CIKM '11: Proceedings of the 20th ACM International Conference on Information and Knowledge Management, Glasgow, UK, October 2011

Fact-Based Question Decomposition for Candidate Answer Re-Ranking

Aditya Kalyanpur Siddharth Patwardhan Branimir Boguraev
Adam Lally Jennifer Chu-Carroll
IBM T.J.Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598
{adityakal,siddharth,bran,alally,jencc}@us.ibm.com

ABSTRACT

Factoid questions often contain one or more assertions (facts) about their answers. However, existing question-answering (QA) systems have not investigated how the multiple facts may be leveraged to enhance system performance. We argue that decomposing complex factoid questions can benefit QA, as an answer candidate is more likely to be correct if multiple independent facts support it. We categorize decomposable questions as parallel or nested, depending on processing strategy required. We present a novel decomposition framework—for parallel *and* nested questions—which can be overlaid on top of traditional QA systems. It contains decomposition rules for identifying fact sub-questions, a question-rewriting component and a candidate re-ranker. In a particularly challenging domain for our baseline QA system, our framework shows a statistically significant improvement in end-to-end QA performance.

Categories and Subject Descriptors

I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*text analysis, language parsing and understanding*

General Terms

Algorithms, Design

Keywords

Question Answering, Question Decomposition

1. INTRODUCTION

We look at complex factoid questions, specifically ones which contain multiple facts related to the correct answer, as in e.g. *Which company with origins dating back to 1876 became the first U.S. company to have 1 million stockholders in 1951?*. Two facts of relevance here are the time-frame of the company's origins, and it becoming the first to reach a particular landmark. We use the decomposed facts to garner independent, but potentially mutually-reinforcing, support for the correct answer from independent sources of evidence.

Recent approaches to QA acknowledge that some questions need decomposing, to tease out information beyond what "single-shot" QA systems generally assume. They tend to defer to discourse and/or semantics of the question, and use complex processes like textual entailment [4], question refocusing [2, 3], or temporal/spatial analysis [6, 2]. Decomposition work has mostly looked at "beyond factoid" questions. In contrast, we develop decomposition to improve quality of QA for a broad set of *factoids*. The questions we use (from the popular TV quiz show Jeopardy!) cover numerous topics, and make for an excellent test bed for open-domain QA.[1]

Our definition of "decomposable" questions—containing independent support for the correct answer—leads to categorizing such questions into *parallel* and *nested*. The example above, and an example from our data—*This country singer who did time in San Quentin was pardoned by Governor Ronald Reagan*—are parallel decomposable: subquestions can be evaluated independently of one another. In contrast, nested questions require sequential processing, with the answer to an 'inner' sub-question plugged into the 'outer': *It was named for Britain's last Stuart monarch, who gave the city its charter in 1708.* In this work we develop a uniform framework for handling decomposable questions of both types, and demonstrate that it is capable of improving the end-to-end performance of a state-of-the-art QA system.

2. FACT-BASED DECOMPOSITION

A single-shot QA system may find answers whose supporting evidence satisfies only some of the question's facts, and ignores the remainder of the question. At the same time, looking at all question terms as a whole, instead of separate meaningful facts, may distract the search for candidate answers. For nested decomposable questions, the missing piece of information implied by the 'inner' fact is often a facilitator to obtaining the correct answer. We would like to be able to enhance such single-shot systems so they can decompose questions (when appropriate) and identify parts thereof to be further processed, in parallel or in sequence.

Figure 1 shows the high-level architecture of such an adaptation. Ours is a "meta" framework overlaid on top of an existing QA system. It can be conceptualized around four main components.

[1]In this data, questions are posed in a declarative format, with highly stylized marking of question focus. This should not detract from referring to them as 'questions'.

CIKM'11, October 24–28, 2011, Glasgow, Scotland, UK.

