Natural Language Engineering (2024), **30**, pp. 217–255
doi:10.1017/S1351324923000037

CAMBRIDGE UNIVERSITY PRESS

ARTICLE

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

Dmytro Kalpakchi* and Johan Boye

Division of Speech, Music and Hearing, KTH Royal Institute of Technology, Stockholm, Sweden
*Corresponding author. E-mail: dmytroka@kth.se

(Received 9 July 2021; revised 3 January 2023; accepted 17 January 2023; first published online 27 February 2023)

Abstract

We propose a multilingual data-driven method for generating reading comprehension questions using dependency trees. Our method provides a strong, 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 in terms of automatic evaluation metrics and shows a good performance in terms of human evaluation.

Keywords: Natural language generation; Evaluation; Multilinguality; Question generation; Reading comprehension

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. If made readily available, such a QG system could be used by second-language learners to test their understanding of a text they have read, or by teachers to quickly generate reading comprehension questions to be used in class. Numerous methods have been proposed for solving the QG task, most of which have been aimed at the English language. 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. State-of-the-art QG methods are data-driven and rely on the availability of large-scale data sets, such as SQuAD (Rajpurkar *et al.* 2016) – a question-answering data set repurposed for QG. At the core of these methods for English are modern language models based on the transformer architecture, such as BERT (Devlin *et al.* 2019) or GPT-3 (Brown *et al.* 2020). It is well known that the training of such language models requires large resources, both in terms of computing power and available text material in digital form. 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 scarcity of textual resources for pre-training of large-scale language models, for example, BERT or GPT-3, for languages other than English. Training data for English BERT consisted of the BooksCorpus (800M words) and English Wikipedia (2500M words at that time). Currently, English Wikipedia is the largest Wikipedia in the world containing around

© The Author(s), 2023. Published by Cambridge University Press. This is an Open Access article, distributed under the terms of the Creative Commons Attribution licence (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted re-use, distribution, and reproduction in any medium, provided the original work is properly cited.

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

6.1 million articles and 3.6 billion words.[a] In contrast, Wikipedias for languages with a richer inflectional morphology are significantly smaller, for example, Russian (1.7 million articles and 0.8 billion words[b]), Finnish (0.5 million articles and 0.14 billion words[c]) or Turkish (0.4 million articles and 0.11 billion words[d]). There are no guarantees that pre-trained BERT models on much smaller textual resources will result in as useful representations as those for English, especially for languages with richer morphology. Furthermore, GPT-3, currently available only for English, was trained on a substantially larger data set than BERT with English Wikipedia comprising only 0.6% of all training data tokens (although it was oversampled as a high-quality data set).

The second challenge is the lack of large-scale training data sets for QG 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 data set, which contains more than 100,000 questions. Obtaining a good-quality data set of a similar size is very expensive, especially for languages with fewer native speakers around the world.

The third challenge is knowing how well available methods developed for English would generalise to other languages, especially synthetic ones with a richer inflectional morphology and a 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 fourth 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 rule-based baselines. However, most other languages lack QG baselines, leaving researchers to wonder if the obtained performance is worth the computational resources spent on training the model.

In this article, we are addressing all of the mentioned challenges by proposing a novel, mostly deterministic method, called Quinductor[e] (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 (substantially more than the number of languages with the pre-trained BERT or GPT-3). The method does require a language-specific QG data set, but its size can be orders of magnitude smaller than SQuAD. Quinductor thus provides a middle ground between data-hungry computationally expensive methods based on neural networks and methods based on manually crafted templates. 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; Agarwal, Shah, and Mannem 2011; Mostow and Jang 2012). 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

[a] https://en.wikipedia.org/wiki/Special:Statistics.
[b] https://ru.wikipedia.org/wiki/Служебная:Статистика.
[c] https://fi.wikipedia.org/wiki/Toiminnot:Tilastot.
[d] https://tr.wikipedia.org/wiki/Ozel:Istatistikler.
[e] The code is available at https://github.com/dkalpakchi/quinductor.

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

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).

Rodrigues (2020) induced patterns relying on both constituency and dependency parsers to generate questions in English. On top of that, their system uses a plethora of other resources, such as a named entity recognizer, WordNet (Miller 1995), FrameNet (Baker, Fillmore, and Lowe 1998) and VerbNet (Kipper *et al.* 2000). These resources are of high quality, but most of them are very expensive and labour-intensive to construct and are thus unavailable for languages other than English.

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 data sets 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; Dong *et al.* 2019; Liao *et al.* 2020) and auto-regressive models based on encoder–decoder architectures (Du, Shao, and Cardie 2017; Song *et al.* 2018; Zhao *et al.* 2018; Bahuleyan *et al.* 2018; Kim *et al.* 2019; Liu *et al.* 2019). Note that 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 data set for Hindi and Chinese.

3. Methodology

Let D be a data set 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 data set 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

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

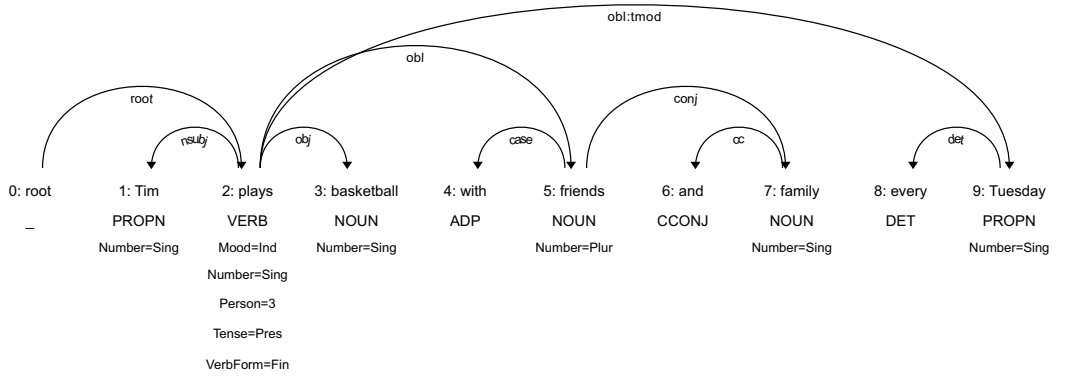

Figure 1: The dependency tree for the sentence 'Tim plays basketball with friends and family every Tuesday'.

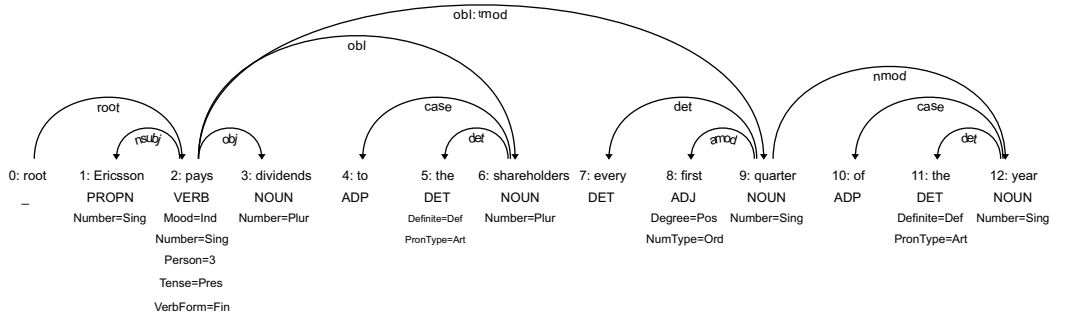

Figure 2: The dependency tree for the sentence 'Ericsson pays dividends to the shareholders every first quarter of the year'.

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> ?`
(Tim) (play) (basketball) (with friends and family)

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

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.

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

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

Sept 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.

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

Sept 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 generalisation is addressed by a novel shift-reduce algorithm, described in Section 3.3. Automatic induction of guards is described in Section 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*, that is, 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.

- `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] [r.lemma] [r.obj] <r.obl>?`
(Tim) (play) (basketball) (with friends and family) **OR** (every Tuesday)

(5) `<r.obl>`
(with friends and family) **OR** (every Tuesday)

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

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

that those expressions correspond to different nodes. Note that differentiating between nodes with the same dependency relations is the only purpose of IDs, that is, we do NOT require the new sentences using QA-templates to have exactly the same IDs, as it would obviously hinder generalisation.

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> ? =>`
(Tim) (play) (basketball) (with friends and family)
`<r.obl:tmod#9>`
(every Tuesday)

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, for example, '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:

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

- `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 generalise 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 generalisation 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 generalise poorly),

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

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 generalise 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 generalise 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), that is, we exclude all templates with a maximal IDF among their constant words exceeding $\log\left(\frac{N}{\frac{N}{4}}\right) = \log(4)$, where N is the number of documents in the corpus.

For demonstrating the algorithm, we consider the sentence 'The longest river in Brazil is the Amazon river' and the QA-pair 'What is the longest river in Brazil?' – 'Amazon river', which we will refer to as *an example QA-pair*.

