Natural Language Engineering **20** (4): 441–468. © Cambridge University Press 2013
doi:10.1017/S1351324913000326

Parallel and nested decomposition for factoid questions

BRANIMIR BOGURAEV, SIDDHARTH PATWARDHAN, ADITYA KALYANPUR, JENNIFER CHU-CARROLL and ADAM LALLY

IBM Thomas J. Watson Research Center, 1101 Kitchawan Road, Yorktown Heights, NY 10598. USA
e-mail: {bran, siddharth, adityakal, jencc, alally} @us.ibm.com

(*Received 5 September 2012; revised 12 April 2013, 22 August 2013; first published online 29 October 2013*)

Abstract

Typically, automatic Question Answering (QA) approaches use the question in its entirety in the search for potential answers. We argue that decomposing complex factoid questions into separate facts about their answers is beneficial to QA, since an answer candidate with support coming from multiple independent facts is more likely to be the correct one. We broadly categorize decomposable questions as *parallel* or *nested*, and we present a novel question decomposition framework for enhancing the ability of single-shot QA systems to answer complex factoid questions. Essential to the framework are components for *decomposition recognition*, *question rewriting*, and *candidate answer synthesis and re-ranking*. We discuss the interplay among these, with particular emphasis on decomposition recognition, a process which, we argue, can be sufficiently informed by lexico-syntactic features alone. We validate our approach to decomposition by implementing the framework on top of IBM Watson™, a state-of-the-art QA system, and showing a statistically significant improvement over its accuracy.

1 Introduction

Question Answering (QA) systems for factoid questions typically adopt a 'single-shot' approach for the task. Single-shot QA implicitly assumes that the question contains a single nugget of information (as in Example (1)).

(1) *In which city are the headquarters of GE located?*

To answer the question, these approaches attempt to locate the factual information (the location of GE's headquarters) in their underlying resources. Largely a legacy of the nature of questions in the TREC QA track (Voorhees 2002), this tactic works in most cases where the assumption holds that a question is focused upon a single fact, and support for it may be found in a single resource.

Our work deals with more complex factoid questions, specifically ones containing multiple facts related to the correct answer. Because such facts may be independent of each other, they may well reside in different resources – and thus outside the scope of a single-shot search query.

https://doi.org/10.1017/S1351324913000326 Published online by Cambridge University Press

(2) *Which company has origins dating back to the 1870s and became the first US company to have 1 million stockholders?*

Example (2) shows a question with two facts about its answer (a company): its origins date back to the 1870s, and it became the first in the United States to have 1 million stockholders. We turn to *question decomposition* to identify and leverage the separate facts within the question, using them to garner support for the correct answer from independent sources of evidence. Our hypothesis is that the more independent facts support an answer candidate, the more likely it is to be the correct answer.

We focus here on decomposition applied to improving the quality of QA over a broad set of *factoid* questions. In contrast to most work on decomposition to date, which tends to appeal to discourse and/or semantic properties of the question (Section 2), we exploit the notion of *a fact* to view decomposition as circumscribed largely by the syntactic characteristics of questions. Facts are entity-relationship expressions, where the relation may be an N-ary predicate. Most informative – and thus most useful – facts are those that contain at least one named entity (including temporal or locative expressions).

The particular relationship between independent facts in any given question leads us to categorize 'decomposable' questions broadly into two types: *parallel* and *nested*. Examples (2) above and (3) below are parallel decomposable: sub-questions can be evaluated independent of one another. In contrast, nested questions require their decompositions to be processed in sequence, with the answer to an 'inner' sub-question plugged into the 'outer'. In Example (4), the inner sub-question is marked in brackets; its answer, 'cirrhosis', then leads to an outer question '*In the treatment of cirrhosis, which drug reduces portal venous blood inflow*', the answer to which is also the answer to the original question.

(3) *Which 2011 tax form do I fill if I need to do itemized deductions and I have an IRA rollover from 2010?*

(4) *In the treatment of [a condition that causes bleeding esophageal varices], which drug reduces portal venous blood inflow?*

Questions like these are found in domains such as medical, legal, financial, etc. Independent of domain and type, however, they share a common characteristic: If a search query is constructed from all the facts collectively describing the answer, very few (if any) relevant documents are likely to be found, with undesired consequences for the identification of potential answer-bearing passages. The notion of decomposition thus goes hand in hand with that of recursively applying a QA system to the individual facts (sub-questions), followed by suitable re-composition of the candidate answer lists for the sub-questions.

Some of our earlier decomposition works were motivated by such considerations: Kalyanpur *et al.* (2011) offer a brief overview of the decomposition framework, within which we then discuss particular heuristics for recognizing decomposable questions (Kalyanpur *et al.* 2012b).

This paper presents, in depth, our evolved decomposition approach. We describe the framework, and how it relates to a class of generic QA architectures. We follow

https://doi.org/10.1017/S1351324913000326 Published online by Cambridge University Press

with details of the new, and refined, strategies we have developed for recognizing and typing decomposable questions across the board. We then discuss the subsequent processing of sub-questions, together with their candidate answer lists, and illustrate how to keep this compliant with, and pluggable into, an existing QA system pipeline. Our evaluation demonstrates that informed re-ranking and composition of 'single-shot' responses can improve the performance on decomposable questions of IBM Watson,[1] an existing state-of-the-art QA system.

Watson has been developed to answer open-domain factoid questions, and demonstrated the breadth of questions it can handle by competing with, and defeating, human champions in the game of Jeopardy!,[2] a popular TV quiz show in the United States. The dataset of question/answer pairs we use[3] illustrates particular challenges for Watson, not least in the broad domain it covers and the complex language used.

Decomposable questions– complex, in the sense we introduced earlier – are prevalent in a set of particularly challenging Jeopardy! data: Final Jeopardy! (FJ) (we discuss these in some detail in the context of evaluating our approach in Section 7). These questions stand to benefit from a decomposition-based approach, and our work on decomposition described here demonstrably helps improve performance: we observe a statistically significant gain of 1.5% in Watson's accuracy on these questions.

2 Related works

A variety of approaches to question answering cite 'decomposition', in the context of addressing question complexity. Typically, they fall within the broad definition of the term: derivation of sub-questions from the original question, followed by appropriate use and/or combination of their answer sets, according to the relationship between the sub-questions and the original. In most work to date, however, 'complex' refers to questions requiring non-factoid answers: e.g. multiple sentences or summaries of answers (Lacatusu, Hickl and Harabagiu 2006), connected paragraphs as answers (Soricut and Brill 2004), explanations and/or justification of an answer (Katz, Borchardt and Felshin 2005), answers in the form of lists (Hartrumpf 2008) or lists of sets (Lin and Liu 2008), and so forth. Altogether, work in the decomposition space has been driven largely by the needs of answering 'beyond factoid' questions.

This work, on the other hand, addresses the more focused task of strictly defining 'decomposable' factoid questions, and improving the quality of an open-domain QA system by developing a strategy for decomposition-informed analysis of factoids.

In the literature, we find descriptions of processes like *local decomposition* and *meronymy decomposition* (Hartrumpf 2008), semantic decomposition using *knowledge templates* (Katz *et al.* 2005), *question refocusing* (Katz *et al.* 2005; Hartrumpf 2008), and *textual entailment* (Lacatusu *et al.* 2006) to connect, through semantics and

[1] IBM Watson is a trademark of International Business Machines Corp.
[2] Jeopardy! is a registered trademark of Jeopardy Productions, Inc.
[3] `http://www.jeopardy.com`.

https://doi.org/10.1017/S1351324913000326 Published online by Cambridge University Press

discourse, the original question with its numerous decompositions. In general, such processes appeal to domain semantics, and thus do not limit themselves to using only material explicitly present in the question. In contrast, for us this is a substantial, but important (and enabling) constraint we place upon our decomposition algorithms in order to retain the ability to do open-domain question answering.

Without implementation details, we have to assume that notions of decomposition such as cited above require access to fairly well-circumscribed domain models – not an assumption we can make for an open-domain QA application. Again, this is why we constrain our decomposition algorithms to use only material explicitly present in the question.

Closer to our strategy are notions like the syntactic decomposition of Katz *et al.* (2005), and the temporal/spatial analysis of Saquete *et al.* (2004) and Hartrumpf (2008). Still, our approach differs in at least two significant ways. We offer a principled solution to the problem of the final combination and ranking of candidate answers returned from multiple decompositions by means of training a model to weigh the effects of decomposition recognition rules. We also note that spatial and temporal decomposition are just special cases of solving nested decomposable questions.

Some of the decomposition works to date do also look at factoid questions. Katz *et al.* (2005) deploy syntactic decomposition for questions with more than one 'fact' (ternary expression) pertaining to the answer, with facts associated with different knowledge resources. Such distribution across resources motivates (syntactically driven) step-wise answer generation. Processing of, for instance, *When was the 20th President of the US born?*, will route (different) triples to *Biography.com* and the *Internet Public Library*. These resources will find, respectively, the identity of the 20th US President and his date of birth.