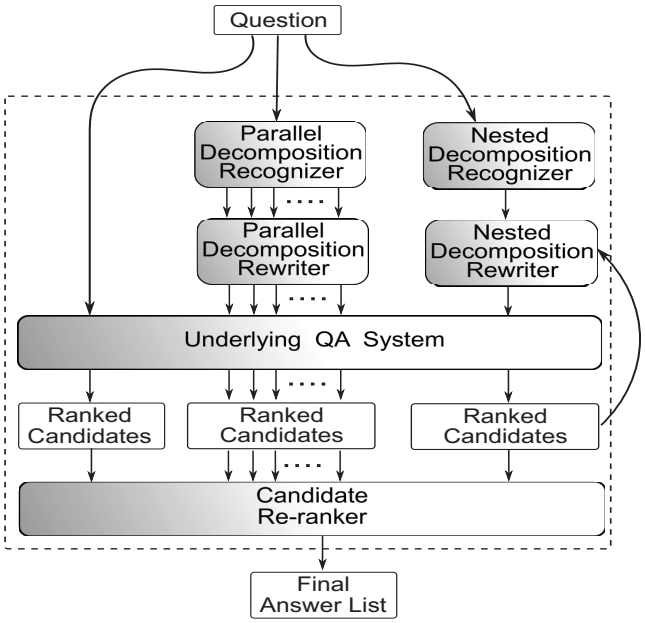

Figure 1: Fact-Based Decomposition Framework

Decomposition Recognizers analyze the question and identify decomposable parts (facts) using largely lexico-syntactic cues (Section 3). *Question Rewriters* reformulate the facts as sub-questions, adding key contextual information (Section 4.1). *Underlying QA System* generates, for any factoid question, a ranked list of answer candidates, each with a *confidence* corresponding to the probability of the answer being correct [1]. *Candidate Re-rankers* combine ranked answers to the original question with solutions for the decomposed facts (sub-questions) and generate a uniformly ranked answer list. Candidate answer confidences are used either by a machine learning-based approach or by a heuristic selection strategy to do the final ranking (Section 4.2).

We use the QA system described in [1]. However, our meta-framework will host any system that satisfies two criteria: it can solve factoid questions by providing answers with confidences reflecting their probability of being correct; it can separate the question's topic information from its main content by weighing the former less than the latter.

The figure highlights the distinction between parallel and nested processing: the parallel decomposition components produce multiple (two or more) sub-questions, submitted to the underlying QA system; the nested components generate inner-outer sub-question pairs, processed via a feedback loop re-invoking the underlying QA capability.

3. DECOMPOSITION RECOGNIZERS

Finding the question segments representative of independent facts about the correct answer is not trivial: multiple facts can be 'weaved' into a single complex question in a variety of ways: modifiers to the focus[2], subordinate clauses, attached prepositional phrases, may all be expressing facts. From a fact-checking perspective, we consider facts to be expressions of relations between the focus and one or more entities (named entities, dates or quotations) as these tend to be more substantial or meaningful. This, coupled with a set of syntactic cues we have found to be reliable indicators for decomposition, allows us to generalize recognition rules for question decomposition.

[2]The *focus* is the part of the question referring to the answer.

3.1 Decomposition Rules

Our rules exploit fine-grained lexico-syntactic information, and fall within three major categories: *independent subtrees*, *composable units* and *segments with qualifiers*. The rules in each rule set target both parallel and nested configurations. Details of rule sets fall outside of the scope of this paper; below we outline some highlights, as representative examples.

Independent Subtrees: Intuitively, potentially independent facts—capturing unique information about the answer—would be in clauses distinct from the rest of the question: e.g. relative or subordinate constructions. In the absence of more focused contextual information (see below), such configurations are indicative of parallel decomposition. The syntactic labels connecting such subtrees to the focus are generally good indicators for "breaking away" a subtree from the question as a decomposable fact. Thus in *The name of this character, first introduced in 1894, comes from the Hindi for "bear"*, an independent fact about the answer is "*this character, introduced in 1894*" (note that "*this character*" is the focus). This category also triggers on conjunctions as decomposition points, as in *Its original name meant "bitter water" and it was made palatable to Europeans after the Spaniards added sugar*.

Configurational information, reinforced by lexical cues, is used to determine the question's decomposition profile, parallel or nested. The syntactic contour of the first example in this section shows that both the main and subordinate clauses characterize the (same) focus entity: a clear indicator that the sub-questions are parallel. Conversely, *A controversial 1979 war film was based on a 1902 work by this author* exhibits a very different set of configurational properties. The focus is just one of several underspecified entities which do not 'share' their facts (e.g. via a common head). This is indicative of nestedness, with the inner sub-question around an unspecified, non-focus element (*"a controversial 1979 war film"*), and the outer taken from the complement of the original question (*"[film] was based on a 1902 work by this author"*). Nested decomposition returns a pair of sub-questions; but there may be more than one way to decompose into an inner-outer.

Composable Units: An alternate strategy for identifying facts is to "compose" them by combining elements from the question. In contrast to "breaking off" independent subtrees, rules in this set combine different parts of the tree into a sub-question. For instance, the focus with its pre- and post-modifiers (if they are sufficiently specific) can be interpreted as separate, parallel, fact(s). Alternatively, finding the focus itself in a modifier position may be a signal of nestedness: in *To honor his work, this man's daughter took the name Maria Celeste when she became a nun in 1616*, the focus is dominated by an underspecified node, *"daughter"*. Traversing such a tree without descending to focus level "carves out" an inner sub-question, itself focused on the dominating (linking) node: *"To honor his work, [this] daughter took the name Maria Celeste ..."*.

