1,2}) \tag{C.9}$$

Using this observation, GK $\gamma_{1,2}$ can be rewritten to a more familiar form:

$$\gamma_{1,2} = \frac{P(c_{1,2}) - P(d_{1,2})}{P(c_{1,2}) + P(d_{1,2})} \tag{C.10}$$

Now assume we have a third judge as well and we calculate $\gamma_{1,2}$, $\gamma_{1,3}$, and $\gamma 2, 3$. Then to evaluate the agreement between three judges we simply take an average of them. Let us see what it amounts to.

$$\gamma_{1-3} = \frac{\gamma_{1,2} + \gamma_{1,3} + \gamma_{2,3}}{3} \tag{C.11}$$

$$= \frac{\frac{P(c_{1,2})-P(d_{1,2})}{P(c_{1,2})+P(d_{1,2})} + \frac{P(c_{1,3})-P(d_{1,3})}{P(c_{1,3})+P(d_{1,3})} + \frac{P(c_{2,3})-P(d_{2,3})}{P(c_{2,3})+P(d_{2,3})}}{3} \tag{C.12}$$

The quantity in Equation C.12 cannot be simplified further resulting neither in a valid probability nor in a generalized version of GK γ.

Instead the derivation process can easily be extended to N judges ($N > 2$), as follows, resulting in a generalized version of GK γ, dubbed γ_N. Let c_N, d_N and t_N be the events that a randomly selected pair of items is concordant, discordant or tied (respectively) by any pair selected from N judges, then the following holds.

$$P(c_N) = \sum_{(a,b)\in\Pi_N} \sum_{(i,j)\in\Pi_M} P(s_{ai} < s_{aj})P(s_{bi} < s_{bj}) + P(s_{ai} > s_{aj})P(s_{bi} > s_{bj}) \tag{C.13}$$

$$P(d_N) = \sum_{(a,b)\in\Pi_N} \sum_{(i,j)\in\Pi_M} P(s_{ai} < s_{aj})P(s_{bi} > s_{bj}) + P(s_{ai} > s_{aj})P(s_{bi} < s_{bj}) \tag{C.14}$$

$$P(t_N) = \sum_{(a,b)\in\Pi_N} \sum_{(i,j)\in\Pi_M} P(s_{ai} = s_{aj}) + P(s_{bi} = s_{bj}) \tag{C.15}$$

$$P(c_N|\widetilde{t_N}) = \frac{P(c_N)}{1 - P(t_N)} \tag{C.16}$$

$$P(d_N|\widetilde{t_N}) = \frac{P(d_N)}{1 - P(t_N)} \tag{C.17}$$

$$\gamma_N = \frac{P(c_N) - P(d_N)}{P(c_N) + P(d_N)} = \frac{\sum\limits_{(a,b)\in\Pi_N} P(c_{a,b}) - \sum\limits_{(a,b)\in\Pi_N} P(d_{a,b})}{\sum\limits_{(a,b)\in\Pi_N} P(c_{a,b}) + \sum\limits_{(a,b)\in\Pi_N} P(d_{a,b})} \tag{C.18}$$

Π_M is a 2-combination of the set of item indices between 1 and M, Π_N is a 2-combination of the set of judge indices between 1 and N,

Appendix D: Samples of generated questions

Here we present samples of generated questions for English, Finnish and Russian. All sentences, questions and suggested answers are lowercased, since we have empirically found Stanza's tokenizers and dependency parsers to perform more consistently when text is lowercased. Both QA-pairs and sentences were also lowercased for human judges during evaluation.

Sentence: diphenhydramine was first made by george rieveschl and came into commercial use in 1946
Question: who made diphenhydramine?
Suggested answer: george rieveschl

Sentence: parallax (from ancient greek παράλλαξις (parallaxis), meaning 'alternation') is a displacement or difference in the apparent position of an object viewed along two different lines of sight, and is measured by the angle or semi-angle of inclination between those two lines.
Question: what is a displacement in the apparent position of an object viewed along two different lines of sight?
Suggested answer: parallax

Sentence: the lowest temperatures are registered in july and august (18°c - 64°f) and the highest in february (maximum temperature 28°c - 82.4°f [1]), the summer season in the southern hemisphere.
Question: where are the lowest temperatures registered?
Suggested answer: july

Sentence: the nobel peace prize was first awarded in 1901 to frédéric passy and henry dunant — who shared a prize of 150,782 swedish kronor (equal to 7,731,004 kronor in 2008) — and, most recently, to denis mukwege and nadia murad in 2018
Question: where was the nobel peace prize awarded?
Suggested answer: frédéric in 1901

Sentence: in 1986, the first statute aimed at defense contractor employee whistleblower protection was enacted.
Question: when was the first statute aimed at defense contractor employee whistleblower protection enacted?
Suggested answer: 1986