Such decomposition is crucially driven by knowledge of the resource(s) identified as appropriate for any given type of triples. At the same time, this approach makes no distinction between parallel and nested decomposition, nor does it offer guidance on how to interpret the results of multiple decomposition rules firing (resulting in alternative question-into-facts breakdowns). As we already mentioned, what we will offer here is the use of a machine learning (ML) framework to determine optimal (and ultimately correct) way to compose and re-rank candidate answers from different decompositions.

Additionally, Katz, Lin and Quan's (2002) decomposition strategy is dependent upon the mechanism for annotating knowledge resources with metadata concerning their scope and depth. One thing that is not clear is how this strategy scales up (if at all) in a system like ours, where reliance upon redundancy is built into the design, and support for an answer comes from multiple knowledge sources as a matter of basic design principle.

Another common trend in factoid question decomposition is, in particular, *temporal decomposition* (Saquete *et al.* 2004; Hartrumpf 2008), where the question is rephrased by plugging the value of a temporal expression in place of the expression itself. For us, this is just a special case of nested decomposition. In any case, such systems do not provide a general solution for decomposition.

https://doi.org/10.1017/S1351324913000326 Published online by Cambridge University Press

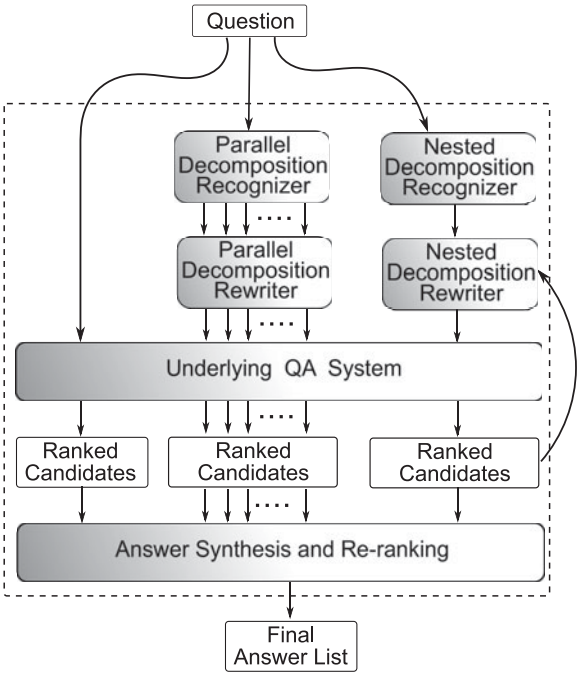


Fig. 1. Fact-based decomposition framework.

The closest similarity our fact-based decomposition has with an established approach is with the notion of asking additional questions in order to derive constraints on candidate answers (Prager, Chu-Carroll and Czuba 2004). However, the additional questions there are generated through knowledge of the domain, making that technique hard to apply in an open domain setting. In contrast, we developed a domain-independent approach to question decomposition, in which we use the question context alone in generating queriable constraints.

3 Fact-based decomposition

Enhancing a single-shot QA system with a capability for incremental solving of decomposable questions requires recognizing that a question is decomposable, and engaging in a staged processing of its sub-question parts. Whether parallel or nested, the system needs to identify the multiple facts, and configure itself as appropriate. Figure 1 shows our fact-based decomposition meta-framework (meta, as it builds on top of an existing QA system). It comprises four main components as illustrated in the figure.

Decomposition Recognizers analyze the input question and identify decomposable parts using a set of predominantly lexico-syntactic cues (Section 5). *Question Rewriters* rewrite the sub-questions found by the recognizer, retaining key contextual information (Section 6.1). *Underlying QA System* generates, for any factoid question, a ranked list of candidate answers, each with a *confidence* of the answer being correct. *Answer Synthesis and Re-ranking* is a placeholder for the particular process

https://doi.org/10.1017/S1351324913000326 Published online by Cambridge University Press

which tries to combine ranked candidate answers obtained to the original question with solutions for the decomposed facts into a uniform ranked answer list. In general, different combination functions may be appropriate for different types of decomposable questions. Thus, for the classes of parallel and nested questions, our decomposition strategies (described in Sections 6.2 and 6.3) defer to an Answer Merger (Gondek *et al.* 2012). A different class of questions (outside the scope of this paper) can be characterized as selecting from or aggregating over lists; e.g. *'How many countries belonged to the EU in 1994?'*, or *'Who was the first female track and field athlete to win medals in 5 different events at a single Olympics?'*. In cases like these, other synthesis operations may be required; cf. Hartrumpf's (2008) operational decomposition, or Lin and Liu's (2008) multi-focus questions. For yet more examples of different combination functions, see also special question-solving techniques of Prager, Brown and Chu-Carroll (2012).

We use Watson, our open-domain QA system (described in Section 4) as base. Our primary motivation was not just availability; as we shall see shortly, the open-endedness of its domain, and the challenging nature of its data, presented ample incentive to develop and experiment with factoid question decomposition strategies. Such strategies, however, are not limited by our choice. Any system can be plugged into our meta-framework as long as two conditions hold. The system should be able to solve factoid questions by providing answers with confidences which, broadly speaking, correlate with the likelihood that the candidate is a correct answer of the question. It should also maintain context/topic information for the question separately from the question's main content. In our data, explicit question categories carry that information (see Section 4.1).

Parallel and nested processing are distinct: Note the two different pathways in Figure 1. Multiple parallel facts are submitted to the base QA system; in contrast, inner–outer sub-question pairs are processed via a feedback loop. Furthermore, the base system is invoked on the full question for nested decomposable patterns, *and* on its decompositions.

The flow in Figure 1 is somewhat simplified for clarity. Recognition of decomposable questions (as we show in Section 5) relies on the input being analyzed in certain ways: syntactic structure, mapping to predicate–argument structure (PAS), identification of question focus. If an underlying QA system does not deploy such functionalities, the decomposition recognizers will have to incorporate them, as appropriate. If, as we will see in the next section, these (and more) NLP capabilities underlie the process of base QA system's Question Analysis, the decomposition recognizers (and rewriters) would in reality be at a point in the pipeline *after* question analysis. (Answer synthesis and re-ranking is truly positioned outside the base QA system.)

4 Question answering system overview

The core QA system underlying our decomposition framework is a state-of-the-art system designed to answer factoid questions. Figure 2 presents an overview of Watson's QA pipeline. It is a massively parallel probabilistic evidence-based

https://doi.org/10.1017/S1351324913000326 Published online by Cambridge University Press

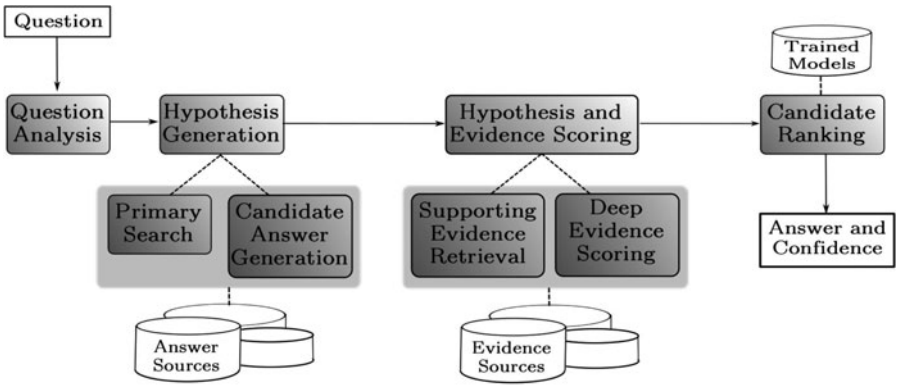


Fig. 2. Overview of the QA pipeline.

architecture for QA, comprising the following major stages (Ferrucci *et al.* 2010); the first and the last are particularly relevant for interfacing to the decomposition framework.

4.1 Question analysis

The first stage of the pipeline performs a detailed analysis to identify key elements of the question (such as focus, lexical answer type (LAT), question class, question category, etc.) used by later stages of the pipeline. The focus is the part of the question that refers to the answer, and typically encompasses the string representing the (lexical) answer type. (In the data we use, questions are posed in a declarative format, with stylized marking of question focus; this should not detract from referring to them as 'questions'.) An additionally distinguishing characteristic of Jeopardy! questions is that they are always accompanied by a category. This category is not, strictly speaking, indicative of a semantic type; it simply provides an additional dimension to the broad question content. An example question, with category of FICTIONAL ANIMALS, is shown below.

(5) FICTIONAL ANIMALS: *The name of this character, introduced in 1894, comes from the Hindi for 'bear'.*

In (5), the focus is *'this character'*, and the LAT is captured by the string *'character'*, which refers to the type of the answer sought. The notion of LAT complements that of semantic type of the answer, as it is crucial to Watson's ability not to be tied to any particular domain (for a detailed discussion, see Lally *et al.* (2012)). The system employs various lexico-syntactic rules for focus and LAT detection, and also uses a statistical ML model to refine the LAT(s). As we discuss later, focus and LAT are used during decomposition.