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 (shown in Figure 3). The sentence transformation applied to question from the example QA-pair would then take the following form:

(8) `What [r.cop#6] [r.det#7] [r.nsubj#3.amod#2] [r.nsubj#3]`
(is) (the) (longest) (river)
`[r.nsubj#3.nmod#5.case#4] [r.nsubj#3.nmod#5]?`
(in) (Brazil)

Whereas sentence transformation applied to the answer of the example QA-pair would be:

(9) `[r.compound#8] [r]`
(Amazon) (river)

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 (for example, see dependency trees in Figure 3). T_{q_i} is then traversed in linear order $L_{T_{q_i}}$. 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 FORM or LEMMA. If no matching nodes are found, n_q is replaced by its FORM. 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 expressions will be referred to as **LLTE**. The reason behind introducing LLTE is that certain words might appear more than once in s_i, for instance, the word 'river' (appearing as both `r` and `r.nsubj#3` in Figure 3). We therefore need the heuristic method for finding the correct occurrence of the word. This heuristic, called LLTE resolution, is presented below. We provide an example for the LLTE of the question from the example QA-pair in the grey box marked 'LLTE' in Figure 3. 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 FORM of the token itself).

A template will generalise if many syntactic structures can be merged into subtree-level expressions, which is the goal of the next shift-reduce step of Quinductor. Hence, the result of sentence

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

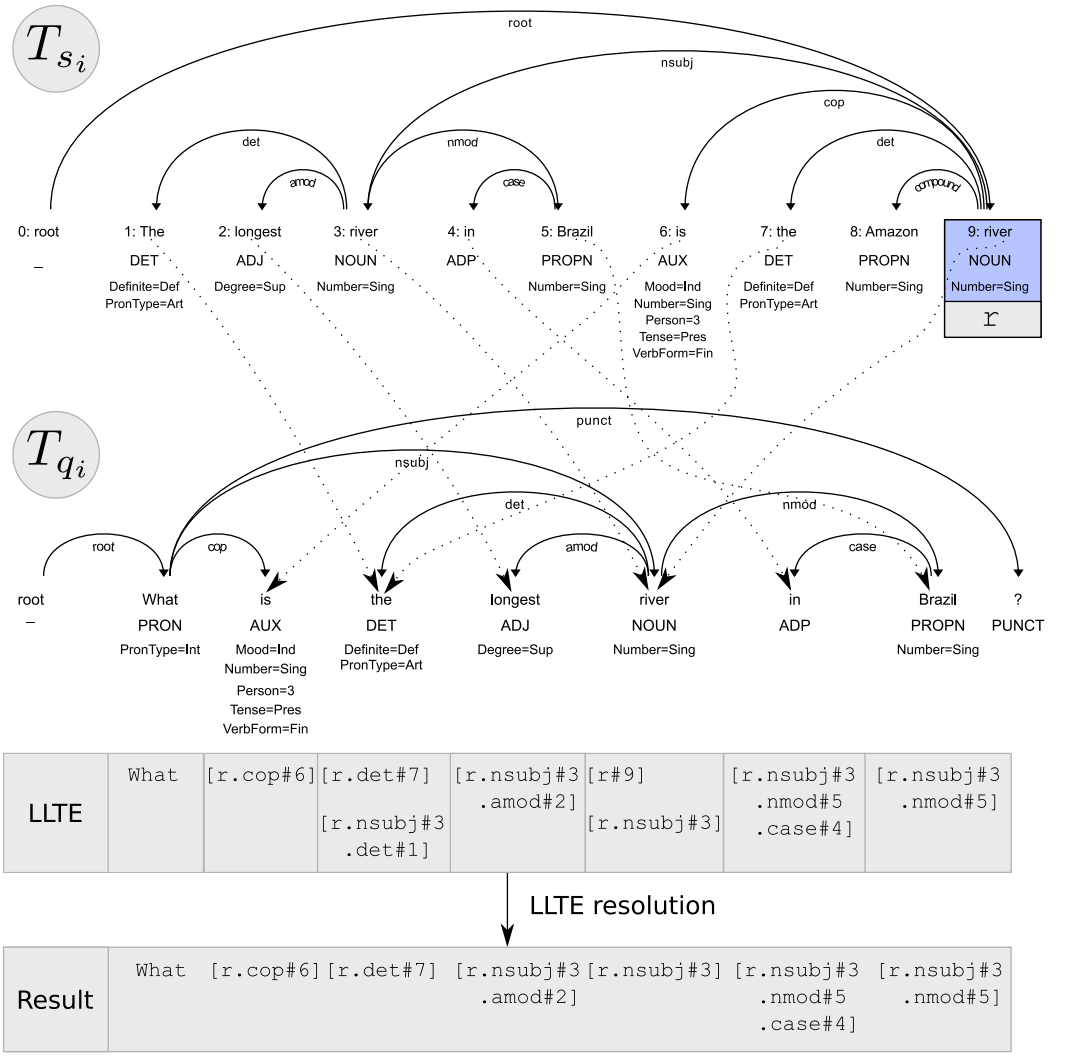

Figure 3: Illustration of the sentence transformation step on the question from the example QA-pair.

transformation should contain as many long *contiguous* phrases as possible. Achieving this goal imposes two requirements on LLTE: each list of LLTE should contain only one expression, and the resulting LLTE should contain a maximal possible amount of contiguous phrases. To address both requirements, we designed a heuristic for *LLTE resolution* that filters out template expressions from these lists until both of them contain one template expression each. To perform the filtering, we use the sum of absolute ID differences for the whole of question given each candidate pair of template expressions in positions under consideration. To construct candidate pairs, we process the lists from the left to right and combine every expression with every other in the order of their appearance. To give an abstract example, if we have two lists [1, 2] and [3, 4], then we would construct the following candidate pairs [(1, 3), (1, 4), (2, 3), (2, 4)].

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

Table 1. Sums of absolute ID differences for alternative representations of the words 'the' and 'river' for resolving LLTE in Figure 3

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

To exemplify the heuristic, consider LLTE shown in Figure 3. As we can see, most of the words already have one corresponding template expression, except for the words 'the' and 'river' with 2 template expressions each. For these words, we need to apply our heuristic. The sums of absolute ID differences for different combinations of the template expressions for the words 'the' and 'river' are presented in Table 1. To exemplify these calculations, for the candidate template expressions `[r.det#7]` and `[r#9]` (for the words 'the' and 'river,' respectively). First, we skip the constant 'What', since it is not a template expression. Then we calculate the difference between the IDs of the template expressions in the next two adjacent positions, that is, `r.cop#6` and `r.det#7`, which amounts to $|7-6|=1$. Then we take template expressions in the next two adjacent positions, that is, `r.det#7` and `r.nsubj#3.amod#2`. Recall that we defined the ID of a template expression to be the ID of the last element of the relchain, hence the absolute difference in this case amounts to $|2-7|=5$. And we continue this procedure until we reached the end of the question, which in total amounts to $|7-6|+|2-7|+|9-2|+|4-9|+|5-4|=19$, which corresponds to the value in the first row of Table 1.

We assume that the question with most contiguous phrases should have LLTE with the minimal sum of absolute ID differences. In our example, two candidate pairs have exactly the same sum of 9, in which case we pick the pair that comes first in the list (recall that the list is sorted in the ascending order by the distance from the root node of s_i in edges), that is, `[w.det#7]` for 'the' and `[w.nsubj#3]` for 'river', resulting in the following sentence transformation

(10) `What [r.cop#6] [r.det#7] [r.nsubj#3.amod#2] [r.nsubj#3]`
(is) (the) (longest) (river)
`[r.nsubj#3.nmod#5.case#4] [r.nsubj#3.nmod#5]?`
(in) (Brazil)

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 expressions in each list in LLTE (recall, that those are ordered by the distance from the root, in edges). We want to stress that this is a heuristic and will not result in the optimal ordering all the time. Furthermore, we believe that there is 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 generalisable, 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:

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

- **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.

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 find their longest common prefix, 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:

a. the common relchain is not empty;
b. 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 generalisation chances.

Step 3 If the aforementioned conditions are met, `stackTop` and `stackTop2` can be replaced by the subtree-level 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 exemplify, shift-reduce is applied to the template (10), obtained in Section 3.3.1, as follows. All the configurations, referred to in the description below, can be found in Figure 4.

(1) Put the whole template on the buffer, the stack is empty (the configuration A).
(2) Perform shifts until the two topmost elements on the stack are template expressions (3 SHIFTs were needed in our case, leading to the configuration B).
(3) `stackTop` is [`r.det#7`] and `stackTop2` is [`r.cop#6`], their common relchain is empty, hence the first merging condition fails. Perform a number of SHIFTs until the merging conditions succeed (2 SHIFTs in our case, leading to the configuration C).
(4) Now `stackTop` is [`r.nsubj#3`] and `stackTop2` is [`r.nsubj#3.amod#2`], their common relchain is `nsubj#3` and it differs from the relchains of both elements by at most one dependency relation. Both merging conditions are satisfied, perform a REDUCE and replace these two node-level expressions by one subtree-level expression containing their common relchain, that is, <`r.nsubj#3`>. Check if there are any more nodes in this subtree except the ones whose node-level expressions we just removed. In this case, `The`, `in` and `Brazil` belong to the same subtree, so we should subtract them. This results in the subtree-level template expression <`r.nsubj#3 - det#1 - nmod#5.case#4 - nmod#5`*> (dubbed <`TE3`>) and leads to the configuration D. REDUCE is finished, perform a SHIFT, leading to the configuration E.
(5) `stackTop` is [`r.nsubj#3.nmod#5.case#4`] and `stackTop2` is the result of the step (4). Their common relchain, `nsubj#3`, differs from relchain of `stackTop` by 2 dependency relations. Merging conditions failed, perform a SHIFT, leading to the configuration F.
(6) Now `stackTop2` is [`r.nsubj#3.nmod#5.case#4`] and `stackTop` is [`r.nsubj#3.nmod#5`], their common relchain is `nsubj#3.nmod#5`. Merging conditions are satisfied, the REDUCE action is triggered, replacing both node-level expressions

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

Sentence s_i The longest river in Brazil is the Amazon river
Question q_i What is the longest river in Brazil?