Segments with Qualifiers: This group of rules detects focus modifiers which are relative qualifiers: *"the first"*,

"only", *"the westernmost"*. Such qualifiers need to be "completed" by information from elsewhere: *"the third man"* is meaningless without a supporting clause, e.g. *"the third man ... to climb Mt. Everest"*. The rules here combine characteristics of the other two rule sets, to manipulate independent subtrees and compose them with modifiers to the focus.

3.2 Decomposition Filters

Several heuristic filters reduce over-generation; primarily, they discard facts that do not contain a named entity, a quoted string, or a temporal (time or date) expression—this is in line with our definition of fact, earlier in this section.

We also discard facts with significant overlap with the entire question, or with other facts from previously applied rules. This is based on a (partial order) prioritization of rules, reflecting intuitions (borne by inspecting rule results) of how informative the facts generated by the rule are.

4. USING DECOMPOSITION

4.1 Sub-question Rewriting

Submitting a decomposed sub-question, as-is, to the underlying QA system meets serious problems: sub-questions are often much shorter than the original, and tend to not have a unique answer. Moreover, some of the information from the full question that was dropped in the sub-question may offer relevant contextual cues crucial for coming up with the correct answer. In extreme cases, loss of context may lead to recall failures—thus defeating the whole purpose of decomposing in the first place.

Sub-question rewriting provides such crucial contextual information. For a sub-question Q_i, we first obtain the set of all named entities and nouns (minus stopwords) in the original question text that are not present in Q_i. We then insert these keywords into the topic of the original question.[3] Finally, the underlying QA system is given the enriched topic/sub-question pair; recall that our decomposition framework (Section 2) expects to be able to take advantage of the differential weighting of information in a question's topic and content. Sub-question rewriting thus ensures that the larger context of the original question is still taken into account when evaluating a sub-question, although with less weight.

4.2 Candidate Re-ranking

The different decomposition regimes require different strategies for using the sets of candidate answers, with confidences. In the parallel case, a final set of uniformly ranked answers needs to be composed from the set of ranked answer lists produced by independently solving the sub-questions. In the nested case, a subset of the answer list to the inner sub-question needs to be selected for substitution into the outer.

Parallel Decomposition A simple approach to assigning a final score for each candidate answer is to multiply scores for each of the sub-questions—given the assumption that they are independent. However, sub-question rewriting breaks this assumption. Also, different decomposition rules have different precision and recall: thus sub-questions ought not to be weighed equally. In the extreme case of rules producing bad decompositions, it makes sense to fall back to the original question: therefore the confidence of the 'single-shot' pass (Figure 1) should also inform the final decision. Consequently, we train a machine learning model to combine information across sub-question answer confidences.

To capture this range of information, the model uses the following features: whether a candidate was a top answer to the full (non-decomposed) question; confidence value for a candidate answer to full question; number of sub-questions with candidate answer in top 10; and a feature for every (parallel) rule set, whose value is the confidence of the candidate answer for a fact derived by this rule set.

If 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 produces multiple sub-question, the corresponding (rule) feature value for the candidate answer is set to the sum of the confidences obtained for that answer across all the sub-questions. For the machine-learning algorithm, we use the Weka implementation [7] of logistic regression with instance weighting (weights are tuned over the development set).

Nested Decomposition Each candidate answer from the inner sub-question's answer list, if substituted into the outer sub-question, has ramifications on end-to-end accuracy: incorrect answers from the inner lead to incorrect final answers. We rely on the ability of the underlying QA system to produce meaningful confidences for its answers. Thus only the top answer to the inner sub-question is considered for substitution in the outer—if its confidence exceeds some threshold (obtained by tuning over the development set).

With an answer to the inner sub-question, we are now in a position to rewrite the outer. Similar to the case for parallel sub-questions, we also adhere to the policy of providing additional relevant contextual information to the system (cf. Section 4.1).

A simple heuristic strategy appeals to joint probability of the inner-outer pair to perform candidate re-ranking: the confidence of the top answer to the outer sub-question is multiplied with that of the top answer to the inner question; the product is compared with the top-ranking confidence for the full question; the answer with the higher confidence is selected as final.

5. EVALUATION

5.1 Data

Our data set—with ground truth for both training a system and evaluating its performance—is a collection of Jeopardy! question-answer pairs (from `www.j-archive.com`). Given our focus on question decomposition, only Final Jeopardy! (FJ) questions are in our test set: they tend to be complex, with multiple constraints, and typically much harder to answer (for humans and the underlying QA system alike). Approximately 3000 FJ questions are split into 1138 for training, 517 for development and 1269 for testing (blind data).