The question analysis stage also includes several general purpose natural language processing (NLP) components, such as parser, named entity detector, coreference resolver, etc., which are used by almost all of the components in the rest of the pipeline. These NLP capabilities are described in detail in Lally *et al.* (2012), together with an evaluation of their impact on end-to-end system performance. For this work,

https://doi.org/10.1017/S1351324913000326 Published online by Cambridge University Press

specifically, the decomposition recognizer relies on a PAS built over a syntactic parse of the input question, itself delivered by the English Slot Grammar (ESG) parser (McCord 1989; McCord, Murdock and Boguraev 2012). The latter citation also offers details concerning the parse-to-PAS process. In particular, it explains the properties of a tree-to-PAS transformation, causing the latter to be represented as a graph, with cycles and multiple incoming edges to certain predicate nodes.

4.2 Hypothesis generation

Following question analysis, the pipeline issues several queries derived from the question – largely based on syntactic and semantic relationships in it – to search the corpus of Watson's knowledge resources for relevant documents and passages, and then uses a variety of candidate answer generators to produce a list of hypotheses: candidate answers to the question. For details on effective query formulation against Watson's different knowledge sources, and on passage retrieval and candidate answer generation, see Chu-Carroll *et al.* (2012b).

The configuration of Watson applied to the Jeopardy! challenge used a broad variety of content sources, primary among which were Wikipedia and Wiktionary; the motivation for source selection is presented in Chu-Carroll *et al.* (2012c). Nothing about the decomposition task suggests that enhancing Watson for decomposition would require different, or additional, resources. In fact, during the experiments evaluating our approach to decomposition (Section 7), the content sources were held constant.

4.3 Hypothesis and evidence scoring

The system gathers supporting textual evidence from the corpus for each candidate answer and attempts to compute a match (or a degree of alignment) between the question text and the supporting passages found for a given candidate. For this, it uses a wide variety of answer scorers to evaluate the evidence. For Jeopardy!, Watson deployed over thirty answer scorers; detailed descriptions of some of them can be found in Kalyanpur *et al.* (2012a); Murdock *et al.* (2012a, 2012b); and Wang *et al.* (2012).

We make a special note here of the role of the Jeopardy! category provided with the question. While most of the answer scorers focus on the information provided in the question, some of the scorers also use the set of terms in the category as topically related words for the answer. The category words, therefore, play a role in answer scoring and contribute to the final confidences of the candidate answers.

We highlight the role of the Jeopardy! category terms because the question re-writer (described in Section 6) relies on categories to ensure that none of the contextual information is lost in the sub-questions.

4.4 Candidate ranking

Finally, the system uses ML models to weigh the analyzed evidence and rank the candidate answers. The models (Gondek *et al.* 2012) generate a confidence for each

https://doi.org/10.1017/S1351324913000326 Published online by Cambridge University Press

candidate being the correct answer to the given question, and the system answers with the top-ranked candidate. Watson can also choose to refrain from answering if it has a low confidence in all of its candidates. This ML framework – a logistic regression classifier – is used in the base system independent of decomposition processing, which, as we will see below, deploys its own ML for answer synthesis and re-ranking (cf. Figure 1).

5 Decomposition recognizers

The primary goal in decomposing questions is to identify facts involving the entity being asked for (in Section 4.1 we introduced the term 'focus'), which are simpler than the full question and solvable independently (Section 1). Most prior question decomposition work (Section 2) tends to defer to semantic, discourse, and other domain-specific information; in contrast, we recognize decomposable questions primarily on the basis of their syntactic characteristics. This is important for our claim that the decomposition framework outlined in Section 3 is generally applicable to multiple QA tasks and system configurations.

5.1 Decomposition patterns

There are numerous syntactic cues that are reliable indicators for decomposition, and it is predominantly such cues we exploit for driving the recognition and typing of decomposable questions. These include: ways in which clauses are embedded into subordinate positions, ways in which arguments cluster around an underlying predicate, distinctions between 'bare' and 'information-rich' noun phrases (essentially relating the syntactic structure of a noun phrase to constraints on its head noun), and so forth. For instance, the insight that a subordinate clause usually expresses an independent fact about an entity, within a larger question, leads to a conceptualization that patterns for detecting such subordinate constructions would identify questions as candidates for decomposition. In Section 5.1.1 we present a detailed example of such category of patterns.

As we discuss later in Section 7.2, we have no ground truth for question decompositions, and therefore have to rely on our intuitions – rationalized by data analysis of complex decomposable questions – to articulate cues like the ones listed above.

Even though pattern-defining notions like clause subordination and embedding, and pre- and post-modification of head nouns, are inherently syntactic, our decomposition framework is designed to operate over the PAS derived from the syntactic parse of the question. We use PAS graphs (see Section 4.1), instead of parse trees as output by the ESG parser, since they provide a simplified view over 'raw' syntactic trees, as well as normalize across ranges of different tree structures which express essentially the same meaning (McCord *et al.* 2012), offer detailed description of PAS, its relationship to syntactic output, and the process of parse-to-PAS transformation. An extra level of convenience is provided by the PAS retaining the dependency labels from the original syntactic representation (as illustrated in Figure 3: in essence, the PAS is not only stating that a predication `introduce[X,character,in(1894)]`

https://doi.org/10.1017/S1351324913000326 Published online by Cambridge University Press

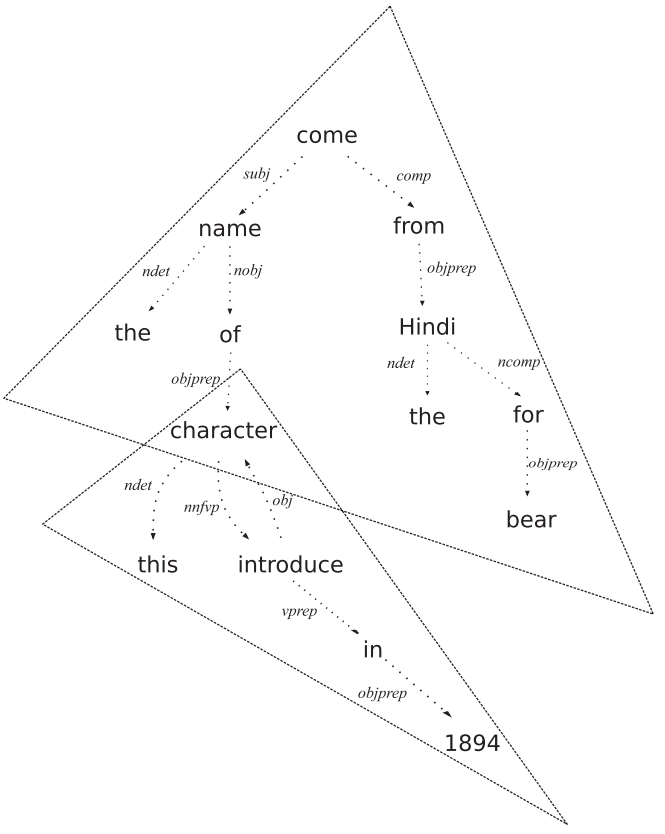


Fig. 3. PAS graph for 'The name of this character, first introduced in 1894, comes from the Hindi for "bear"': an example of independent subtrees.

holds, but also that its syntactic realization was that of a non-finite verb phrase modifying a noun, `nnfvp`, as indicated by the incoming edge label to the 'introduce' node).

We identify three major categories of syntactically informed decomposition patterns: *independent subtrees*, *composable units*, and *qualified segments*. Patterns are implemented as rules matching over (PAS) graphs. Pattern categories are realized as rule sets. The specific rules underlying an implementation depend on the language style, domain, and syntactic analysis framework deployed by a particular QA system configuration. Across decomposition frameworks targeting different base systems, rules may, or may not, have to be modified (consider, for instance, that different analytical frameworks will have different ways of representing non-finite verb clauses). We emphasize, however, that the three pattern categories above are *general*, in the sense that they capture relationships between syntactic characteristics of a question and its status with respect to decomposability (namely, subordinate clauses, of which non-finite verb phrases are one instance, are indicative of a decomposable question). Thus, differences in rule sets implementations do not affect our analysis of *syntactically cued* decomposition recognition.