Result of the Sentence transformation (is used for Shift-reduce)

```
What [r.cop#6] [r.det#7] [r.nsubj#3 [r.nsubj#3] [r.nsubj#3 [r.nsubj#3
                           .amod#2]                .nmod#5     .nmod#5]
                                                   .case#4]
```

Shift-reduce

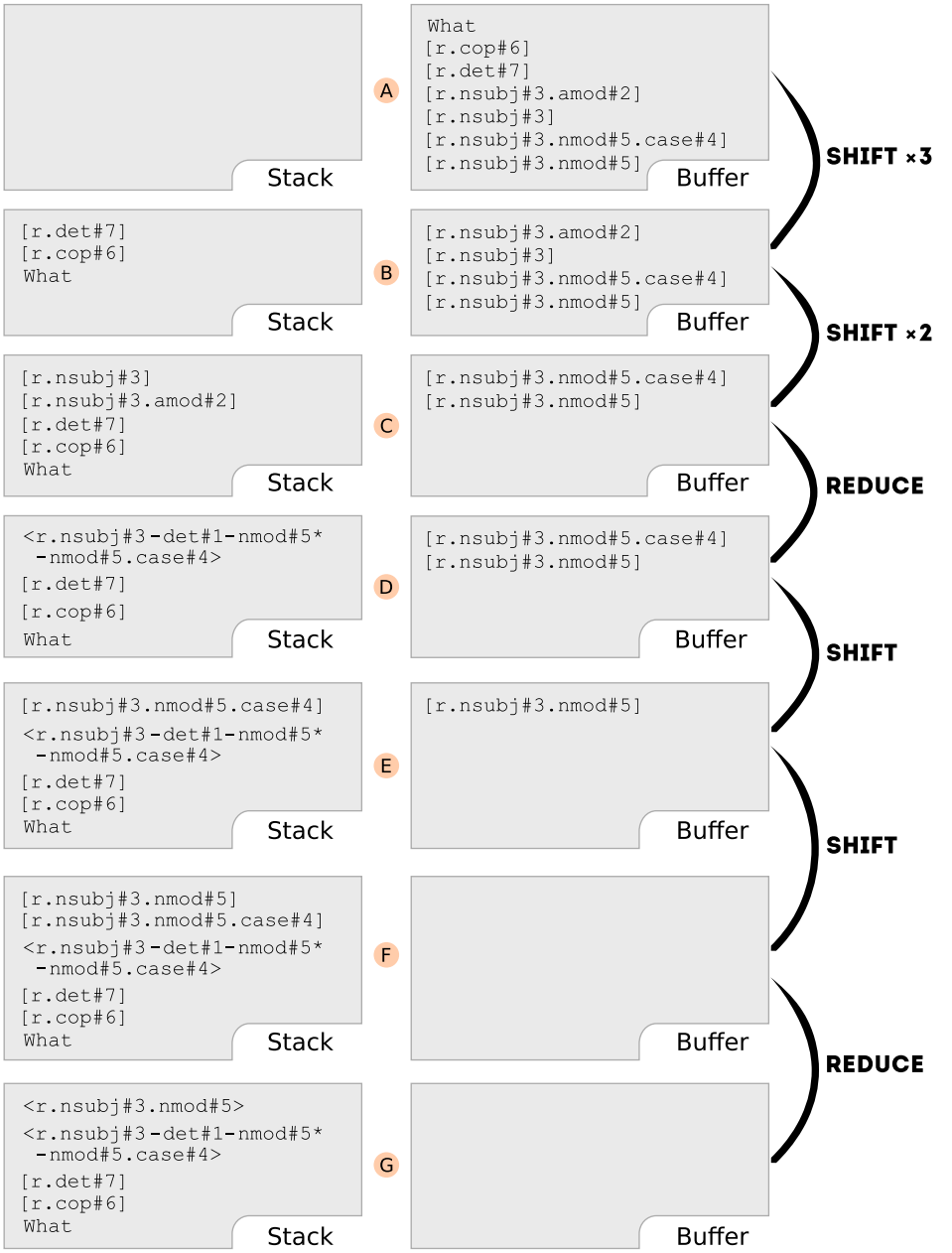

Figure 4: Illustration of the shift-reduce step on the question from the example QA-pair.

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

by <r.nsubj#3.nmod#5>. There are no unaccounted nodes in this subtree, hence there is nothing to subtract, leading to the configuration G. The buffer is empty, so the algorithm terminates. The resulting template (11) is on the stack *in the reversed order*.

(11) What [r.cop#6] [r.det#7] <TE3> <r.nsubj#3.nmod#5>?
(is) (the) (longest river) (in Brazil)

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 generalisable.

The step can be performed only if we have a template expression with more than one negative (otherwise there is nothing to merge). To perform this step, we need to create a mapping for the dependency tree of the original sentence between each node and its direct children, referred to as *a direct children mapping, DCM*. Then, for every template expression, check if any subset of negatives matches any set of children from the mapping. In case of a match, all matched negatives should be swapped for the corresponding subtree root.

To exemplify, consider template (11), obtained after shift-reduce in the previous section. There is only one template expression with negatives: <r.nsubj#3 - det#1 - nmod#5* - nmod#5.case#4>. Indeed, this template expression has 3 negatives: - det#1, - nmod#5* and - nmod#5.case#4. Hence, we can attempt merging negatives. The step of merging negatives for the template (11) is illustrated in Figure 5. The direct children mapping for every node in the original dependency tree is marked as 'DCM X'.

As mentioned previously, we can attempt merging negatives only on one template expression: <r.nsubj#3 - det#1 - nmod#5* - nmod#5.case#4>. Now we need to examine every DCM and check whether it contains any subset of negatives from the given template expression. Indeed, the mapping 'DCM 5' contains two negatives (- nmod#5*, and - nmod#5.case#4) marked with a red dashed line in Figure 5. Hence, these two expressions can be swapped for the expression of their subtree, nmod#5. The resulting template expression then becomes <r.nsubj#3 - det#1 - nmod#5> (dubbed <TE3'>) and the final induced template is template (12)

(12) What [r.cop#6] [r.det#7] <TE3'> <r.nsubj#3.nmod#5>?
(is) (the) (longest river) (in Brazil)

The advantage of the resulting template expression after merging negatives is that it makes no assumptions on the structure of a nominal modifier nmod#5. Indeed, the original template assumed that by subtracting nmod#5* and nmod#5.case#4, the whole subtree corresponding to nmod#5 is removed. While this is true for this particular example, nominal modifiers (and many other constructs) can, however, vary in structure and thus the template without merging negatives will generalise poorly to sentences with nominal modifiers (or other constructs) of different structure. To address this problem, the negatives of each template expression should be merged as much as possible to their common parent.

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.

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

Sentence s_i The longest river in Brazil is the Amazon river
Question q_i What is the longest river in Brazil?

Result of the Shift-reduce (is used for Merging negatives)

```
What [r.cop#6][r.det#7] <r.nsubj#3-det#1-nmod#5*-nmod#5.case#4> <r.nsubj#3.nmod#5>
```

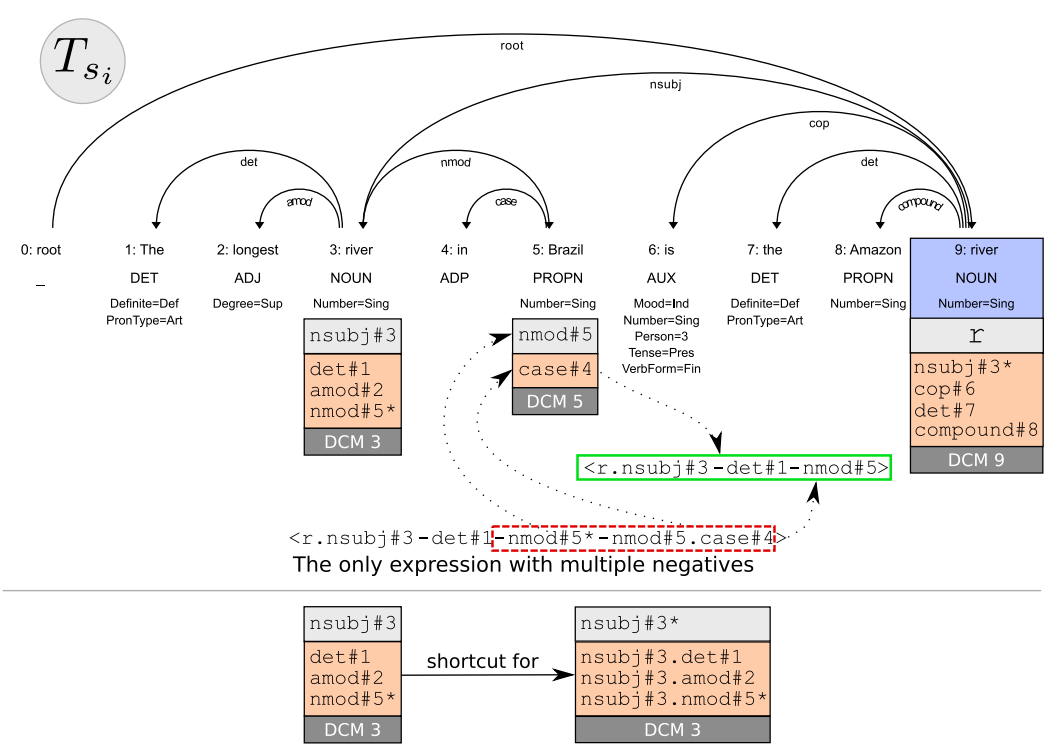

Figure 5: Illustration of merging negatives on the question from the example QA-pair.

