Extracting Simplified Statements for Factual Question Generation

Michael Heilman and Noah A. Smith

Language Technologies Institute, Carnegie Mellon University
Pittsburgh, PA 15221, USA
{mheilman,nasmith}@cs.cmu.edu

Abstract. We address the problem of automatically generating concise factual questions from linguistically complex sentences in reading materials. We discuss semantic and pragmatic issues that appear in complex sentences, and then we present an algorithm for extracting simplified sentences from appositives, subordinate clauses, and other constructions. We conjecture that our method is useful as a preliminary step in a larger question generation process. Experimental results indicate that our method is more suitable for factual question generation applications than an alternative text compression algorithm.

1 Introduction

This paper addresses question generation (QG) for the purpose of creating reading assessments about the factual information that is present in expository texts such as encyclopedia articles and textbooks. We believe that QG has the potential to help teachers efficiently assess students' acquisition of basic factual knowledge from reading materials, thereby enabling teachers to focus on more complex questions and learning activities. We also believe that research on factual QG can inform research on the generation of more complex reading questions as well as questions for other applications, some of which are described in [17].

Factual questions about reading materials, either from teachers or students, are usually short and targeted at a single piece of information. For example, in the Microsoft Research Question-Answering Corpus,[1] which consists of questions generated by students and excerpts from encyclopedia articles that answer them, the average length of questions is 7.2 words. However, sentences in texts are typically much longer on average because various constructions such as conjunctions, subordinate clauses, and appositives enable writers to convey multiple pieces of information in a single sentence. Consider Ex. 1, which contains a non-restrictive appositive (*the country's paramount leader*), a participial phrase (*returning ...*), and two clauses conjoined by *so*.[2] An important issue in QG is how to generate concise questions from these sorts of complex sentences.

[1] Available at http://research.microsoft.com.

[2] "Subway Blasts Kill Dozens in Moscow" by C. Levy, *New York Times*, downloaded March 29, 2010.

(1) *Prime Minister Vladimir V. Putin, the country's paramount leader, cut short a trip to Siberia, returning to Moscow to oversee the federal response. Mr. Putin built his reputation in part on his success at suppressing terrorism, so the attacks could be considered a challenge to his stature.*

One possible solution is to use techniques from text compression [11, 6]. The goal of compression is to take as input a possibly long and complex sentence, and produce as output a single shortened version that conveys the main piece of information in the input. However, as noted above, sentences often convey multiple pieces of information. In QG, we may want to generate questions not just about the information in the main clause, but also about the information embedded in various nested constructions. For example, from the two sentences in Ex. 1, we might produce the following sentences to allow a QG system to generate a more comprehensive set of questions:[3]

(2) *Prime Minister Vladimir V. Putin is the country's paramount leader.*
(3) *Prime Minister Vladimir V. Putin cut short a trip to Siberia.*
(4) *Prime Minister Vladimir V. Putin returned to Moscow to oversee the federal response.*
(5) *Mr. Putin built his reputation in part on his success at suppressing terrorism.*
(6) *The attacks could be considered a challenge to his stature.*

In this paper, we present a method for extracting multiple, simple, syntactically and semantically correct factual statements from complex sentences. These simple sentences can be readily transformed into questions. There already exist frameworks where questions are generated through transformations of extracted declarative sentences, notably [8, 9]. Our method fits particularly well into sentence-extraction stage of that approach, but could also serve as a preprocessing step for other QG systems.[4] Our method is linguistically motivated by research on semantic and pragmatic phenomena. We present the results of a small-scale study of the quality of the output of our system and compare to a text compression approach [6].

2 Extraction of Textual Entailments

The task of extracting simple sentences from a complex input sentence is essentially the task of generating a particular subset of the possible sentences that a reader would assume to be true after reading the input. That is, we aim to generate a restricted set of textual entailments, using the informal definition of

[3] Ideally, we would also like to resolve anaphora such as *his*. Due to the difficulties of coreference resolution, we leave that to future work.