Table 1 shows example decompositions within pattern categories; note that within a category, typically there are rule sets for parallel and nested decomposition types.

https://doi.org/10.1017/S1351324913000326 Published online by Cambridge University Press

Table 1. *Decomposition categories and examples*

Independent subtrees		
(1.P) Parallel		
clause	*Its original name meant 'bitter water' and it was made palatable to Europeans after the Spaniards added sugar*	**Fact #1**: Its original name meant 'bitter water' **Fact #2:** It was made palatable to Europeans after the Spaniards added sugar
complementary	*'American Prometheus' is a biography of this physicist who died in 1967*	**Fact #1**: This physicist who died in 1967 **Fact #2**: 'American Prometheus' is a biography of this physicist
(1.N) Nested		
coincidental	*When* '60 Minutes' *premiered, this man was US President*	**Inner Fact**: When *60 Minutes* premiered **Outer Fact**: When this man was president
based-on	*A controversial 1979 war film was based on a 1902 work by this author*	**Inner Fact**: A controversial 1979 war film **Outer Fact**: Film was based on a work by this author
named-for	*Article of clothing named for an old character who dressed in loose trousers in commedia dell'arte*	**Inner Fact**: An old character who dressed in loose trousers in commedia dell'arte **Outer Fact**: Article of clothing named for character
Composable units		
(2.P) Parallel		
verb-args	*He launched his lecturing career in 1866 with a talk later titled 'Our fellow savages of the Sandwich Islands'*	**Fact #1**: He launched his lecturing career in 1866
focus-mod	'The Mute' *was the working title of this 1940 novel by a female author*	**Fact #1**: This 1940 novel by a female author
triple	*His rise began when he upset Robert M. La Follette, Jr. in a 1946 Senate primary*	**Fact #1**: He upset Robert M. La Follette, Jr.
(2.N) Nested		
explicit-link	*To honor his work, this man's daughter took the name Maria Celeste when she became a nun in 1616*	**Inner Fact**: To honor his work, [this] daughter took the name Maria Celeste, when … **Outer Fact**: This man's daughter
descriptive-np	*The word for this congressional job comes from a fox-hunting term for someone who keeps the hunting dogs from straying*	**Inner Fact**: A fox-hunting term for someone who keeps the hunting dogs from straying **Outer Fact**: The word for this congressional job comes from term
Qualified segments		
(3.P) Parallel		
qualifier	*Winning in 1965 and 1966, he was the first man to win the Masters golf tournament in 2 consecutive years*	**Fact #1**: He was the first man to win the Masters golf tournament in 2 consecutive years

https://doi.org/10.1017/S1351324913000326 Published online by Cambridge University Press

Overall, we have the order of 120 rules for nested decomposition, and seventy for parallel. (The smaller number of rules for parallel decomposition is attributable to the fact that the syntactic composition of two independent facts is simpler than that for nested ones.)

More than one rule set may trigger over the same question; for clarity, we only show, in the third column, the result of activating the particular rule set, in the first column, over an example question, in the second column. When multiple decomposition hypotheses are at play, decisions about which rules do eventually apply to a question are deferred to the decomposition filters (Section 5.2) and the ML approach for weighing the contributions of rules (Section 6.2).

5.1.1 Independent subtrees

A good source of independent sub-questions within a question is in clauses likely to capture a unique piece of information about the answer, distinct from the rest of the question. Relative or subordinate clauses (not in a superlative or ordinal context; see Section 5.1.3, Qualified Segments) are examples of 'independent subtrees' and are indicative of parallel decomposition. Parse configurations that connect such subtrees to the focus are generally good indicators of a sub-question: we treat them as cues to 'break off' a subtree from the question as a decomposable fact.

Figure 3 shows a PAS graph where this type of rule set is triggered (see Example (5) in Section 4.1). Applied to the entire question, *'The name of this character, first introduced in 1894, comes from the Hindi for "bear"'*, the rules detect that the *'character'* node (which is also the focus of the question) performs a dual function structurally: it dominates a relative clause, as well as fills a (verb) argument slot in a higher clause. (This is indicated schematically by the two overlapping triangles.) The sub-graphs corresponding to the triangles can be viewed as partitions of the graph into two syntactically coherent segments, each describing different aspects of a shared node. The two partitions thus represent potential[4] sub-questions: *'this character, first introduced in 1894'*, and *'the name of this character comes from the Hindi for "bear"'*.

This category also includes rules using conjunctions as decomposition points (at various levels of the syntactic parse), as in Example (3) in Section 1 earlier: '*Which 2011 tax form do I fill if I need to do itemized deductions*' and '*Which 2011 tax form do I fill if I have an IRA rollover from 2010*'.

Parallel decomposition of this type is captured in two rule sets, *clause* and *complementary*, which differ primarily in that 'complementary' rules attempt to derive two separate sub-questions, while the 'clause' rules attempt to locate independent sub-questions in the original question PAS. Examples in Table 1 (Row (1.P)) illustrate this distinction.

As stated earlier, our rules are implemented as pattern-matching over the PAS graphs. For a matching engine we use Prolog unification; see Lally *et al.* (2012) for the motivation of this approach. A PAS graph is represented in the Prolog

[4] Not all decompositions turn into sub-questions; see Section 5.2, Decomposition Filters .

https://doi.org/10.1017/S1351324913000326 Published online by Cambridge University Press

space as a set of predications (and Prolog assertions) which explicitly associate predicate–argument tuples with the dependency labels capturing the corresponding parent–child subtree configurations. Observe, for instance (in the PAS representation of the above example; Figure 3), that the syntactic relationship between the *'come'* and *'name'* predicate nodes is that of `subj`, and that the following Prolog assertions hold:

```
isPredicateNode(054).
isPredicateNode(004).
lemmaForm(054,"come").
lemmaForm(004,"name").
subj(054,004).
```

Note that this representation augments the pure predication information (`come[name,...]`) with a record of what syntactic construction this predication is derived from (`subj`). The ESG parser indicates clausal post-modifiers to noun parse nodes by appropriately labeling the edge connecting the node to its subordinate subtree. Clausal edge labels include: `nrel` (e.g. *'the character that children love'*), `nrela` (*'the character children all over the world love'*), `nadjp` (*'the character famous for his bravery'*), `nnfvp` (*'the character conceived in 1894'*), and `nsubconj` (*'the character, as described in* "The Jungle Book"').

In such an environment, a rule implementing the pattern arising from our observation concerning relative or subordinate clauses being reliable indicators of decomposability would be stated as follows:

```
decomposableFact('parallel', FactAnchor) :-
   noun(FactAnchor),
   ( nrel(FactAnchor, _) ;
     nrela(FactAnchor, _) ;
     nadjp(FactAnchor, _) ;
     nnfvp(FactAnchor, _) ;
     nsubconj(FactAnchor, _) ),
   isFocus(FactAnchor).
```

In essence, the rule is looking for a noun node which participates in a predication derived from a subordinate clause (in this case, the dependency relation assigned by the parser is `nnfvp`); additionally, the rule will only fire when the noun node is the focus of the question (recall that the focus is the part of the question that refers to the answer: see Section 4.1). Having identified the *'character'* node as matching these particular constraints for a decomposable fact, a string rendering of the sub-question is derived by walking the terminal nodes of this node's sub-graph.

Another rule in the parallel rule set examines the graph structure above the question focus (we offer only a gloss, as details would require references to code snippets, which are immaterial to the essential points of this paper):

```
decomposableFact('parallel', FactClause) :-
   verb(FactClause),
%  collect arguments to FactClause (subj, obj, comp, vprep, vsubconj, ...)
%  which might be descriptive of a fact;
%  one of them should 'subsume' the focus node.
```

https://doi.org/10.1017/S1351324913000326 Published online by Cambridge University Press

With respect to the example in Figure 3, this rule would traverse the 'upper' sub-graph, including, but not descending below, the *'character'* node. Between them, the two rules derive two independent facts. Note that the two rules above are applied 'in parallel': thus, it may be the case that they both, or just one of them, will fire. A parallel decomposition, therefore, will not always return more than one independent sub-question.

Note that each rule 'knows' if it is performing a decomposition in a parallel, or nested, context. We simply apply all rules over the question; if the proposed decompositions make sense (e.g. they pass the filters, described in Section 5.2), they are submitted to the respective pathways (parallel or nested) of the decomposition framework (Figure 1).

For nested decomposition, we have three rule sets: *coincidental*, *based-on*, and *named-for*. These use lexical cues to detect specific semantic relations within the question that could indicate nestedness. For instance, the 'coincidental' rules identify sub-questions with finite subordinate clauses pre- or post-modifying the main part of the question, which indicate an inner–outer sub-question pattern: e.g. *'She was 6 years old and still known by the name Catalina when Columbus set sail for the New World'*; *'When* "60 Minutes" *premiered, this man was US president'*. The subordination of *'When* "60 Minutes" *premiered'*, away from the focus-enclosing top level of the question and itself underspecified, suggests that it can be viewed as a sub-question, solved first, followed by the outer sub-question *'Who was US President in 1968?'*.

The 'based-on' and 'named-for' rules detect sub-questions where the answer to the original question is based on or named for the answer to the inner sub-question (Table 1, row (1.N)). Note that in different domains, different relations may correlate with nestedness, for instance, *disease-causes-symptom* in a medical setting; cf. the inner sub-question, *'a condition that causes bleeding esophageal varices'*, from Example (4) in Section 1. The general pattern would still apply, even if we need different rule(s) to implement it.

Configurational information is also used to determine whether the question exhibits parallel or nested decomposition profile. Thus, the syntactic structure of Example (3) in Section 1 shows that two clauses characterize the same entity, *'2011 tax form*', which is also the focus: a clear indicator that the sub-questions are parallel. Similar sharing, even though not directly via a common head, is present in Figure 3.