Motivated by the example above, guards consist of base and complementary guards. A base guard contains the requirements for using templates and is created in 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.

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

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), and `n.morph has Number=Sing, n.pos is VERB` are the only complementary guards.

(13)
```
n.morph has Number=Sing, n.pos is NOUN, n.cop exists, n.det
exists, n.nsubj exists, n.nsubj.nmod exists, n.nsubj.morph
has_not PronType=Rel -> template12
```

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*(`NOUN/Number=Sing|DET/Definite=Def, 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, for example, (`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 basic filtering and involves two procedures. The first one aims at filtering out QA pairs with a single-word answer, whose pos-morph expression has never occurred in the training corpus as an answer. The procedure prevents generating single-word answers containing only function words (e.g., 'the', 'himself', 'to'). The second one ensures that a pair of question word and pos-morph expression of the root of the corresponding answer appeared in the training data at least once using a question-word model. This procedure contributes towards minimising the obvious mismatches between the question word and the answer (e.g., 'where'-question with a date as the answer).

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. In our experiments, $\lambda_4 = 0.000001, \lambda_3 = 0.01 - \lambda_4, \lambda_2 = 0.1 - \lambda_3, \lambda_1 = 0.9$. Note that the values of $\lambda_i, i \in [1, 4]$ can be estimated from data, as described by Jurafsky and Martin (2018, Section 3.5.3); however, we opted for setting them heuristically as those estimates will be unreliable for low-resourced languages with small amounts of textual data.

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

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}.

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

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

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

Language	QP position	Filtered training set	Filtered 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	14,771 (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

Equation (3) provides the final score, which is a convex combination between the scores calculated in Equations (1) and (2). For our experiments, we have performed a manual search for $\alpha \in [0.1, 0.9]$ and found that $\alpha = 0.8$ ranked grammatically correct questions with suitable answers highest more frequently than the other alternatives.

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

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 (or equal to) 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 utilised a data set called TyDi QA (Clark *et al.* 2020). The data set is a question-answering benchmark based on Wikipedia articles for 11 typologically diverse languages. Eight of these languages have available UD treebanks and trained dependency parsers in Stanza package (Qi *et al.* 2020), which we have utilised for inducing templates in all languages. For both training and evaluation, we have excluded Yes/No-questions resulting in the filtered 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.

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

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

5. Evaluation

Essentially, automatic evaluation metrics, such as BLEU (Papineni *et al.* 2002), ROUGE (Lin 2004), METEOR (Agarwal and Lavie 2008) and 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) reports 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. First, 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. Second, 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 standardised questionnaires and/or guidelines for human evaluation of automatically generated questions and answers. Amidei *et al.* (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 (Amidei *et al.* 2018, Table 7). 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 *et al.* (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.* (2021). 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) was calculated using the nlg-eval package (Sharma *et al.* 2017) and METEOR using METEOR-1.5 package[f] (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 subset of the TyDi QA data set containing 8 typologically diverse languages (see more information about the subset in Section 4). Different languages required somewhat different pre-processing steps, which are documented in Appendix B.

[f]METEOR-1.5 does not fully support all languages used in TyDi QA data set, so we set language to 'other' for all languages other than English.

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

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

Table 4. Descriptive statistics of the TyDi QA training data for different languages. Recall that 'satisfactory questions' have at least one word in common with the original sentence

	fi	ja	te	ar	id	ko	ru	en
Satisfactory questions (1)	74%	96%	68%	90%	83%	44%	62%	91%
Answer uses subwords (2)	9%	10%	53%	14%	5%	60%	10%	5%
(1) but not (2)	68%	86%	35%	78%	79%	19%	56%	87%

5.1.1. Automatic evaluation

We have evaluated Quinductor on all languages present in the TyDi QA data set 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 data set. Only the top-ranked generated question (if any) was considered for automatic evaluation with the respective automatic evaluation metrics reported in Table 3. The proportion of source sentences (the ones where the correct answer is found), for which Quinductor could generate anything at all is reported in Table 6.

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

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

Table 5. Descriptive statistics of the templates produced using the filtered training set of the TyDi QA data set for different languages. Support per template is the number of sentences from the training set that yield the same template

	fi	ja	te	ar	id	ko	ru	en
Number of induced templates	496	25	48	1104	340	9	85	254
Support per template								
Mean	1.22	1.08	1.06	1.1	1.14	1	1.09	1.03
Standard deviation	1.08	0.27	0.24	0.58	0.45	0	0.5	0.22
Median	1	1	1	1	1	1	1	1
Minimum	1	1	1	1	1	1	1	1
Maximum	17	2	2	13	4	1	5	4

Table 6. Descriptive statistics of the questions induced on the filtered development set of the TyDi QA data set for different languages using the templates, mentioned in Table 5. 'SS' stands for 'source sentence(s)', that is, the sentence in which the correct answer is found. '$\geq$ 1 applicable template' means that at least 1 question was induced from the given SS

(1) SS with $\geq$ 1 applicable template	928	263	152	904	693	92	345	569
(2) SS with $\geq$ 1 generated question after basic filtering	871	183	4	725	629	0	258	520
(3) SS with $\geq$ 1 generated question after mean filtering	611	97	4	462	558	0	93	409
(3) as % of the filtered dev. set	54%	14%	0.5%	46%	77%	0%	10%	64%
Generated questions per SS								
Mean	19.9	1.1	0.03	14.4	6.2	0	1.9	10.4
Standard deviation	30.7	1.3	0.16	29.6	6.2	0	2.7	12.8
Median	10	1	0	4	4	0	1	6
Minimum	0	0	0	0	0	0	0	0
Maximum	482	9	1	240	68	0	23	94

Note, however, that even with a very limited set of induced templates Quinductor managed to find source sentences (152 for Telugu and 92 for Korean, as reported in Table 6), for which at least one template was applicable (meaning the guard was passed and the question could be induced). However, the questions were not deemed to be of sufficient quality, since most of them did not pass even basic filtering. This highlights the important property of Quinductor: sometimes the method will choose not generate anything if the question is obviously of insufficient quality (which basic and mean filtering attempt to capture). This is in contrast to some neural question generation methods, which are often forced to generate something for all input data.

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). Note, however, that Quinductor managed to induce templates for both Russian and Japanese on a smaller subset of the development sets than for other languages, except Telugu and Korean (10% for Russian and 14% for Japanese). So these performance differences might be

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

attributed to the types of questions available in the data set and not to the Quinductor method itself.

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 (should be treated with caution, as mentioned previously). 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 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 data set and requires further investigation on other data sets.

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. Fifty 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.[g] 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.

[g]https://www.prolific.co/.

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

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 *et al.* (2019) are introduced in our approach.

First, 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 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.

Second, Goodman-Kruskall's γ (GK γ) was designed to measure rank correlation between ordinal judgements of two judges. Amidei *et al.* (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 generalisation 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 judgement pairs (i.e., ranked in the same order by judges i and j) or discordant judgement pairs (ranked in the reversed order by judges i and j).

Inter-annotator agreement (IAA) for the conducted human evaluations are reported in Table 7 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 *et al.* (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 A.3 in Appendix A 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 A.1 in Appendix A

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

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

Table 7. Inter-annotator agreement per criterion. Q stands for 'Question' and SA – for 'Suggested answer'

		en		fi		ru	
Criterion	IAA metric	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

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 8.

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 overinformative across all languages, and would not benefit from more information (except for English). Most of the cases for English 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

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

Table 8. Proportion of generated QA pairs where both median and mode are the same

		en		fi		ru	
Criterion	Best if	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%

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 overinformative, 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 summarised in Table 9. 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 10 and 11.

As can be seen in Table 11, 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 (see samples in Table 10).

5.2. Comparison to other methods on the SQuAD data set

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 data set (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[h] made by Du *et al.* (2017) and others relying on the split made by Zhou *et al.* (2018).

[h]The SQuAD split is available at https://github.com/xinyadu/nqg.

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

Table 9. 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 co-reference	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?

Table 10. 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

In this article use the split by Du *et al.* (2017), composed of 70,484 QA-pairs in the training set and 11,877 QA-pairs in the test set (note that we did not use development set in our experiments). Hence, we 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. Note that we use both the splits, the provided tokenizations and the extracted source sentences for each question as provided in the `data/processed` folder of the original repository. The default Quinductor implementation (dubbed QA-templates) also needs the correct answers, so we have extracted those from the provided raw files and followed the same tokenization procedure as described by Du *et al.* (2017). We have also tried inducing templates only for questions without correct answers (dubbed Q-templates). This is to see whether the induced templates will end up being more generalisable as a consequence of more relaxed guards (not having constraints related to the correct answer).