[4] We consider the set of simplification rules described in [8, 9] to be an early prototype of the more complete system described here. We did not expect a formal experimental comparison to that prototype to be informative.

entailment from the *recognizing* textual entailment task [7, 1]. While it is important to generate outputs that are true given an input sentence, we are not satisfied with outputs that merely preserve meaning. Rather, we seek short sentences such as Exs. 2–6 that are likely to lead to concise questions.

Our method for extracting meaning-preserving, simplified sentences depends on two linguistic phenomena: semantic entailment and presupposition (both of which we subsume into the informal notion of textual entailment).

3 Extraction and Simplification by Semantic Entailment

Semantic entailment is defined as follows: A **semantically entails** B if and only if for every situation in which A is true, B is also true [12].

This section describes how we extract simplifications from complex sentences by removing adjunct modifiers and discourse connectives and by splitting conjunctions of clauses and verb phrases. These transformations preserve the truth conditions of the original input sentence, while producing more concise sentences from which questions can be generated.

3.1 Removing Discourse Markers and Adjunct Modifiers

Many sentences can be simplified by removing certain adjunct modifiers from clauses, verb phrases, and noun phrases. For example, from Ex. 7, we can extract Ex. 8 by removing the discourse marker *however* and the relative clause *which restricted trade with Europe.*

(7) *However, Jefferson did not believe the Embargo Act, which restricted trade with Europe, would hurt the American economy.*
(8) *Jefferson did not believe the Embargo Act would hurt the American economy.*

Ex. 8 is true in all situations where Ex. 7 is true, and is therefore semantically entailed. Discourse markers such as *however* do not affect truth conditions in and of themselves, but rather serve to inform a reader about how the current sentence relates to the preceding discourse. Adjunct modifiers do convey meaning, but again do not affect semantic entailment. Of course, many adjuncts provide useful information that we should preserve for later QG steps. For example, prepositional phrases that identify the locations and times at which events occurred can be the target of *where* and *when* questions, respectively.

Our algorithm (presented in §5) extracts simplified, entailed statements by removing the following adjuncts and discourse markers:

– non-restrictive appositives (e.g., *Jefferson, the third U.S. President, ...*)[5]
– non-restrictive relative clauses (e.g., *Jefferson, who was the third U.S. President, ...*)
– parentheticals (e.g., *Jefferson (1743–1826) ...*)
– participial modifiers of noun phrases (e.g., *Jefferson, being the third U.S. President, ...*)
– verb phrase modifiers offset by commas (e.g., *Jefferson studied many subjects, like the artist Da Vinci. ...*)
– modifiers that precede the subject (e.g., *Being the third U.S. President, Jefferson...*)

We only remove verb phrase modifiers that are offset by commas (e.g., we simplify *When I talked to him, John was busy.* but not *John was busy when I talked to him.*) Whether or not to remove other adjunct verbal modifiers is a challenging question. It may be useful for QG to drop other adjunct modifiers from very long sentences, but deciding which to drop and which to keep is left to future work.

3.2 Splitting Conjunctions

We also split conjunctions of clauses and verb phrases, as in Ex. 5 and Ex. 6 above. In most cases, the conjuncts in these conjunctions are entailed by the original sentence. For example, given *John studied on Monday but went to the park on Tuesday*, both *John studied on Monday* and *John went to the park on Tuesday* are entailed. Exceptions where conjuncts are not entailed include the following: conjunctions with *or* and *nor*, which we do not split; and conjunctions within downward monotone contexts such as negations or non-factive verbs (see [14, Chapter 6] for a discussion of this phenomenon), which we do split (we leave proper handling of this relatively rare case to future work).[6]

4 Extraction by Presupposition

In addition to the strict notion of semantic entailment, the pragmatic phenomenon of presupposition plays an important role in conveying information. The semantically entailed information in complex sentences often covers only a fraction of what readers understand. Consider again Ex. 7 about the Embargo Act. If our algorithm were to only extract semantic entailments as in §3, then it

[5] Appositives and relative clauses can be restrictive or non-restrictive. Restrictive versions resolve referential ambiguities and are not typically offset by commas (e.g., *the man who was tall...*). The non-restrictive ones provide additional information and are offset by commas (e.g., *John, who was tall,...*).