Conversely, '*A controversial 1979 war film was based on a 1902 work by this author'* exhibits a very different set of syntactic characteristics. There are two underspecified entities (including one containing the focus) – *'a controversial 1979 war film'* and *'a 1902 work by this author'* – both characterized as head-plus-modifiers syntactic units. However, there is no 'sharing' of the separate characterizations (facts) via a common element (as *'this character'* does in Figure 3). This indicates nestedness: the inner sub-question is the one around the underspecified, but non-focus, element (*'a controversial 1979 war film'*); the outer is *'[film] was based on a 1902 work by this author'* (Table 1, row (1.N)).

https://doi.org/10.1017/S1351324913000326 Published online by Cambridge University Press

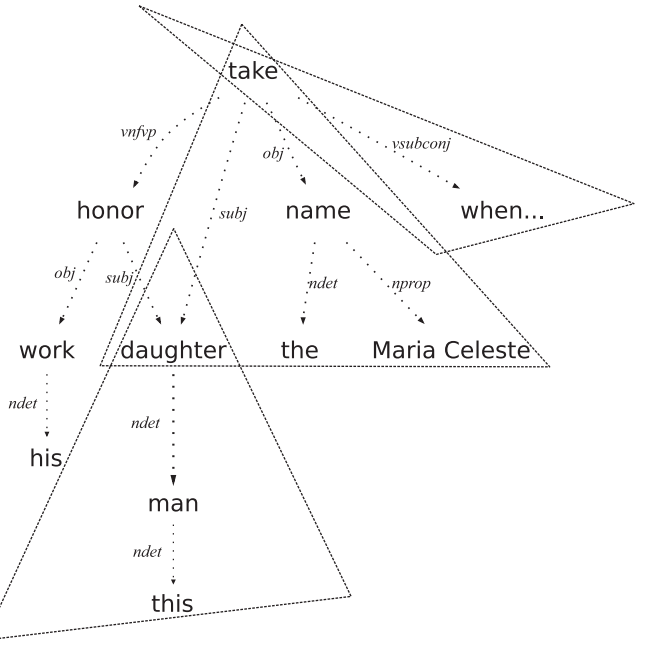


Fig. 4. Example of composable units.

5.1.2 Composable units

An alternate strategy for identifying sub-questions is to 'compose' a fact by combining elements from the question. In contrast to the previous category, the 'composable units' rules combine separate parts of the PAS into a fact. For instance, a sub-question can be created by associating the focus head with its premodifiers and postmodifiers. If the premodifiers and postmodifiers are sufficiently specific, we obtain reasonably independent sub-questions, with parallel-decomposable behavior.

Three parallel decomposition rule sets are defined in this category: *verb-args*, *focus-mod*, and *triple* (see Table 1, row (2.P)). The rules in *verb-args* 'compose' a fact from the verb and its arguments (subject, object, PP complements). The *focus-mod* rules combine the head of the focus NP with its modifiers to generate a sub-question. *Triple* rules are similar to *verb-arg* rules in that they also generate facts by bridging across arguments to an underlying predicate, but unlike the *verb-arg* rules, this composition always links only two of the arguments to the underlying predicate, creating e.g. subject–verb–object or subject–verb–complement triples. The resulting facts state fewer constraints compared to full question predications, and are thus more likely to retrieve an answer-bearing passage.

Here also, a particular configuration around the focus may indicate a question requiring nested processing. For nested, the 'composable units' category has two rule sets: *explicit-link* and *descriptive-np* (Table 1, row (2.N)).

In contrast to questions where modifiers of the focus can be cues for parallel decomposition (i.e. the *focus-mod* rules above), the 'explicit-link' rules detect nested decomposition, signaled by the focus itself being a modifier. Figure 4 is an example for this type of rule set, applied to the PAS graph of the question: *'To honor his work,*

https://doi.org/10.1017/S1351324913000326 Published online by Cambridge University Press

this man's daughter took the name Maria Celeste when she became a nun in 1616'. Here the focus ('*this man*') is a possessive determiner to an underspecified node ('*daughter*'). Intuitively, traversing the graph without descending to the level of the focus would 'carve out' an inner sub-question itself focused on that underspecified node ('*daughter*'): *'[this] daughter took the name Maria Celeste when ...'* (also see Table 1, row (2.N)).

Operationally, the two upper triangles in Figure 4 denote the sub-graphs which describe aspects of the inner sub-question: the basic predication covering the '*daughter*' node and a *vsubconj* ('*when*') post-modification to it. The sub-question itself is composed as concatenation of the corresponding question fragments. The bottom triangle in the figure demarcates the skeleton of the outer sub-question. Answering (the inner) with *'Virginia Gamba'* allows the framework to plug it into the outer sub-question.

Two inter-related rules play part in this decomposition. The first detects a syntactic configuration indicative of nestedness (focus node in modifier position to a higher node):

```
decomposableFact('nested-outer', FactAnchor) :-
   noun(FactAnchor),
   ( ( ndet(FactAnchor,FactAnchorMod) ); % "this man's daughter"
     ( nobj(FactAnchor,PP),
       objprep(PP,FactAnchorMod) );      % "the daughter of this man"
     ( nrel(FactAnchor,SubClause),
       arg(SubClause,FactAnchorMod) );   % "a war that ended in this year"
     ........
   ),
   isFocus(FactAnchorMod).
```

The second is very similar to the `decomposableFact` rule introduced earlier, for parallelly decomposable questions, but here we require additionally that `FactAnchor` participate in both inner and outer sub-questions:

```
decomposableFact('nested-inner', FactClause, FactAnchor) :-
   verb(FactClause),
%  collect arguments to FactClause (subj, obj, comp, vprep, vsubconj, ...)
%  which might be descriptive of a fact;
%  one of them should be a node (FactAnchor) 'subsuming' the focus node
   decomposableFact('nested-outer', FactAnchor).
```

The 'descriptive-np' rule set finds 'parenthetical' descriptions of underspecified nouns in the primary question, as in e.g. *'This arboreally named area was made famous by [a prince in the region noted for impaling enemies on stakes]'*: the nested-decomposable nature of this question is captured in the descriptive phrase (in square brackets) functioning as an inner sub-question.

5.1.3 Qualified segments

This category of rules covers situations where the modifier of the focus is a relative qualifier, for instance '*the first*', '*only*', '*the westernmost*', and so forth. In such cases, information from another clause is usually required to complete the relative qualifier:

https://doi.org/10.1017/S1351324913000326 Published online by Cambridge University Press

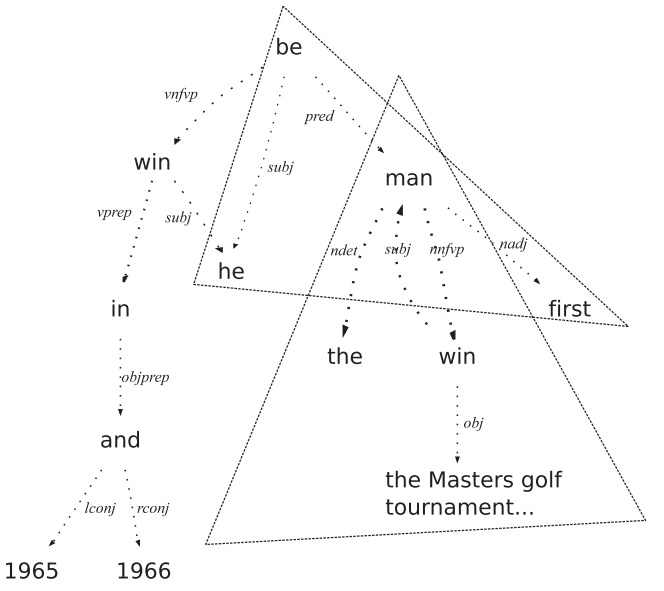


Fig. 5. Example of qualified segments.

consider e.g. the incomplete (underspecified) '*the third man*' versus the fact '*the third man … to climb Mt. Everest*'). For configurations like these, rules in this category combine the characteristics of 'composable units' rules with those of 'independent subtrees'. An example decomposition for *'Winning in 1965 and 1966, he was the first man to win the Masters golf tournament in two consecutive years'* is shown in Figure 5. Having detected a suitably qualified focus node, we find the larger sub-graph above it representing the basic predication over the focus (the top triangle in Figure 5), and the sub-graph below it, for the supporting relative clause elaborating on the qualification (the bottom triangle). We 'compose' these to derive a sub-question. As illustrated in Table 1 (row (3.P)) , for parallel decomposition our rule set covers sub-questions expressed as superlatives. We do not have any rules of this type for the nested case.

5.2 Decomposition filters

All three pattern categories above rely only on a syntactic analysis of the question. When rules fire, they also identify question segments proposed as sub-questions.

Patterns tend to over-generate. Left unconstrained, they suggest decomposition of questions that are, arguably, not decomposable to begin with. For example, the *triple* rule set for parallel decomposition incorrectly decomposed Example (6), to derive *'Charles Dickens called it'* as a sub-question. Similarly, the *descriptive-np* rule set for nested decomposition incorrectly decomposed Example (7) generating the inner sub-question: *'This tiered steeple of St. Bride's Church in London'*.

(6) Holiday Stuff: *In an 1850 essay Charles Dickens called it 'that pretty German toy'*
(7) Architecture & Society: *The tiered steeple of St. Bride's Church in London inspired the traditional form of this festive item*

https://doi.org/10.1017/S1351324913000326 Published online by Cambridge University Press

Table 2. *Distribution of rule types*

	Parallel	Nested
All rules	238	138
Ind. subtrees	140	39
Comp. units	167	113
Seg. w/qualifiers	17	9
Total size of data set	517	

Also, by themselves, the rules cannot guarantee that the sub-questions are specific enough to point to a single answer. For example, the facts *'this controversial 1979 war film'* and *'this quaint old brownstone building'* are syntactically identical. From a semantic point of view, however, the first fact is more likely to point to a unique answer as opposed to the second one, underscoring the fact that it is unrealistic to rely solely on syntax for decomposing questions.