For SQuAD we have used only automatic evaluation metrics, since some articles did not perform human evaluation at all (Zhao *et al.* 2018; Song *et al.* 2018; Kim *et al.* 2019; Dong *et al.* 2019),

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

Table 11. Human judgements of the examples in Table 10. 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

Table 12. 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 *et al.* (2017). Note that the cases when Quinductor did not generate any question were excluded when calculating all metrics. Best results for methods and baselines from related work, as well as for Quinductor are indicated in bold

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 *et al.* (2017)	43.09	12.28	16.62	39.75
Bahuleyan *et al.* (2018)	30.87	5.08	NA	NA
Vanilla seq2seq[1]	31.34	4.26	9.88	29.75
H&S[2]	**38.50**	**11.18**	**15.95**	**30.98**
Ours (QA-templates)	**30.31**	**9.43**	16.53	31.19
Ours (Q-templates)	29.08	8.8	**17.87**	**32.27**

[1]The result for this model as reported by Du *et al.* (2017).
[2]The result for the model by Heilman and Smith (2009), as reported by Du *et al.* (2017).

and others (Du *et al.* 2017; Bahuleyan *et al.* 2018) used different criteria and evaluation guidelines making a fair comparison impossible.

Only the top-ranked generated question (if any) was considered for automatic evaluation. As can be seen our method performs better than all baselines reported by Du *et al.* (2017) (shown in italics in Table 12) in terms of METEOR and ROUGE-L, and substantially better on BLEU-4 compared to the vanilla seq2seq model reported by Du *et al.* (2017). Indeed, this cements the position of our as a rather strong baseline that could be used as a point of reference for languages other than English. However, the performance lags behind to a substantial degree, compared to most other methods reported in literature, which is, again, not an unexpected quality for the baseline method.

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

Table 13. Descriptive statistics of the templates produced using the training set of the SQuAD data set (split by Du *et al.* 2017). Support per template is the number of sentences from the training set that yield the same template. 'SD' stands for 'standard deviation'

	Ours (QA-templates)	Ours (Q-templates)
Number of induced templates	5549	6219
Support per template		
Mean (SD)	1.06 (0.64)	1.05 (0.41)
Median (Min-Max)	1 (1-35)	1 (1-14)

Table 14. Descriptive statistics of the questions induced on the test set of the SQuAD data set (split by (Du *et al.* 2017)) using templates, mentioned in Table 13. 'SS' stands for 'source sentence(s)', that is, the sentence in which the correct answer is found. '$\geq$ 1 applicable template' means that at least 1 question was induced from the given SS. 'SD' stands for 'standard deviation'

	Ours (QA-templates)	Ours (Q-templates)
(1) SS with $\geq$ 1 applicable template	6472	6588
(2) SS with $\geq$ 1 generated question after basic filtering	6256	6588
(3) SS with $\geq$ 1 generated question after mean filtering	5656	6439
(3) as % of the test set	47.62%	54.21%
Generated questions per SS		
Mean (SD)	70.5 (130.0)	183.3 (217.0)
Median (Min-Max)	27 (0-2282)	116 (1-4971)

When comparing two variations of Quinductor itself, we see no substantial performance differences. We note, however, that the variation with Q-templates induced substantially more templates with a roughly similar support per template on average. The version with Q-templates also managed to apply templates to about 6.5 percentage points more source sentences than the default version (as can be seen in Table 14), which might explain a slight dip in BLEU scores in Table 12. We note that METEOR score was slightly higher for the variation with Q-templates, indicating that the matches between the generated and the gold questions are slightly better aligned. ROUGE-L score were also slightly higher, indicating that the longest common subsequences between the generated and gold questions were slightly longer. These two observations along with the fact that the variation with Q-templates managed to generate questions for substantially more source sentences, hint that this variation might be a viable alternative if only questions are of interest.

5.3. Comparison to other methods on the Monserrate benchmark

Many data sets used for benchmarking in QG provide only a few questions to be evaluated against for each source passage/sentence. This is problematic, as discussed previously, since one can pose many valid questions to each and every sentence (and many more to a passage as a whole). Rodrigues, Nyberg, and Coheur (2022) contribute towards overcoming this limitation by releasing the Monserrate QG benchmark, which attempts at providing an exhaustive list of possible questions for each source sentence. In this setting, the previously discussed automatic evaluation metrics become clearly more relevant, since every generated question will be compared to

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

Table 15. Comparison to state-of-the-art QG methods on the Monserrate benchmark. Note that the cases when Quinductor did not generate any question were included when calculating all metrics. Best results for methods from related works and Quinductor are indicated in bold

System	BLEU-1	BLEU-4	METEOR	ROUGE-L
Heilman and Smith (2009)[1]	83.71	**45.56**	46.38	**69.00**
Du *et al.* (2017)[1]	77.40	26.63	37.58	63.71
GEN SubT[1]	**95.93**	28.45	30.67	60.60
GEN FlexSubT[1]	86.91	35.25	40.63	64.66
GEN Args[1]	81.80	40.61	**46.44**	65.81
Ours (QA-templates)	63.70	27.70	31.90	48.81
Ours (Q-templates)	**79.81**	**38.69**	**40.56**	**59.88**

[1]The result for this model as reported by Rodrigues *et al.* (2022).

Table 16. Descriptive statistics of the questions induced on the Monserrate benchmark using templates, mentioned in Table 13. 'SS' stands for 'source sentence(s)', that is, the sentence in which the correct answer is found. '$\geq$ 1 applicable template' means that at least 1 question was induced from the given SS. 'SD' stands for 'standard deviation'

	Ours (QA-templates)	Ours (Q-templates)
(1) SS with $\geq$ 1 applicable template	71	72
(2) SS with $\geq$ 1 generated question after basic filtering	65	72
(3) SS with $\geq$ 1 generated question after mean filtering	58	71
(3) as % of the test set	79.45%	97.26%
Generated questions per SS		
Mean (SD)	73.86 (130.9)	177.4 (207.91)
Median (Min-Max)	27 (0-768)	96 (2-1207)

more than 1 alternative (on average 26 alternatives in the Monserrate corpus). We have applied Quinductor's guards and templates generated on the SQuAD data set (which was also used for the evaluation in Section 5.2). The results for both QA-templates and Q-templates are reported in Table 15. For calculating the evaluation metrics, we have used the `runMaluuba.py` script provided in the GitHub repository for Monserrate.[i] Only the top-ranked generated question (if any) was considered for automatic evaluation. Note that, contrary to Table 12, we include the cases when Quinductor did not generate any questions when calculating the metrics reported in Table 15. This is because we did not manage to make the evaluation script run without errors otherwise, which also gives us confidence that the metrics reported in Rodrigues *et al.* (2022) on this benchmark were obtained in this way (contrary to the results in Table 12). As previously, we report the descriptive statistics of the induced questions in Table 16.

As can be seen from Table 15, Quinductor performs much better on this benchmark with the less-restrictive Q-templates than QA-templates. In fact, Table 16 shows that the Quinductor variation with QA-templates generated questions for 79.45% of the source sentences, which is 17.81

[i]https://github.com/hprodrig/MONSERRATE_Corpus.

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

percentage points fewer than the one with Q-templates. This shows the conservative side of our default approach, where we prefer not to generate a QA-pair at all if either question or answer are deemed of insufficient quality. Evidently, generating only questions appears to be easier than QA-pairs.

Notably, the Quinductor variation with QA-templates performs worse than most other methods on all metrics (except METEOR for GEN SubT system). The variation with Q-templates performs substantially better, and in fact outperforms the system by Du *et al.* (2017) on all metrics except ROUGE-L. Both the system by Heilman and Smith (2009) and all versions of GEN system perform better than Quinductor with Q-templates in BLEU-1 and ROUGE-L. However, Quinductor with Q-templates performs better or on par with GEN SubT and GEN FlexSubT and is not that far behind the other systems in terms of BLEU-4 and METEOR, proving Quinductor to be a reasonably strong baseline method. Bearing in mind that the system by Heilman and Smith (2009), as well as the GEN system were both designed with English-specific knowledge or components (e.g., a named-entity recognizer), it is not unexpected that their performance is better than Quinductor's. Even though Quinductor uses no language-specific knowledge or components, it yet demonstrates a performance on par or not far behind in terms of BLEU-4 and METEOR, making Quinductor a promising candidate for the baseline QG method for languages other than English.

6. Discussion

We have shown that the Quinductor method is a strong baseline method as concerns automatic evaluation metrics, 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 data sets relying on subwords in either questions or answers are proven to not work well with Quinductor (e.g., Korean and Telugu in TyDi QA data set). 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 data set 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 factor of Quinductor is that every generated question will be based on a single sentence. Thereby, the questions will be of the information-locating kind, which is the basic and least advanced rung on the reading comprehension ladder (OECD 2019, p. 34). However, the ability of quickly scanning a text and locating the answer to a specific question is a very important component in reading comprehension (Dreher and Guthrie 1990; Moore 1995; Rouet and Coutelet 2008). It is also a non-trivial task for second-language learners, as it requires knowledge of decoding, the vocabulary, and the grammar of the language to be learnt, all important aspects

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

of second-language reading comprehension (Jeon and Yamashita 2014). We therefore believe that the questions generated by Quinductor can be highly useful in language teaching and training.