[6] We do not split conjunctions other than those conjoining clauses and verb phrases. In particular, we do not split noun phrases. Leaving conjoined noun phrases is probably advisable for factual QG applications, in particular because of difficult semantic and pragmatic issues involving nouns (e.g., as in *John and Susan were friends*).

would miss possibly useful questions about facts such as Ex. 9 because they are not semantically entailed by Ex. 7 (note how *the Embargo Act would hurt the American economy* is also not entailed).

On the other hand, if an algorithm were to extract from all nested constructions and naïvely copy features such as negation from the main clause, then it would output many false sentences, such as Ex. 10 (from the clause *which restricted trade with Europe*).

(9) *The Embargo Act restricted trade with Europe.*
(10) *The Embargo Act did not restrict trade with Europe.*

As the examples show, information in certain syntactic constructions is not semantically entailed but rather presupposed, or assumed to be true, by the reader [12, Chapter 4]. The meaning conveyed by these constructions is not affected by non-factive verbs like *believe* and *doubt*, negation, modal operators, and other constructions that affect the meaning of the main clause of the sentence.

The phenomenon of presupposition, often subsumed by the term "conventional implicature" [12], can be formally defined as follows: A **presupposes** B if and only if: (a) if A is true, then B is true, and (b) if A is false, then B is true.[7]

Many presuppositions have clear syntactic or lexical associations, or "triggers." These triggers facilitate the extraction of simple statements such as Ex. 2 and Ex. 4 above, which can lead to useful questions. A list of presupposition triggers is provided by [12].[8] The algorithm we present in §5 uses presupposition triggers to extract simplified sentences from the following constructions (in each example provided below, our method will extract *Jefferson was the third U.S. President*):

- non-restrictive appositives (e.g., *Jefferson, the third U.S. President,...*)
- non-restrictive relative clauses (e.g., *Jefferson, who was the third U.S. President,...*)
- participial modifiers (e.g., *Jefferson, being the third U.S. President,...*)
- temporal subordinate clauses (e.g., *Before Jefferson was the third U.S. President...*)

This set of presupposition triggers is a subset of the adjunct modifiers that our method removes when simplifying sentences (§3.1). They cover only subset of the adjunct modifiers because it was not clear how to create reliable and concise extraction rules for all of them. For example, extracting from certain parenthetical phrases, such as in *Jefferson (1743–1826)*, would likely require components to identify time expressions and to generate appropriately formatted output. We leave such extensions to future work.

[7] There are cases where A is neither true nor false, such as *The King of France is tall*. Such instances are often accounted for by abandoning the assumption of bivalence and allowing a third truth value: "neither-true-nor-false." [12, Section 4.2].

[8] The constructions discussed in §3 can be viewed as triggers for semantic entailment.

Algorithm 1 `extractSimplifiedSentences`(t)

$T_{result} \leftarrow \emptyset$
$T_{extracted} \leftarrow \{t\} \cup$ **extract** new sentence trees from t for the following: non-restrictive appositives; non-restrictive relative clauses; subordinate clauses with a subject and finite verb; and participial phrases that modify noun phrases, verb phrases, or clauses.
for all $t' \in T_{extracted}$ **do**
  $T_{result} \leftarrow T_{result} \cup$ `extractHelper`(t')
end for
return T_{result}

Algorithm 2 `extractHelper`(t)