To mitigate against the rules over-generating, we apply the following heuristic filters to the proposed sub-questions. The set of filters was developed and tuned on the basis of error analysis.

We look for decomposable facts pertaining only to the question focus. We discard sub-questions that do not contain either a named entity, a quoted string, or a time or date expression (these are detected by the ESG parser): cf. *'this quaint old brownstone building'*. Additionally, we discard sub-questions that almost completely overlap the entire question or a sub-question from a prior rule. A partial priority order is imposed on rule application, based on observations of how informative the facts generated by a rule are; this order is reflected on a per-type basis in Table 1: e.g. within type row (2.P) we prefer *verb-args* to *triple* since the latter tends to produce less constrained facts than the former: cf. *'Charles Dickens called it'*.

Some statistics concerning how often rules are activated, in an end-to-end system configuration, are given in Section 7.3. Below we offer an estimate of the relative sizes of parallel- and nested-decomposable questions, detected on a per-rule category basis.

5.3 Analysis

In Section 7.2 we discuss the difficulties of creating a reliable gold standard for decomposable questions, made even more challenging by the fact that there may well be multiple ways to decompose a question, and that the utility of different decompositions (and therefore the choice of an optimal one) tends to be tied to the overall content of the background corpus.

In this paper, we are primarily interested in the impact of the decompositions on end-to-end QA performance (see Section 7.3). Still, even without component-level evaluation, we can get an estimate of the coverage of the decomposition recognizers. Table 2 presents a distribution of the decomposition pattern categories after applying the recognizers to a batch of 517 representative Jeopardy! questions used as our

https://doi.org/10.1017/S1351324913000326 Published online by Cambridge University Press

development set (Section 7.1).[5] Approximately 50% of the questions are identified as parallel decomposable, and about 25% are identified as nested. The recall of the recognizers is not clear from this, but Table 2 does give an idea of the proportion of questions that go through the question rewriting and answer re-ranking process later in the decomposition framework.

6 Using decomposition

In essence, decomposition recognition informs two processes. According to the question type, parallel or nested, the appropriate pathway in the framework (Figure 1) needs to get instantiated. (Recall that by this stage in the processing, sub-questions have been typed, and provenance information from the decomposition recognizers is available, to determine dispatching of a sub-question.) Before sub-questions are submitted to the base QA system, they may need augmentation to facilitate the recursive system invocation. Then the answer sets obtained from sub-questions processing need to be analyzed and rationalized to determine the final answer to the original question.

6.1 Question rewriting

For parallel decomposition, the goal is to solve the original question Q by solving sub-questions independently and combining results appropriately. For example, consider the Jeopardy! question[6]

(8) HISTORIC PEOPLE: *The life story of this man who died in 1801 was chronicled in an A&E Biography DVD titled* 'Triumph and treason'

We get two decompositions:

Q_1: *This man who died in 1801*
Q_2: *The life story of this man was chronicled in an A&E Biography DVD titled* 'Triumph and treason'

Submitting sub-questions– unmodified – to the base QA system raises at least two problems. Sub-questions are often much shorter than the original question, and in many cases no longer have a unique answer. Moreover, some of the information from the original question that was dropped in a sub-question may be relevant contextual cues that the QA system needs to come up with the correct answer. Q_1 above illustrates these problems: it does not have a unique answer, and suffers from a recall problem (the correct answer is not in the candidate answers list of the base system when it considers this sub-question alone).

Our solution is to insert contextual information into sub-questions. In a two-step process for a sub-question Q_1, we obtain the set of all named entities and nouns

[5] Note that multiple rules can apply to a question.
[6] Recall that Jeopardy! questions are typically phrased in declarative format; also, recall that questions are associated with category information, which further contextualizes the search for the answer.

https://doi.org/10.1017/S1351324913000326 Published online by Cambridge University Press

(ignoring stopwords) in the original question text outside Q_1, and we insert these keywords into the original question category. In Jeopardy! questions, the category field is the context/topic information which the underlying QA system needs in order to use the decomposition framework, as stated in Section 3. In general, a QA system may derive such information in a variety of ways, e.g. by exploiting the problem description in a technical assistance QA setting, or a patient's medical history in a medical QA setting. What is important here is that the base system treats such information differently from the question itself. Rewriting takes advantage of this differential weighting to ensure that the larger context of the original question is taken into account when evaluating a sub-question, albeit with less weight.

The rewritten Q_1/Q_2 for Example (8) are:

(8-1) HISTORIC PEOPLE (A&E BIOGRAPHY DVD 'TRIUMPH AND TREASON'): *This man who died in 1801*

(8-2) HISTORIC PEOPLE (1801): *The life story of this man was chronicled in an A&E Biography DVD titled* 'Triumph and treason'

The keywords are inserted in parentheses, to ensure a clear separation between the original category terms and the context terms added. Other systems may need a different rewriting tactic.

The above rewriting technique is used for both parallel and nested decomposable questions. For the nested case, there is an additional rewriting step that needs to be done – after solving the inner question, we need to substitute its answer into the outer when solving for it. Thus, the first example in Table 1, row(1.N) would have its inner focus *'When* "60 Minutes" *premiered"* replaced with *'In 1968'* creating the outer question *'In 1968, this man was US President'* whose solution is the answer to the original question.

6.2 Candidate answer re-ranking: parallel

The base QA system will process the rewritten category/sub-question pairs, and will produce a set of ranked candidate lists with confidences (cf. Section 4.4). These need to be combined into a final answer list for the original question, accounting for information across all sub-question candidate lists.

If the score for a candidate answer generated by the base QA systems corresponded to a probability that this answer was correct, and if the sub-questions derived by decomposition were independent, a final score for each candidate answer to the original question could be calculated as a product of the scores for all of its sub-questions. These assumptions, however, are not valid. The confidence of a candidate answer is not a probability – in the strict mathematical sense – that it is correct (cf. Sections 3 and 4.4). Also, even if sub-questions were independent of each other, question rewriting breaks this assumption as it brings information from the remainder of the question into the sub-question context.

Furthermore, the sub-questions are generated by decomposition rules that have varying precision and recall, and thus should not be weighted equally. If the rules' quality could be evaluated precisely, by itself, an approach could be designed to weigh them appropriately when generating the final confidence. But, as we discuss in

https://doi.org/10.1017/S1351324913000326 Published online by Cambridge University Press

Table 3. *Features in parallel re-ranking model*

Feature name	Description
Orig. top answer	Binary feature signaling whether candidate was top answer to non-decomposed question.
Orig. confidence	Confidence for candidate answer to non-decomposed question.
# Facts matched	Number of sub-questions which have candidate answer in top 10.
Rule – verb-args Rule – clause Rule – qualifier Rule – focus-mod Rule – complementary Rule – triple	Features corresponding to the rule sets used in parallel decomposition – each feature takes a numeric value, which is the confidence of the QA system on a fact identified by the corresponding rule set.

the evaluation section (7.2), this is impractical, given the difficult nature of creating a question decomposition gold standard. We also have to consider the possibility of bad (and wrong) sub-questions – e.g. due to a bad parse – when a fall-back to the original question would necessitate that the confidence for the candidate answer for the entire question be also considered in making a final decision.

For all these reasons we use a ML approach to combine information across sub-question answer confidences, with features designed to capture the above information. The features used in the model are shown in Table 3; for features associated with specific rules, the value of a feature is the answer confidence for a decomposed sub-question found by the rule. In case a candidate answer is not in the answer list of the full question or any of the decomposed sub-questions, the corresponding feature value is set to *missing*. If a rule generates multiple sub-questions, its corresponding feature value for the candidate answer is set to the sum of the confidences obtained for that answer across all sub-questions. The model is trained using Weka's (Witten and Frank 2000) logistic regression algorithm with instance weighting.

6.3 Candidate answer re-ranking: nested

Nested questions decompose into inner/outer question pairs. The task is to solve the inner question first, substitute the answer obtained, based on its confidence, into the outer, and solve that for the final answer. This is contingent upon selecting answers to the inner question which might profitably be plugged into the outer; substituting incorrect answers will only lead to noisy final answers, with negative impact to overall accuracy.

We rely on the ability of the underlying QA system to produce meaningful confidences for its answers, and only consider the top answer to the inner question for substitution into the outer – if its confidence exceeds some threshold.

Finally, the answers to the outer question need to be related to the full question answer list to produce the final ranked answers. For answer re-ranking, we use

https://doi.org/10.1017/S1351324913000326 Published online by Cambridge University Press

the following heuristic selection strategy. For the answer obtained through decomposition and rewriting, we compute aggregate confidence as the product of the inner-question answer confidence and the outer-question answer confidence. We then compare this value with that of the confidence for the top answer to the entire question, as computed by the base QA system (see the left-most pathway in Figure 1). The answer with the higher confidence is then selected as our final answer. Note that this re-ranking strategy is different from the one used in parallel decomposition where we combine results from multiple sub-questions into a single confidence.