Sentence: кхл была образована в 2008 году и объединяла в себе первоначально 24 команды.
Question: когда была образована кхл?
Suggested answer: 2008 году

Sentence: салли маргарет филд родилась в пасадине, калифорния, 6 ноября 1946 года в семье киноактрисы маргарет филд и армейского офицера ричарда драйдена[1].
Question: когда родилась салли маргарет филд?
Suggested answer: 6

Sentence: 14 июня 1952 в сша была заложена первая в мире апл «наутилус» (english: uss nautilus), и она была спущена на воду 21 января 1954 года[1][2][3].
Question: когда была заложена первая в мире апл «наутилус» (english : uss nautilus)?
Suggested answer: 14 июня 1952

Sentence: санатана родился в 1488 году в бенгальской деревне в провинции джессор.
Question: когда родился санатана?
Suggested answer: 1488

Sentence: металлургический завод был основан николаем второвым несколькими месяцами позже снаряжательного
Question: когда был основан металлургическии завод?
Suggested answer: несколькими

Sentence: jäämerentie oli valmistuessaan 531 kilometriä pitkä ja viisi metriä leveä.
Question: kuinka pitkä jäämerentie oli?
Suggested answer: 531 kilometriä

Sentence: fennomania, suomenmielisyys, suomenkiihko[1] oli suomalaisten kansallisen heräämisen liike, joka syntyi 1800-luvun alkupuolella ja vaikutti erityisesti saman vuosisadan jälkipuolella.
Question: mitä fennomania oli?
Suggested answer: liike

Sentence: kaupungin väkiluku on noin 118000, ja sen pinta-ala on km^2, josta km^2 on vesistöjä.[1] kuopion keskustaajama sijaitsee kallaveteen etelästä työntyvällä kuopionniemellä, joka jakaa kallaveden kahteen toisistaan lähes erilliseen osaan.
Question: paljonko on kaupungin väkiluku?
Suggested answer: 118000

Sentence: sen konsentraatio 25°c:n lämpötilassa on noin 1,004·10-7mol/l eli sen ph-arvo on 7,0
Question: paljonko on sen konsentraatio 25°c:n lämpötilassa?
Suggested answer: 1,004·10-7mol/l

Sentence: pegaso oli ajoneuvojen tuotemerkki, joka kuului vuonna 1945 generalissimus francisco francon valtiollistamaa espanjan ajoneuvoteollisuutta yhdistämällä vuonna 1946 syntyneeseen, madridissa kotipaikkaansa pitäneeseen enasa:an [empresa nacional de autocamiones s.a.).
Question: mitä pegaso oli?
Suggested answer: tuotemerkki

Acknowledgements

This work was supported by Vinnova (Sweden's Innovation Agency) within project 2019-02997. We would also like to thank Lisse-Lotte Hermansson for helping with translating instructions for human evaluation in Finnish, Kristiina Savola for help in assessing results of human evaluation for Finnish and Bram Willemsen for helpful comments and discussions on the matter of evaluation.

References

Afzal, Naveed and Ruslan Mitkov. 2014. Automatic generation of multiple choice questions using dependency-based semantic relations. *Soft Computing*, 18(7):1269–1281.

Agarwal, Abhaya and Alon Lavie. 2008. Meteor, m-bleu and m-ter: Evaluation metrics for high-correlation with human rankings of machine translation output. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 115–118.

Agarwal, Manish, Rakshit Shah, and Prashanth Mannem. 2011. Automatic question generation using discourse cues. In *Proceedings of the 6th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 1–9, Association for Computational Linguistics.

Amidei, Jacopo, Paul Piwek, and Alistair Willis. 2018. Evaluation methodologies in automatic question generation 2013-2018.

Amidei, Jacopo, Paul Piwek, and Alistair Willis. 2019. Agreement is overrated: A plea for correlation to assess human evaluation reliability.

Bahuleyan, Hareesh, Lili Mou, Olga Vechtomova, and Pascal Poupart. 2017. Variational attention for sequence-to-sequence models. *arXiv preprint arXiv:1712.08207*.

Bernhard, Delphine, Louis De Viron, Véronique Moriceau, and Xavier Tannier. 2012. Question generation for french: collating parsers and paraphrasing questions. *Dialogue & Discourse*, 3(2):43–74.

Blaikie, Norman. 2003. *Analyzing quantitative data: From description to explanation*. Sage.

Brown, Tom B, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Callison-Burch, Chris, Miles Osborne, and Philipp Koehn. 2006. Re-evaluating the role of bleu in machine translation research. In *11th Conference of the European Chapter of the Association for Computational Linguistics*.