$T_{result} \leftarrow \emptyset$
move any leading prepositional phrases and quotations in t to be the last children of the main verb phrase.
remove the following from t: noun modifiers offset by commas (non-restrictive appositives, non-restrictive relative clauses, parenthetical phrases, participial phrases), verb modifiers offset by commas (subordinate clauses, participial phrases, prepositional phrases), leading modifiers of the main clause (nodes that precede the subject).
if t has `S`, `SBAR`, or `VP` nodes conjoined with a conjunction $c \notin \{or, nor\}$ **then**
  $T_{conjuncts} \leftarrow$ **extract** new sentence trees for each conjunct in the leftmost, topmost set of conjoined `S`, `SBAR`, or `VP` nodes in t.
  for all $t_{conjunct} \in T_{conjuncts}$ **do**
    $T_{result} \leftarrow T_{result} \cup$ `extractHelper`($t_{conjunct}$)
  end for
else if t has a subject and finite main verb **then**
  $T_{result} \leftarrow T_{result} \cup \{t\}$
end if
return T_{result}

5 Extraction Algorithm

This section presents our algorithm for extracting simplified factual statements from complex input sentences. The algorithm operates on standard simplified Penn Treebank [15] phrase structure trees (i.e., without traces, etc.). The primary method `extractSimplifiedSentences`, shown in high-level pseudo-code in Algorithm 1, takes a tree t as input and returns a set of trees T_{result} as output. A helper function, shown in Algorithm 2, recursively splits conjunctions.

As an optional step (enabled for our experiments), the algorithm moves leading prepositional phrase and quote modifiers to the end of the main verb phrase. This transformation does not affect meaning, but it may lead to slightly more natural questions (e.g., *Who was elected President in 1796?* instead of *In 1796, who was elected President?*).

In our implementation, we use the Stanford Parser [10] to find sentence boundaries, tokenize, and parse. We use the `Tregex` tree searching language

[13] to identify the various constructions (e.g., non-restrictive appositives) that our algorithm operates on. After identifying the key nodes with `Tregex`, we manipulate trees using the Stanford Parser's API, which allows for inserting and deleting children, changing labels on tree nodes, etc.[9]

Due to space limitations, we do not include the extraction rules here; they will be listed in a future publication.[10]

6 Evaluation

To evaluate our implementation, we randomly sampled 25 articles from a set of Encyclopedia Britannica[11] articles about cities in the world (provided by [2]).

6.1 Baselines

We compare to two baselines. The first baseline is the Hedge Trimmer (hereafter, HT) algorithm [6], a rule-based text compression algorithm that iteratively performs various simplification operations until an input is shorter than a user-specified target length. We decided to use HT rather than a statistical sentence compression system because [6] found that HT produces more fluent output (that paper compared HT to a hidden Markov model for compression). We implemented HT using `Tregex` rules, and made two modifications to adapt HT for QG: We set the target length to 15, rather than 10. With the target length at 10 in preliminary experiments, HT often led to ungrammatical output and seemed to drop useful information for a QG application. Also, our implementation did not include the rules described in [6] for dropping determiners and temporal clauses (called "low-content units") because these rules are tailored to the headline generation application in [6].

The second baseline (MC-only), is to use our algorithm (§5) but only extract from the main clause (taking left most conjunct when conjunctions are present). This is an intermediate case between HT and our full system: like HT, it only produces only one output per input. Also, note that the outputs will be a subset of those from the full system.

6.2 Experimental Setup

We ran our system and HT on the 25 encyclopedia articles. All systems used the same parser [10]. We set the maximum sentence length for the parser to be

[9] For example, the following rule matches appositive constructions (see [13] for details on the rule syntax): `NP < (NP=noun !$-- NP $+ (/,/ $++ NP|PP=appositive !$CC|CONJP)) >> (ROOT << /^VB.*/=mainverb)`. For each match, the system creates a new sentence of the form, "`noun` *be* `appositive`." The form of *be* depends on the part of speech of the main verb, `mainverb`, and the number and person of `noun`.

[10] Our system, which is coded in Java and includes the extraction rules, is available at http://www.ark.cs.cmu.edu/mheilman/qg-2010-workshop.

[11] Available at http://www.britannica.com.

60 word or punctuation tokens. For the 5.1% of sentences that exceeded this maximum, all systems simply returned the input sentence as output.