7 Evaluation

In this section, we present an empirical evaluation of our decomposition framework. We demonstrate that the use of question decomposition on a QA task results in statistically significant improvements over a baseline without decomposition. Section 7.1 summarizes details of the data used in our evaluation, Section 7.2 covers the experimental setup and results, and Section 7.3 presents a brief discussion of interesting observations and insights gleaned from the work.

7.1 Evaluation data

As we already mentioned (in Section 1), the Jeopardy! data contains a set of questions which are particularly challenging. Final Jeopardy! questions are often long and complex, with multiple facts or constraints that need to be satisfied. Frequently, the nature of these makes it unrealistic to expect that all of them will be articulated within a single, brief passage in our content sources. This is not a statement specific just to the corpora we prepared for Jeopardy! (Chu-Carroll *et al.* 2012c). Finding the answer to such questions typically requires additional processing, implementing some kind of a one-or-more hops across intermediate stages of answer determination (Chu-Carroll *et al.* 2012a). Question decomposition, in the sense we discuss in this paper, is one such strategy, and FJ questions can significantly benefit from our decomposition approach. Example (5) in Section 4.1 earlier illustrates the decomposition-related characteristics of an FJ question.

FJ questions are also typically much harder to answer than regular Jeopardy! questions, both for humans (based on statistics mined from the website; see Section 7.3) and for our base QA system (based on its performance over FJ questions as compared with regular Jeopardy! questions). The test set comprises close to 3,000 FJ questions, broken into 1,138 for training, 517 for development, and 1,269 questions for testing (as blind data). Since we do not have a gold standard for decomposable questions (see Sections 5.1 and 7.2), this split was made randomly.

7.2 Experiments

The decomposition rules (Section 5) and re-ranking parameters (Sections 6.2 and 6.3) were defined and tuned on the development set. The final re-ranking model was

https://doi.org/10.1017/S1351324913000326 Published online by Cambridge University Press

Table 4. *End-to-end QA evaluation (1,269 test questions)*

System	Accuracy
Baseline	50.05% (635/1,269)
Parallel (no QR)	49.96% (634/1,269)
Parallel (with QR)	50.66% (643/1,269)
Nested (with QR)	50.43% (640/1,269)
Parallel + nested (with QR)	51.06% (648/1,269)

trained over the training set, using the features described in Section 6.2, and logistic regression with instance weighting: the negative instance weight was set 0.25 times lower, based on analysis over the development set, and reflecting the fact that the underlying QA system generates many more negative answers than positive.

The results of applying the decomposition rules followed by the re-ranking model to 1,269 test questions are shown in Table 4. The row labeled 'Baseline' is the performance of the underlying QA system used in our meta-framework without any decomposition components, applied to the same test set.

We evaluated separately the impact on parallel decomposition of our question rewriting strategy (cf. Section 6.1), which maintains contextual information from the original question into the sub-questions ('QR' in Table 4 stands for Question Rewriting). For this purpose, we altered our algorithm to issue the sub-question text as-is, using the original category (without rewriting), retrained the model, and evaluated the resulting model again on the test set. The results are shown in the row labeled 'Parallel (no QR)'. The results for parallel, and separately nested, decomposition with question rewriting are in the rows labeled 'Parallel (with QR)' and 'Nested (with QR)', respectively. Finally, the row labeled 'Parallel + Nested (with QR)' contains the end-to-end performance of the QA system with both parallel and nested decomposition (with rewriting) enabled.

Table 5 presents an end-to-end QA performance on just the 'decomposable' subset of test questions: namely, the set of questions recognized as decomposable by our decomposition recognizers. For parallel decomposition, the system judged 598 of the 1,269 test questions to be decomposable. On the other hand, only 255 questions triggered nested decomposition. Like Table 4, this table too presents QA performance for the system without decomposition (in rows labeled 'Baseline (no decomposition)'), performance of parallel decomposition without and with question rewriting (in rows labeled 'Parallel (no QR)' and 'Parallel (with QR)'), and performance of nested decomposition with question rewriting (in the row labeled 'Nested (with QR)').

We do not offer separate analysis of decomposition recognition. As we mentioned in Section 5.1, there is no gold standard for question decomposition. Manual creation of such a standard is highly nontrivial, largely due to the numerous alternative ways to decompose a question, and to synthesize unique facts from the segments (a particular issue here is that the effectiveness of a given decomposition is affected by the content in the entire underlying corpus). Indeed, this is precisely the motivation for weighting the decomposition rules in a trained re-ranking model (Section 6.2).

https://doi.org/10.1017/S1351324913000326 Published online by Cambridge University Press

Table 5. *QA accuracy on decomposable subsets*

Configuration	Accuracy
Parallel decomposable subset (598 questions)	
Baseline (no decomposition)	56.68% (339/598)
Parallel (no QR)	56.52% (338/598)
Parallel (with QR)	58.02% (347/598)
Nested decomposable subset (255 questions)	
Baseline (no decomposition)	50.58% (129/255)
Nested (with QR)	52.54% (134/255)

Given this, we are interested *only in measuring the impact of decomposition on end-to-end QA performance.*

7.3 Results and discussion

One observation from Tables 4 and 5 is that according to our rules – which, when constrained by the decomposition filters, are reasonably precise – significant percentage of FJ (blind test) questions have decomposition potential. Given that our data analysis separately established that decomposition would help in solving FJ questions (Section 7.1), this offers some validation of the decomposition recognition rules – even though without ground truth for decomposable questions, we cannot perform accuracy analysis.

The parallel decomposition rules were able to decompose a large fraction of the test set (598 out of 1,269 questions: 47%). Interestingly, the performance of the baseline QA system on the decomposable set was 56.6%, which is 6% higher than the overall performance over the entire test set. One reason for this result would be that parallel decomposable questions typically contain more information (more than one fact or constraint that the answer must satisfy) about the same answer, and the system in some cases is able to exploit this redundancy – such as when one fact is strongly associated with the correct answer and there is evidence supporting this in the sources.

A different, important result is that using the decomposition algorithm *without* question rewriting did not show impact over the baseline. This highlights the importance of contextual information for QA. On the other hand, when using rewriting to maintain context (Section 6.1), our parallel decomposition algorithm was able to achieve a gain of 1.4% on the parallel decomposable question set, which translated to an end-to-end gain of 0.6%.

Separately, Table 5 shows that roughly a fifth (255 out of 1,269 questions) of the entire test set were recognized as nested decomposable. The performance of the baseline QA system on the nested decomposable set was roughly the same as the overall performance (and much lower than the parallel decomposable cases). The likely explanation here is that nested questions require solving for an inner fact first, and it is the answer to this which often provides the necessary missing information required to find the correct answer: this staged dependency makes

https://doi.org/10.1017/S1351324913000326 Published online by Cambridge University Press

nested questions much harder to solve than parallel decomposable ones with their multiple independent facts. Our nested decomposition algorithm using the heuristic re-ranking approach (Section 6.3) was able to achieve a gain of 2% on the nested decomposable question set, which translated to an end-to-end gain of 0.4%.

The aggregate impact of parallel *and* nested decomposition was a 1.5% gain in accuracy on the decomposable set, and a 1% gain on end-to-end system accuracy (in our case the questions that are classified as parallel or nested form disjoint sets).

To put these results in perspective, we emphasize that the baseline QA system represents state of the art in solving Jeopardy! questions. The FJ questions, which exclusively comprise our test data, are known to be harder than regular Jeopardy!: qualified Jeopardy! players' accuracy on this kind of questions is 48%,[7] and the underlying QA system has an accuracy close to 51% on previously unseen FJ questions. A gain of 1% end-to-end on such questions, therefore, represents a strong improvement. Also, using the statistical McNemar's test (McNemar 1947), we found the net end-to-end impact to be statistically significant at a 99% confidence interval.

As we showed earlier in the paper, decomposition relies crucially on correctly instantiated PAS graphs corresponding to syntactic analysis of the questions. Given that the PAS builder is, in effect, a mapper for syntactic dependency trees, correct syntactic parses would ensure correct PAS structures. Thus, of all the NLP components in the question analysis pipeline (Section 4), the ones that make our approach to decomposition both viable and effective are the ESG parser and the focus and LAT detector: independent accuracy analysis of the parser shows it to perform in the 90 percentile for accuracy (92% over Jeopardy! data, and 88.7% over Wikipedia content) (McCord *et al.* 2012); similarly, the focus/LAT detection (itself also dependent upon the parser) performs at F1 =0.796, with a very high per-question recall, 0.905.

We note that our error analysis of the test questions shows a wide variety of reasons for their failures beyond question decomposition. To further improve the system on this test set would require advances beyond deciding whether to take a single-shot or decomposable approach to questions, which is beyond the scope of this paper.

Broadly speaking, decomposable questions are questions which 'over-specify' the answer. Search queries derived from such questions are likely to result in recall failures – because the juxtaposition of multiple facts about the answer may not be encountered in a QA system's content sources. The decomposition-based approach to answering such questions relies crucially on the assumption that searching for separate, smaller facts which are independently true of the answer would be more productive than searching for co-located mentions of *all* the facts (Section 1).