It might be said that Quinductor still requires the use of a dependency parser to be trained on a sizeable data set, and thereby moving the problem rather than solving it. However, first, the Universal Dependencies framework includes 200 treebanks for over 100 languages (and counting). Second, we have shown that Quinductor could induce templates even for Telugu, whose dependency parser is trained on a treebank with only 6K tokens. Third, 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.

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. We are very grateful to the anonymous reviewers for their detailed and insightful comments that helped improving this article substantially.

Conflicts of interest. The authors declare none.

References

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

Agarwal A. and Lavie A. (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*, Columbus, Ohio. Association for Computational Linguistics, pp. 115–118.

Agarwal M., Shah R. and Mannem P. (2011). Automatic question generation using discourse cues. In *Proceedings of the Sixth Workshop on Innovative Use of NLP for Building Educational Applications*, Portland, Oregon. Association for Computational Linguistics, pp. 1–9.

Amidei J., Piwek P. and Willis A. (2018). Evaluation methodologies in automatic question generation 2013-2018. In *Proceedings of the 11th International Conference on Natural Language Generation, Tilburg University*, The Netherlands. Association for Computational Linguistics, pp. 307–317.

Amidei J., Piwek P., and **Willis A.** (2019). Agreement is overrated: A plea for correlation to assess human evaluation reliability. In *Proceedings of the 12th International Conference on Natural Language Generation*, Tokyo, Japan. Association for Computational Linguistics, pp. 344–354.

Bahuleyan H., Mou L., Vechtomova O. and Poupart P. (2018). Variational attention for sequence-to-sequence models. In *Proceedings of the 27th International Conference on Computational Linguistics*, Santa Fe, New Mexico, USA. Association for Computational Linguistics, pp. 1672–1682.

Baker C.F., Fillmore C.J. and Lowe J.B. (1998). The Berkeley FrameNet project. In *COLING 1998 Volume 1: The 17th International Conference on Computational Linguistics*.

Bernhard D., De Viron L., Moriceau V. and Tannier X. (2012). Question generation for french: Collating parsers and paraphrasing questions. *Dialogue & Discourse* **3**(2), 43–74.

Blaikie N. (2003). *Analyzing Quantitative Data: From Description to Explanation*. London: Sage.

Brown T., Mann B., Ryder N., Subbiah M., Kaplan J.D., Dhariwal P., Neelakantan A., Shyam P., Sastry G., Askell A., Agarwal S., Herbert-Voss A., Krueger G., Henighan T., Child R., Ramesh A., Ziegler D., Wu J., Winter C., Hesse C., Chen M., Sigler E., Litwin M., Gray S., Chess B., Clark J., Berner C., McCandlish S., Radford A., Sutskever I., and **Amodei D.** (2020). Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., pp. 1877–1901.

Callison-Burch C., Osborne M. and Koehn P. (2006). Re-evaluating the role of Bleu in machine translation research. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy. Association for Computational Linguistics, pp. 249–256.

Chan Y.-H. and Fan Y.-C. (2019). A recurrent BERT-based model for question generation. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, Hong Kong, China. Association for Computational Linguistics, pp. 154–162.

Clark J.H., Choi E., Collins M., Garrette D., Kwiatkowski T., Nikolaev V. and Palomaki J. (2020). TyDi QA: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics* **8**, 454–470.

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

Denkowski M. and Lavie A. (2014). Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, Baltimore, Maryland, USA. Association for Computational Linguistics, pp. 376–380.

Devlin J., **Chang M.-W.**, **Lee K. and Toutanova K.** (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota. Association for Computational Linguistics, pp. 4171–4186.

Dong L., **Yang N.**, **Wang W.**, **Wei F.**, **Liu X.**, **Wang Y.**, **Gao J.**, **Zhou M. and Hon H.-W.** (2019). Unified language model pre-training for natural language understanding and generation. 32.

Dreher M.J. and Guthrie J.T. (1990). Cognitive processes in textbook chapter search tasks. *Reading Research Quarterly* **25**(4), 323–339.

Dryer M.S. (2005). 93 position of interrogative phrases in content questions.

Du X., **Shao J. and Cardie C.** (2017). Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vancouver, Canada. Association for Computational Linguistics, pp. 1342–1352.

Fleiss J.L. (1971). Measuring nominal scale agreement among many raters. *Psychological Bulletin* **76**(5), 378.

Gates D.M. (2011). How to generate cloze questions from definitions: A syntactic approach. In *2011 AAAI Fall Symposium Series*.

Gatt A. and Krahmer E. (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 L.A. and Kruskal W.H. (1979). Measures of association for cross classifications, pp. 2–34.

Heilman M. and Smith N.A. (2009). Question generation via overgenerating transformations and ranking. Technical report, Language Technologies Institute, Carnegie Mellon University, CMU-LTI-09-013.

Jeon E.H. and Yamashita J. (2014). L2 reading comprehension and its correlates: A meta-analysis. *Language Learning* **64**(1), 160–212.

Jurafsky D. and Martin J.H. (2018). Speech and language processing (draft). Preparation [cited 2020 June 1] Available from: https://web.stanford.edu/~jurafsky/slp3.

Kalpakchi D. and Boye J. (2020). UDon2: A library for manipulating Universal Dependencies trees. In *Proceedings of the Fourth Workshop on Universal Dependencies (UDW 2020)*, Barcelona, Spain (Online). Association for Computational Linguistics, pp. 120–125.

Khullar P., **Rachna K.**, **Hase M. and Shrivastava M.** (2018). Automatic question generation using relative pronouns and adverbs. In *Proceedings of ACL 2018, Student Research Workshop*, Melbourne, Australia. Association for Computational Linguistics, pp. 153–158.

Kim Y., **Lee H.**, **Shin J. and Jung K.** (2019). Improving neural question generation using answer separation. *Proceedings of the AAAI Conference on Artificial Intelligence* **33**(01), 6602–6609.

Kipper K., **Dang H.T. and Palmer M.** (2000). Class-based construction of a verb lexicon. In *AAAI/IAAI*, pp. 691–696.

Kumar V., **Joshi N.**, **Mukherjee A.**, **Ramakrishnan G. and Jyothi P.** (2019). Cross-lingual training for automatic question generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy. Association for Computational Linguistics, pp. 4863–4872.

Landis J.R. and Koch G.G. (1977). The measurement of observer agreement for categorical data. *Biometrics* **33**(1), 159–174.

Liao Y., **Jiang X. and Liu Q.** (2020). Probabilistically masked language model capable of autoregressive generation in arbitrary word order. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online. Association for Computational Linguistics, pp. 263–274.

Lin C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, Barcelona, Spain. Association for Computational Linguistics, pp. 74–81.

Liu B., **Zhao M.**, **Niu D.**, **Lai K.**, **He Y.**, **Wei H.**, and **Xu Y.** (2019). Learning to generate questions by learningwhat not to generate. In *The World Wide Web Conference*, WWW'19, New York, NY, USA. Association for Computing Machinery, pp. 1106–1118.

Mazidi K. and Nielsen R.D. (2015). Leveraging multiple views of text for automatic question generation. In *International Conference on Artificial Intelligence in Education*, Cham. Springer International Publishing, pp. 257–266.

Miller G.A. (1995). Wordnet: A lexical database for english. *Communications of the ACM* **38**(11), 39–41.

Moore P. (1995). Information problem solving: A wider view of library skills. *Contemporary Educational Psychology* **20**(1), 1–31.

Mostow J. and Jang H. (2012). Generating diagnostic multiple choice comprehension cloze questions. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, Montréal, Canada. Association for Computational Linguistics, pp. 136–146.

Nema P. and Khapra M.M. (2018). Towards a better metric for evaluating question generation systems. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium. Association for Computational Linguistics, pp. 3950–3959.

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

Nivre J., **de Marneffe M.-C.**, **Ginter F.**, **Hajič J.**, **Manning C.D.**, **Pyysalo S.**, **Schuster S.**, **Tyers F. and Zeman D.** (2020). Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, Marseille, France. European Language Resources Association, pp. 4034–4043.

OECD (2019). *PISA 2018 Assessment and Analytical Framework*. Paris: PISA, OECD Publishing.

Papineni K., **Roukos S.**, **Ward T. and Zhu W.-J.** (2002). Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics, pp. 311–318.

Qi P., **Zhang Y.**, **Zhang Y.**, **Bolton J. and Manning C.D.** (2020). Stanza: A python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Online. Association for Computational Linguistics, pp. 101–108.

Rajpurkar P., **Zhang J.**, **Lopyrev K. and Liang P.** (2016). SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, Texas. Association for Computational Linguistics, pp. 2383–2392.

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

Rodrigues H., **Nyberg E. and Coheur L.** (2022). Towards the benchmarking of question generation: Introducing the monserrate corpus. *Language Resources and Evaluation* **56**(2), 573–591.

Rodrigues H.P. (2020). *Learning Semantic Patterns for Question Generation*. PhD Thesis, Carnegie Mellon University.

Rosenthal J.A. (1996). Qualitative descriptors of strength of association and effect size. *Journal of Social Service Research* **21**(4), 37–59.

Rouet J.-F. and Coutelet B. (2008). The acquisition of document search strategies in grade school students. *Applied Cognitive Psychology: The Official Journal of the Society for Applied Research in Memory and Cognition* **22**(3), 389–406.