We also conducted a small-scale manual evaluation of the quality of the output. We randomly sampled 150 sentences from the encyclopedia articles. For 50 of these, we extracted using HT; for the remaining 100, we extracted using our method (which subsumes MC-only).[12] When the full system generated multiple outputs from an input, we randomly sampled one of them.

We then asked two people not involved with this research to evaluate the 150 outputs for fluency and correctness, on 1–5 scales, following evaluation setups for paraphrase generation [4] and sentence compression [11]. We defined a correct output sentence as one where at least part of the meaning of the input sentence was preserved (i.e., whether the output was true, given the input). The outputs were mixed together, and the raters were blind to which system produced each of them. The Pearson correlation coefficients between the raters' judgments were $r = 0.92$ for fluency and $r = 0.82$ for correctness. For our analyses, we used the averages of the two raters' judgments.

For a rough estimate of how much of the information in the input sentences was included in the outputs, we computed the percentages of input word types (lemmatized with WordNet [16]) included in at least one output, for each system.

6.3 Results

The results of our experiments, presented in Table 1, show that the full system extracts more outputs per input than the two baselines (average 1.63 compared to 1.0). While the main clause baseline constitutes a majority of the output of the full system (58.1%), the other extraction rules contribute a substantial number of additional outputs. For example, 26.5% of outputs involved splitting conjunctions, and 13% were extracted from appositives. Also, the systems do not unnecessarily extract from already simple sentences: 27.9% of the inputs for the full system were unchanged (e.g., *Brussels lies in the central plateaus of Belgium.*).

The outputs from all systems were substantially shorter than the input sentences, which were 23.5 words long on average (not including punctuation). HT outputs were 12.4 words long on average, while outputs from the full system were 13.4 words long on average.[13]

The manual evaluation showed that our system extracts sentences that are, on average, more fluent and correct than those from HT. The mean fluency rating for the full system was 4.75 compared to 3.53 for HT, a difference which is statistically significant (t-test, $p < 0.001$). The mean correctness rating for the full system was 4.67 compared to 3.75 for HT ($p < 0.001$).

[12] We used 2 disjoint sets of sentences so the rater would not compare extractions for the same sentence. We included 100 for the full system rather than 50 in order to get accurate estimates for main clause baseline.

[13] The sentence length values assume the sentence splitting by the Stanford Parser works perfectly. In practice, we observed very few mistakes, mostly on abbreviations ending in periods.

	Input	HT	MC-only	Full
Number of sentences extracted	2952	2952	2952	4820
Sentences per input	1.00	1.00	1.00	1.63
Sentence length	23.5	12.4	15.4	13.4
Input word coverage (%)	100.0	61.4	69.1	87.9
Inputs unchanged (%)	100.0	33.8	27.9	27.9
Fluency	–	3.53	4.72	4.75
Correctness	–	3.75	4.71	4.67
Rated 5 for both Fluency and Correctness (%)	–	44.0	78.7	75.0

Table 1. Various statistics comparing the full system to the baselines and the unmodified input sentences. All statistics are averaged across input sentences.

The full system appears to provide better coverage of the information in the input. 87.9% of input words appeared in at least one output from the full system, compared to 61.4% for HT and 69.1% for the main clause baseline.

In an informal analysis of the errors made by the systems, we observed that the HT baseline is often overly aggressive when compressing sentences: it frequently drops key function words and informative modifiers. For example, from Ex 1, it produces the ungrammatical sentence *Prime Minister Vladimir V. Putin cut short a trip returning oversee the federal response*, and the grammatical but terse sentence *Mr. Putin built his reputation.* In contrast, the mistakes from the MC-only baseline and the full system were mostly the result of parser errors.

7 Related Work on Text Simplification

Numerous approaches to text compression and simplification have been proposed; see [6, 5] for reviews of various techniques. One particularly closely related method is discussed in [3]. That method extracts simple sentences from each verb in the syntactic dependency trees of complex sentences. However, the authors do not provide code or evaluate their system on realistic texts, so it is unclear how robust and effective their approach is (in a small evaluation with one text, 55% of the simplifications extracted by their system were acceptable).