It follows that processing decomposable questions via a conventional QA pipeline may lead to not enough support for the correct answer. On the other hand, processing sub-questions derived as fact-based decompositions will find more relevant passages for some, or all, of its decompositions, as relevance is now redefined with respect to sub-question-bound facts about the answer.

[7] Calculated over historical games data, from J-archive (`http://www.j-archive.com`).

https://doi.org/10.1017/S1351324913000326 Published online by Cambridge University Press

Our analysis of system logs corroborates this: We observe apposite supporting passages retrieved by the decomposition-enabled system for sub-questions, while the conventional system produced no recall for some of the facts in full questions. We are unable, however, to quantify this observation: not only due to the lack of gold standard for decompositions but also because we do not have an answer key for such sub-questions. Additionally, we do not have the scaffolding for side-by-side comparison of different system architectures to expose the differences between multiple interactions between passage scoring and candidate answer ranking introduced by different regimes of passage retrieval.

Nonetheless, our work substantiates the notion that there are intrinsic merits to fact-based decomposition when answering questions which may – for whatever reason (e.g. intended complexity, as in FJ, or the realities of complex domains such as financial consulting or medical diagnosis) – exhibit the characteristics of conflating multiple independent facts together in a single question. Detecting decomposable questions, deriving appropriate sub-questions, and composing the individually obtained answers correctly is a very hard task (as evidenced earlier in this section), in the light of which a performance boost as the one we report here is not just a significant improvement of a particular system but also indicative of the value of paying due attention to question details.

8 Conclusion

In this paper, we presented a general-purpose decomposition framework for answering complex factoid questions, which consists of three components: (1) a decomposition recognizer, which identifies the subparts of a decomposable question, (2) a question rewriter, which composes new sub-questions from the identified subparts, taking into account context from the original question, and (3) an answer synthesis and re-ranking component, which synthesizes and ranks final answers based on candidate answers to the sub-questions. Additionally, this framework deploys an underlying factoid QA system for producing answers to the sub-questions. Any QA system that can associate confidence scores with its answers and can make distinctions between the question and the context in which the question should be interpreted can be adopted in this decomposition framework.

We applied our decomposition framework to address two broad classes of complex factoid questions, parallel and nested decomposition questions. These are distinguished by how the identified sub-questions relate to each other, which in turn affects how the candidate answers to the sub-questions are combined to form the final answers. In order to maintain generality and facilitate domain adaptation, the rule-based patterns for decomposition recognition exploit syntactic characteristics of the question that are indicative of sub-question boundaries. To optimally leverage these patterns, a ML model was trained to properly weigh the possibly overlapping, and occasionally conflicting, patterns.

We demonstrated the impact of our question decomposition approach on a state-of-the-art factoid QA system. On a test set of 1,269 challenging FJ questions, 47% of the questions were found to be parallel decomposable and 20% were nested

https://doi.org/10.1017/S1351324913000326 Published online by Cambridge University Press

decomposable. Overall, the system achieved a statistically significant gain of 1.5% in accuracy on these questions, further increasing the system's lead over human Jeopardy! players' performance. Given that factoid (and often, complex) questions are typically found in several real-world domains (e.g. medical, legal, technical support), we expect our decomposition framework to have broad impact, both in open- and specialized-domain question answering.

References

Chu-Carroll, J., Brown, E., Lally, A., and Murdock, J. W. 2012a. Identifying implicit relationships. *IBM Journal of Research and Development* **56**(3.4): 12:1–10.

Chu-Carroll, J., Fan, J., Boguraev, B., Carmel, D., Sheinwald, D., and Welty, C. 2012b. Finding needles in the haystack: search and candidate generation. *IBM Journal of Research and Development* **56**(3.4): 6:1–12.

Chu-Carroll, J., Fan, J., Schlaefer, N., and Zadrozny, W. 2012c. Textual resource acquisition and engineering. *IBM Journal of Research and Development* **56**(3.4): 4:1–11.

Ferrucci, D., Brown, E., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A., Lally, A., Murdock, J. W., Nyberg, E., Prager, J., Schlaefer, N., and Welty, C. 2010. Building Watson: an overview of the DeepQA project. *AI Magazine* **31**(3): 59–79.

Gondek, D. C., Lally, A., Kalyanpur, A., Murdock, J. W., Duboue, P. A., Zhang, L., Pan, Y., Qiu, Z. M., and Welty, C. 2012. A framework for merging and ranking of answers in DeepQA. *IBM Journal of Research and Development* **56**(3.4): 14:1–12.

Hartrumpf, S. 2008. Semantic decomposition for question answering. In *Proceedings of the 18th European Conference on Artificial Intelligence*, pp. 313–7. Amsterdam, Netherlands: IOS Press.

Kalyanpur, A., Boguraev, B., Patwardhan, S., Murdock, J. W., Lally, A., Welty, C., Prager, J., Coppola, B., Fokoue-Nkoutche, A., Zhang, L., Pan, Y., and Qiu, Z. M. 2012a. Structured data and inference in DeepQA. *IBM Journal of Research and Development* **56**(3.4): 10:1–14.

Kalyanpur, A., Patwardhan, S., Boguraev, B., Lally, A., and Chu-Carroll, J. 2011. Fact-based question decomposition for candidate answer re-ranking. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management (CIKM '11)*, pp. 2045–8. Glasgow, UK: ACM Digital Library.

Kalyanpur, A., Patwardhan, S., Boguraev, B., Lally, A., and Chu-Carroll, J. 2012b. Fact-based question decomposition in DeepQA. *IBM Journal of Research and Development* **56**(3.4): 13:1–11.

Katz, B., Borchardt, G., and Felshin, S. 2005. Syntactic and semantic decomposition strategies for question answering from multiple sources. In *Proceedings of the AAAI Workshop on Inference for Textual Question Answering*, pp. 35–41. Menlo Park, CA: AAAI Press.

Katz, B., Lin, J., and Quan, D. 2002. Natural language annotations for the semantic web. In *Proceedings of International Conference on Ontologies, Databases and Applications of Semantics*, pp. 1317–31. Berlin, Germany: Springer-Verlag.

Lacatusu, F., Hickl, A., and Harabagiu, S. 2006. The impact of question decomposition on the quality of answer summaries. In *Proceedings of the Fifth Language Resources and Evaluation Conference*, pp. 1147–52. Genoa, Italy: European Language Resources Association (ELRA).

Lally, A., Prager, J., McCord, M. C., Boguraev, B., Patwardhan, S., Fan, J., Fodor, P., and Chu-Carroll, J. 2012. Question analysis: how Watson reads a clue. *IBM Journal of Research and Development* **56**(3.4): 2:1–14.

Lin, C., and Liu, R. 2008. An analysis of multi-focus questions. In *Proceedings of the SIGIR 2008 Workshop on Focused Retrieval*, pp. 30–6. Singapore: ACM.

McCord, M. 1989. Slot grammar: a system for simpler construction of practical natural language grammars. In *Proceedings of the International Symposium on Natural Language and Logic*, pp. 118–145. Hamburg, Germany: Springer-Verlag.

https://doi.org/10.1017/S1351324913000326 Published online by Cambridge University Press

McCord, M. C., Murdock, J. W., and Boguraev, B. 2012. Deep parsing in Watson. *IBM Journal of Research and Development* **56**(3.4): 3:1–15.

McNemar, Q. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika* **12**(2): 153–7.

Murdock, J. W., Fan, J., Lally, A., Shima, H., and Boguraev, B. 2012a. Textual evidence gathering and analysis. *IBM Journal of Research and Development* **56**(3.4): 8:1–14.

Murdock, J. W., Kalyanpur, A., Welty, C., Fan, J., Ferrucci, D., Gondek, D. C., Zhang, L., and Kanayama, H. 2012b. Typing candidate answers using type coercion. *IBM Journal of Research and Development* **56**(3.4): 7:1–13.

Prager, J., Brown, E., and Chu-Carroll, J. 2012. Special questions and techniques. *IBM Journal of Research and Development* **56**(3.4): 11:1–13.

Prager, J., Chu-Carroll, J., and Czuba, K. 2004. Question answering by constraint satisfaction: QA-by-dossier with constraints. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pp. 574–81. Barcelona, Spain: ACL.

Saquete, E., Martínez-Barco, P., Muñoz, R., and Vicedo, J. 2004. Splitting complex temporal questions for question answering systems. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pp. 566–73. Barcelona, Spain: ACL.

Soricut, R., and Brill, E. 2004. Automatic question answering: beyond the factoid. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 57–64. Boston, MA: ACL.

Voorhees, E. 2002. Overview of the TREC 2002 question answering track. In *NIST Special Publication 500-251: The Eleventh Text Retrieval Conference (TREC 2002)*, pp. 115–123. Gaithersburg, MD: Department of Commerce, National Institute of Standards and Technology.

Wang, C., Kalyanpur, A., Fan, J., Boguraev, B. K., and Gondek, D. C. 2012. Relation extraction and scoring in DeepQA. *IBM Journal of Research and Development* **56**(3.4): 9:1–12.

Witten, I., and Frank, E. 2000. *Data Mining – Practical Machine Learning Tools and Techniques with Java Implementations*. San Francisco, CA: Morgan-Kaufmann.

https://doi.org/10.1017/S1351324913000326 Published online by Cambridge University Press