Rus V., **Cai Z. and Graesser A.** (2008). Question generation: Example of a multi-year evaluation campaign. *Proc WS on the QGSTEC.*

Rus V., **Wyse B.**, **Piwek P.**, **Lintean M.**, **Stoyanchev S. and Moldovan C.** (2010). Overview of the first question generation shared task evaluation challenge. In *Proceedings of the Third Workshop on Question Generation*, pp. 45–57.

Sharma S., **El Asri L.**, **Schulz H. and Zumer J.** (2017). Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation. CoRR, abs/1706.09799.

Song L., **Wang Z.**, **Hamza W.**, **Zhang Y. and Gildea D.** (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)*, New Orleans, Louisiana. Association for Computational Linguistics, pp. 569–574.

van der Lee C., **Gatt A.**, **van Miltenburg E. and Krahmer E.** (2021). Human evaluation of automatically generated text: Current trends and best practice guidelines. *Computer Speech & Language* **67**, 101151.

Vedantam R., **Lawrence Zitnick C. and Parikh D.** (2015). Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Zhao Y., **Ni X.**, **Ding Y. and Ke Q.** (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*, Brussels, Belgium. Association for Computational Linguistics, pp. 3901–3910.

Zhou Q., **Yang N.**, **Wei F.**, **Tan C.**, **Bao H. and Zhou M.** (2018). Neural question generation from text: A preliminary study. In *Natural Language Processing and Chinese Computing*, Cham. Springer International Publishing, pp. 662–671.

Appendix A. Human evaluation details

Human evaluation has been conducted on the Prolific platform.[j] 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 A.2, A.4, A.5. 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 labelled 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.

[j] https://www.prolific.co/.

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

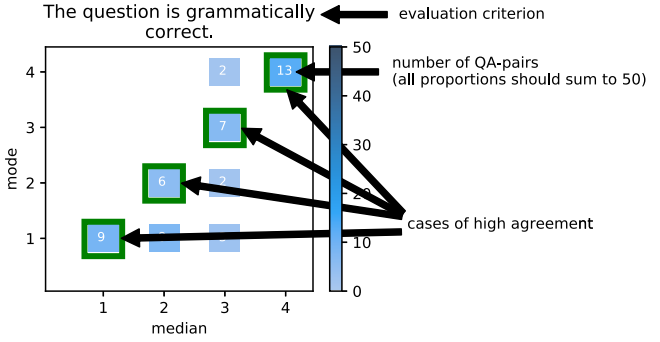

Figure A.1 An annotated example of a bi-variate histogram.

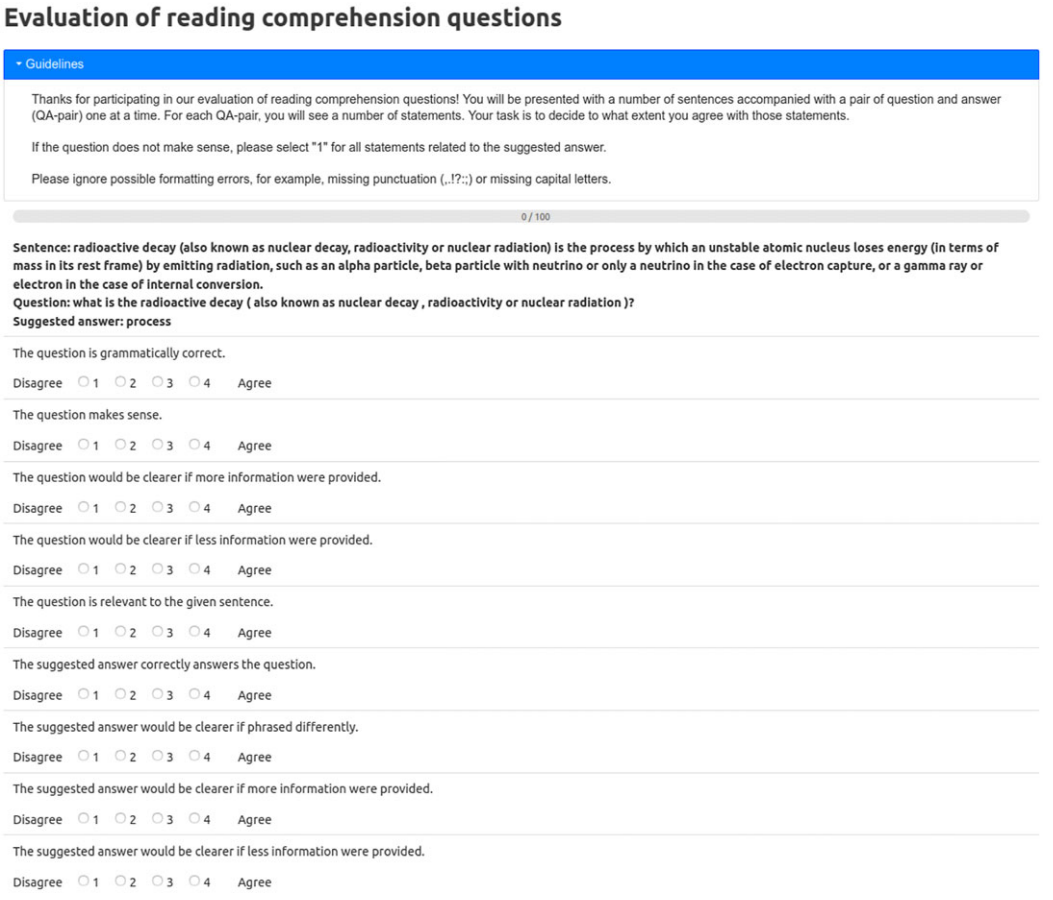

Figure A.2 Evaluation guidelines and questionnaire for English.

The detailed results of human judgements for questions in English, Finnish and Russian are presented in Figure A.3. To aid reader in understanding the presentation format, we provide an annotated histogram in Figure A.1.

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

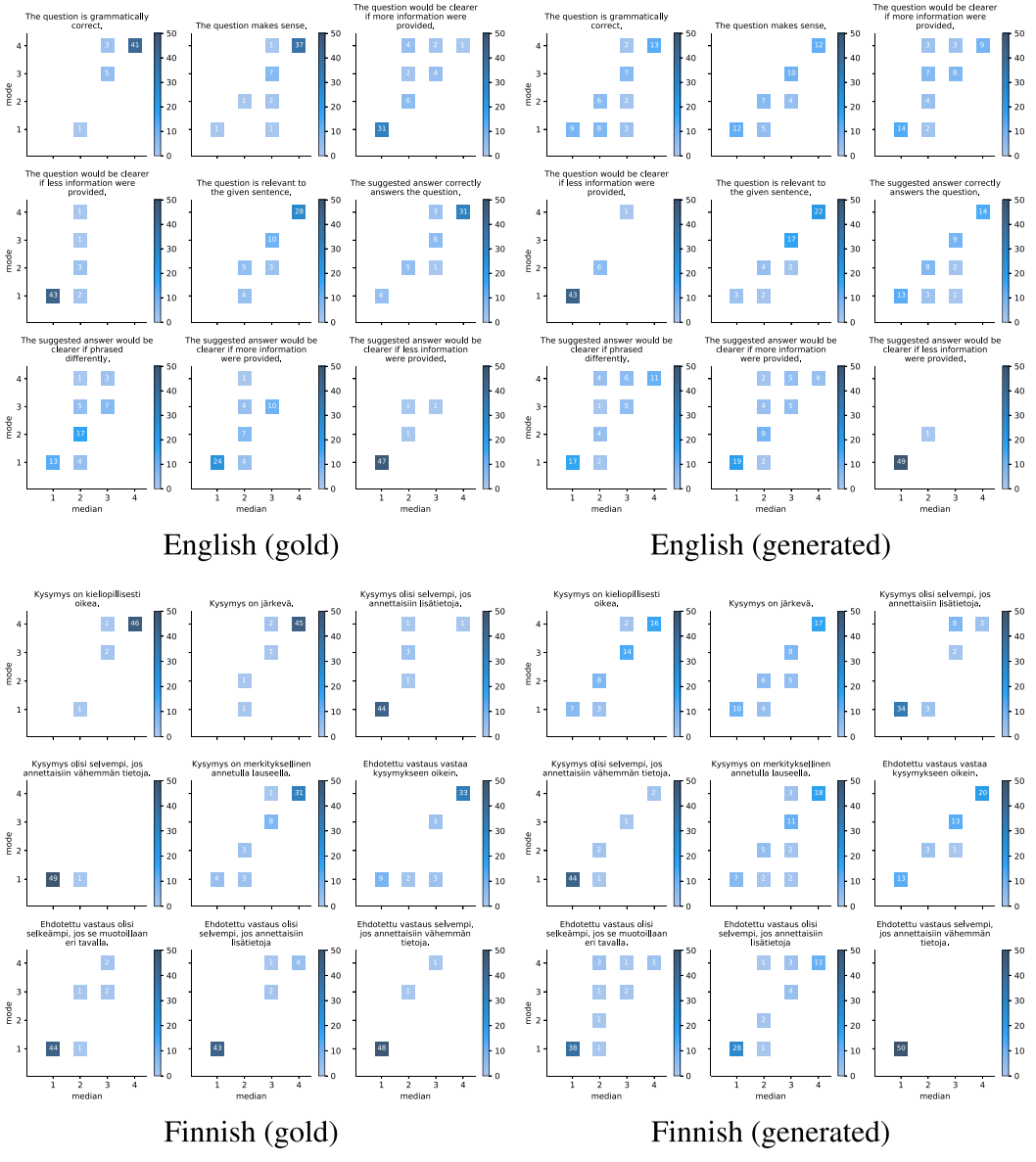

Figure A.3 Bi-variate histograms of human judgements (the order of criteria is the same for all languages).