8 Conclusion

In this paper, we presented an algorithm for extracting simplified declarative sentences from syntactially complex sentences. We motivated our extraction approach by identifying semantic and pragmatic issues that are relevant for QG. We evaluated our approach and showed that it is more suitable for extraction of well-formed entailments than a standard text compression baseline.

Our system can be integrated into existing QG frameworks. While we leave formal, extrinsic evaluations within a QG system to future work, we end by reporting that we have incorporated our approach with the QG system described by [8, 9]. The integrated QG system successfully simplifies sentences and transforms them into questions. For example, from Ex. 1 in the introduction, it produces concise questions such as the following:

(11) What did Prime Minister Vladimir V. Putin return to Moscow to oversee?
(12) Who cut short a trip to Siberia?
(13) Who was the country's paramount leader?
(14) Who built his reputation in part on his success at suppressing terrorism?

Acknowledgments. We acknowledge partial support from the Institute of Education Sciences, U.S. Department of Education, through Grant R305B040063 to Carnegie Mellon University; and from the National Science Foundation through a Graduate Research Fellowship for the first author and grant IIS-0915187 to the second author. We thank the anonymous reviewers for their comments.

References

1. Bar-Haim, R., Dagan, I., Greental, I., Szpektor, I., Friedman, M.: Semantic inference at the lexical-syntactic level for textual entailment recognition. In: Proc. of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing (2007)
2. Barzilay, R., Elhadad, N.: Sentence alignment for monolingual comparable corpora. In: Proc. of EMNLP (2003)
3. Beigman Klebanov, B., Knight, K., Marcu, D.: Text simplification for information seeking applications. On the Move to Meaningful Internet Systems (2004)
4. Callison-Burch, C.: Paraphrasing and Translation. Ph.D. thesis, University of Edinburgh (2007)
5. Clarke, J.: Global Inference for Sentence Compression: An Integer Linear Programming Approach. Ph.D. thesis, University of Edinburgh (2008)
6. Dorr, B., Zajic, D.: Hedge Trimmer: A parse-and-trim approach to headline generation. In: Proc. of Workshop on Automatic Summarization (2003)
7. Giampiccolo, D., Magnini, B., Dagan, I., Dolan, B. (eds.): The third pascal recognizing textual entailment challenge (2007)
8. Heilman, M., Smith, N.A.: Question generation via overgenerating transformations and ranking. Tech. Rep. CMU-LTI-09-013, Language Technologies Institute, Carnegie Mellon University (2009)
9. Heilman, M., Smith, N.A.: Good question! statistical ranking for question generation. In: Proc. of NAACL-HLT (2010)
10. Klein, D., Manning, C.D.: Fast exact inference with a factored model for natural language parsing. In: Advances in NIPS 15 (2003)
11. Knight, K., Marcu, D.: Statistics-based summarization - step one: Sentence compression. In: Proc. of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence (2000)
12. Levinson, S.C.: Pragmatics. Cambridge University Press, Cambridge, MA (1983)
13. Levy, R., Andrew, G.: Tregex and Tsurgeon: tools for querying and manipulating tree data structures. In: Proc. of LREC (2006)
14. MacCartney, B.: Natural language inference. Ph.D. thesis, Stanford University (2009)
15. Marcus, M.P., Santorini, B., Marcinkiewicz, M.A.: Building a large annotated corpus of English: The Penn Treebank. Computational Linguistics 19 (1993)
16. Miller, G.A., Beckwith, R., Fellbaum, C., Gross, D., Miller, K.J.: WordNet: An on-line lexical database. International Journal of Lexicography 3(4) (1990)
17. Rus, V., Graessar, A. (eds.): The Question Generation Shared Task and Evaluation Challenge (2009), available at http://www.questiongeneration.org/