Chan, Ying-Hong and Yao-Chung Fan. 2019. A recurrent bert-based model for question generation. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 154–162.
Clark, Jonathan H, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020. Tydi qa: A benchmark for information-seeking question answering in ty pologically di verse languages. *Transactions of the Association for Computational Linguistics*, 8:454–470.
Denkowski, Michael and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the ninth workshop on statistical machine translation*, pages 376–380.
Dong, Li, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. *arXiv preprint arXiv:1905.03197*.
Dryer, Matthew S. 2005. 93 position of interrogative phrases in content questions.
Du, Xinya, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. *arXiv preprint arXiv:1705.00106*.
Fleiss, Joseph L. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.
Gates, Donna Marie. 2011. How to generate cloze questions from definitions: A syntactic approach. In *2011 AAAI Fall Symposium Series*.
Gatt, Albert and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170.
Goodman, Leo A and William H Kruskal. 1979. Measures of association for cross classifications. *Measures of association for cross classifications*, pages 2–34.
Heilman, Michael and Noah A Smith. 2009. Question generation via overgenerating transformations and ranking. Technical report, Carnegie-Mellon Univ Pittsburgh pa language technologies insT.
Kalpakchi, Dmytro and Johan Boye. 2020. Udon2: a library for manipulating universal dependencies trees. In *28th International Conference on Computational Linguistics, COLING 2020, 8-13 December 2020*, pages 120–125.
Khullar, Payal, Konigari Rachna, Mukul Hase, and Manish Shrivastava. 2018. Automatic question generation using relative pronouns and adverbs. In *Proceedings of ACL 2018, Student Research Workshop*, pages 153–158.
Kim, Yanghoon, Hwanhee Lee, Joongbo Shin, and Kyomin Jung. 2019. Improving neural question generation using answer separation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6602–6609.
Kumar, Vishwajeet, Nitish Joshi, Arijit Mukherjee, Ganesh Ramakrishnan, and Preethi Jyothi. 2019. Cross-lingual training for automatic question generation. *arXiv preprint arXiv:1906.02525*.
Landis, J Richard and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174.
van der Lee, Chris, Albert Gatt, Emiel van Miltenburg, and Emiel Krahmer. 2020. Human evaluation of automatically generated text: Current trends and best practice guidelines. *Computer Speech & Language*, page 101151.
Liao, Yi, Xin Jiang, and Qun Liu. 2020. Probabilistically masked language model capable of autoregressive generation in arbitrary word order. *arXiv preprint arXiv:2004.11579*.
Lin, Chin-Yew. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.
Liu, Bang, Mingjun Zhao, Di Niu, Kunfeng Lai, Yancheng He, Haojie Wei, and Yu Xu. 2019. Learning to generate questions by learningwhat not to generate. In *The World Wide Web Conference*, pages 1106–1118.
Mazidi, Karen and Rodney D Nielsen. 2015. Leveraging multiple views of text for automatic question generation. In *International Conference on Artificial Intelligence in Education*, pages 257–266, Springer.
Mostow, Jack and Hyeju Jang. 2012. Generating diagnostic multiple choice comprehension cloze questions. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 136–146.
Nema, Preksha and Mitesh M Khapra. 2018. Towards a better metric for evaluating question generation systems. *arXiv preprint arXiv:1808.10192*.
Nivre, Joakim, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal dependencies v2: An evergrowing multilingual treebank collection. *arXiv preprint arXiv:2004.10643*.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Qi, Peng, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. *arXiv preprint arXiv:2003.07082*.

Rajpurkar, Pranav, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Randolph, Justus J. 2005. Free-marginal multirater kappa (multirater k [free]): An alternative to fleiss' fixed-marginal multirater kappa. *Online submission*.

Rosenthal, James A. 1996. Qualitative descriptors of strength of association and effect size. *Journal of social service Research*, 21(4):37–59.

Rus, Vasile, Zhiqiang Cai, and Art Graesser. 2008. Question generation: Example of a multi-year evaluation campaign. *Proc WS on the QGSTEC*.

Rus, Vasile, Brendan Wyse, Paul Piwek, Mihai Lintean, Svetlana Stoyanchev, and Cristian Moldovan. 2010. Overview of the first question generation shared task evaluation challenge. In *Proceedings of the Third Workshop on Question Generation*, pages 45–57.

Sharma, Shikhar, Layla El Asri, Hannes Schulz, and Jeremie Zumer. 2017. Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation. *CoRR*, abs/1706.09799.

Song, Linfeng, Zhiguo Wang, Wael Hamza, Yue Zhang, and Daniel Gildea. 2018. Leveraging context information for natural question generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 569–574.

Vedantam, Ramakrishna, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Zhao, Yao, Xiaochuan Ni, Yuanyuan Ding, and Qifa Ke. 2018. Paragraph-level neural question generation with maxout pointer and gated self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3901–3910.

Zhou, Qingyu, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017. Neural question generation from text: A preliminary study. In *National CCF Conference on Natural Language Processing and Chinese Computing*, pages 662–671, Springer.