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

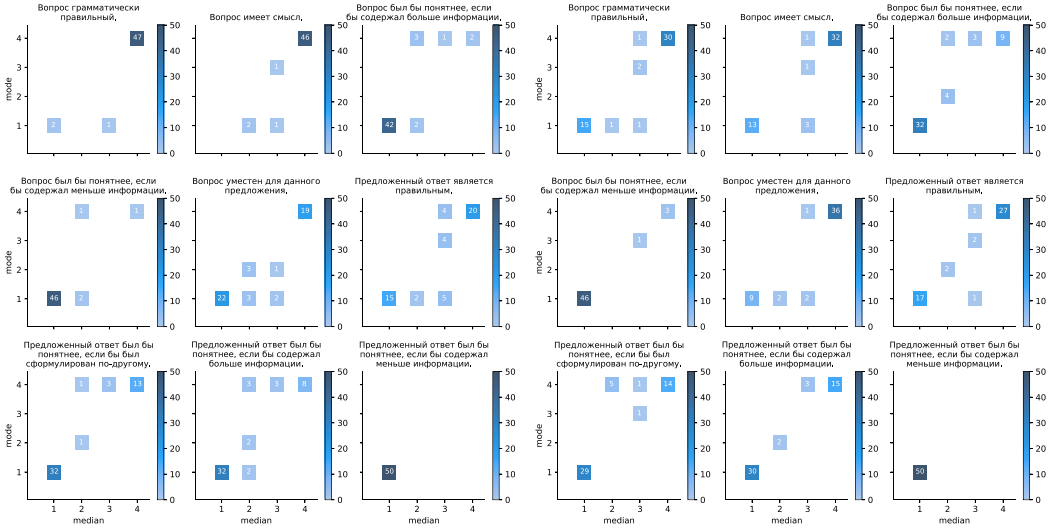

Figure A.3 Continued.

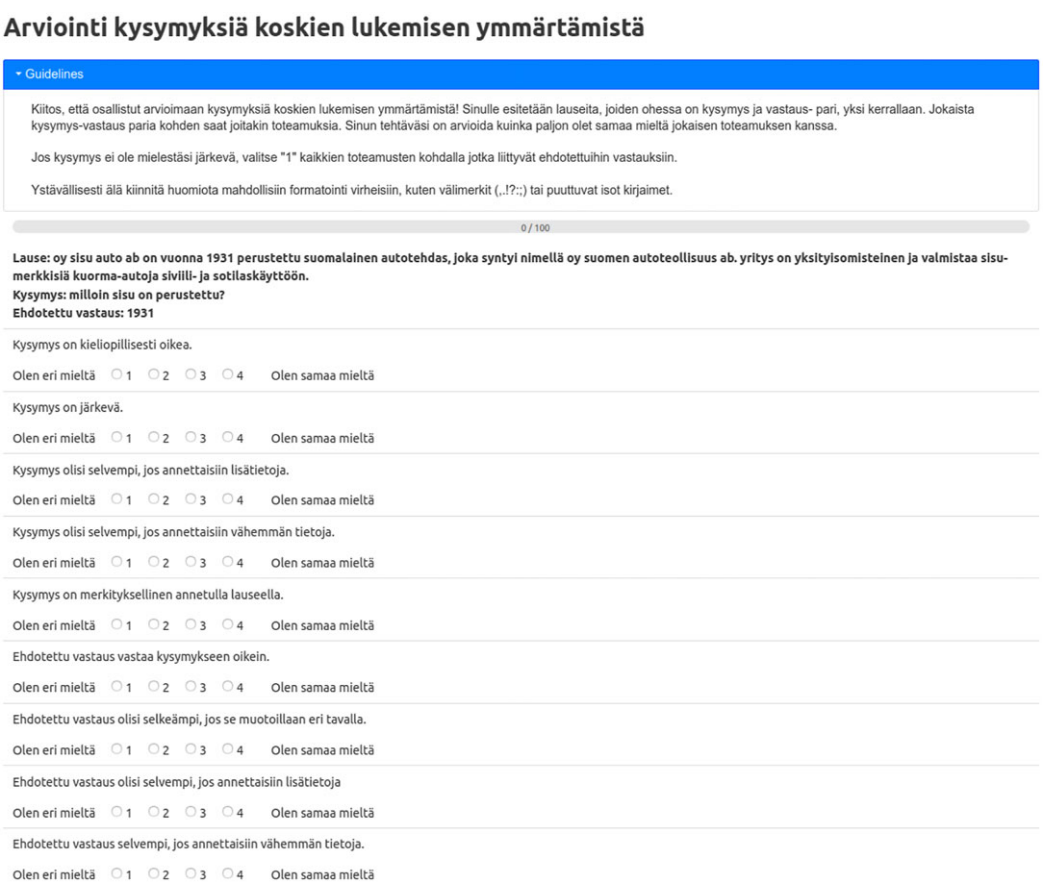

Figure A.4 Evaluation guidelines and questionnaire for Finnish.

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

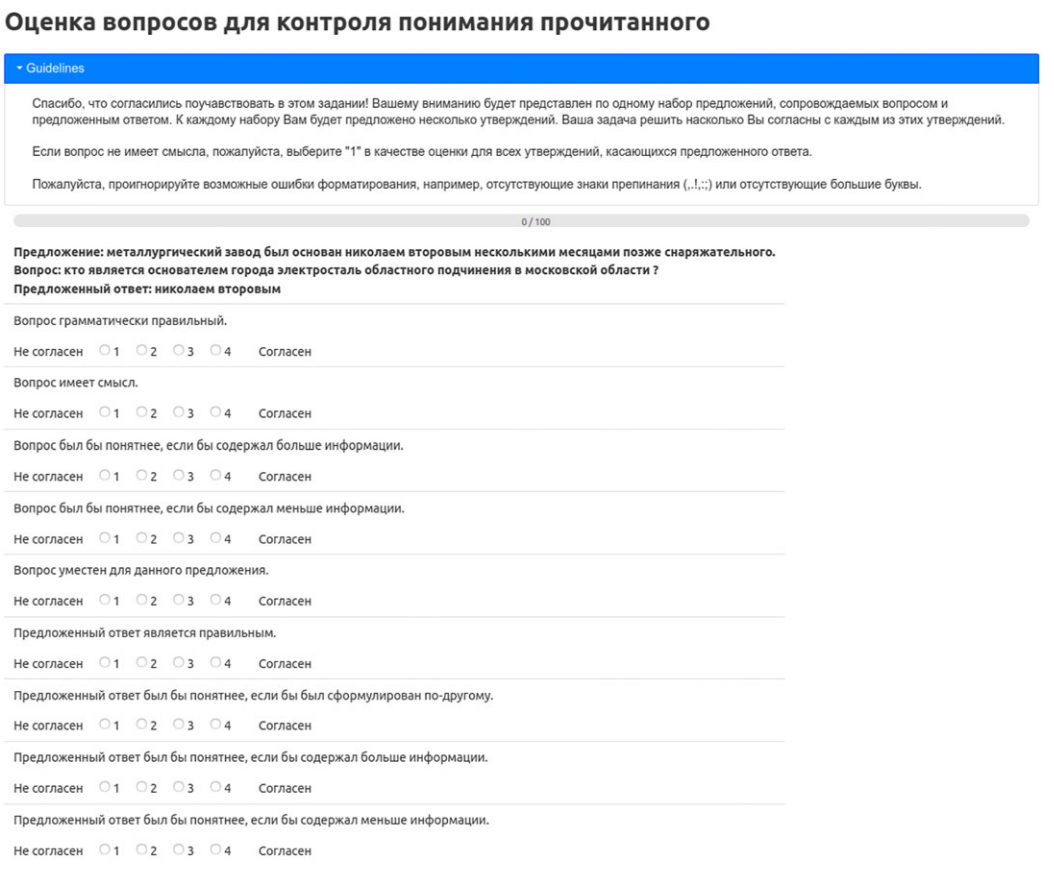

Figure A.5 Evaluation guidelines and questionnaire for Russian.

Appendix B. Data pre-processing steps

The applied pre-processing steps for every language are specified in Table B.1. Note that Arabic is written from right to left, but we found that processing sentences from left to right yielded better results with Quinductor.

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

Preprocessing step	fi	ja	te	ar	id	ko	ru	en
Lowercase	✔	NA	NA	NA	✔	NA	✔	✔
Remove punctuation	✘	✔	✔	✔	✘	✘	✘	✘
Remove diacritics	✘	✘	✘	✘	✘	✘	✔	✘
Processing direction	S	E	E	S	S	E	S	S

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

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 that 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:

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

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

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.

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

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

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

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

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

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

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

Suggested: 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

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

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?

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press

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

Cite this article: Kalpakchi D and Boye J (2024). Quinductor: A multilingual data-driven method for generating reading-comprehension questions using Universal Dependencies. *Natural Language Engineering* **30**, 217–255. https://doi.org/10.1017/S1351324923000037

https://doi.org/10.1017/S1351324923000037 Published online by Cambridge University Press