5.2 Experiments

The parallel decomposition rules were defined and tuned on the development set of 517 questions. The final re-ranking model (with features as in Section 4.2) was built over our FJ training set (Section 5.1). It was trained for logistic regression with instance re-weighting, setting the negative instance weight 0.25 times lower (based on analysis over the develop-

[3]Jeopardy! questions topics are confined to a *category* field.

QA System	End-to-End Accuracy	Decomposable Q Accuracy
PB	635/1269 (50.05%)	339/598 (56.68%)
PD-QR	634/1269 (49.96%)	338/598 (56.52%)
PD+QR	**643/1269 (50.66%)**	**347/598 (58.02%)**
NB	635/1269 (50.05%)	129/255 (50.58%)
ND+QR	**640/1269 (50.43%)**	**134/255 (52.54%)**

Table 1: Evaluating Decomposition

ment set, and motivated by the underlying QA system's tendency to generate many more negative answers than positive ones). To determine the impact of our context-preserving sub-question rewriting strategy (Section 4.1), we modified the algorithm to issue the sub-question as-is, using the original category (topic). The results of applying the parallel decomposition rules followed by the re-ranking model to the 1269 test questions are shown in Table 1.

In the table, PB refers to Parallel Baseline (the performance of the underlying QA system with decomposition disabled), NB refers to Nested Baseline (same configuration as PB but separately run on nested decomposable questions), PD and ND refer to Parallel and Nested Decomposition systems respectively, and QR refers to Question Rewriting.

Nested decomposition strategy (Section 4.2) was similarly evaluated, comparing the (nested) baseline to a heuristic re-ranking approach (whose parameter thresholds were tuned on the development set). Results are also in Table 1.

5.3 Discussion of Results

Table 1 shows the parallel decomposition rules applying to a large fraction of the test set (598 out of 1269 questions). Interestingly, the performance of the baseline QA system on the decomposable set is 56.6%, i.e. 6% higher than the overall performance. One reason for this is that parallel-decomposable questions typically contain more information (more than one fact/constraint to be satisfied by the answer) and the system can, in some cases, exploit this redundancy (e.g. when a fact is strongly associated with the correct answer and there is evidence in the sources supporting this).

Decomposition without sub-question rewriting does not show much impact over the baseline: question context is clearly crucial for QA. With rewriting to maintain context, our parallel decomposition algorithm achieves an improvement of 1.4% (10 gains/2 losses) on the decomposable question set, which translated to an end-to-end gain of 0.6%.

The nested decomposition rules fired on roughly a fourth of the test set (255 out of 1269 questions); 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 is that nested questions are harder to solve than parallel: correct solution to the inner is crucial for finding the answer to the outer, whereas parallel sub-questions are, by definition, independent. Our nested decomposition algorithm using the heuristic re-ranking approach achieves an improvement of 2% (6 gains/1 loss) on the decomposable question set, which translated to an end-to-end gain of 0.4%.

The impact of using both parallel and nested decomposition is a 1.5% gain in accuracy on the decomposable set, and a 1% gain on end-to-end system accuracy (our rules for finding parallel and nested decompositions are disjoint so a given question cannot fall in both classes).

A key point concerning these results is that the baseline QA system represents state-of-the-art in solving Jeopardy! questions. Moreover, our FJ evaluation data is known to be harder than regular Jeopardy! questions. We estimate qualified Jeopardy! players' accuracy on FJ to be 48% (based on player performance statistics from `j-archive`). A gain of 1% end-to-end on such questions is, therefore, a strong improvement. McNemar's test with Yates' correction for continuity [5] found the results to be statistically significant, where significance is assessed for $p < .05$.

6. CONCLUSION

We argue that a question decomposition capability can enhance the quality of factoid QA. We have developed a general-purpose decomposition framework for complex factoid questions, which distinguishes between parallel (independent) and nested (sequential) question types, and accommodates appropriate strategies for solving both. The framework can be overlayed on any QA system that provides answers with confidences, and that considers the topic of the question separate from its main content.

In addition to the typology of decomposable question types (and associated detection rules), we propose a novel question rewriting approach to mitigate the loss-of-context problem when dealing with shorter (non unique) facts, and a re-ranking strategy that suitably combines results of the decomposition analysis with information from the single-shot QA approach.

This decomposition capability brings an improvement to the performance of a state-of-the-art factoid answering QA system—in the domain of Final Jeopardy!, which even qualified human players find difficult—by 1.5% on decomposable questions: a statistically significant gain.

7. REFERENCES

[1] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager, and N. Schlaefer. Building Watson: An overview of the DeepQA project. *AI Magazine*, 2010.

[2] S. Hartrumpf. Semantic Decomposition for Question Answering. In *Proceedings of ECAI-18*, Greece, 2008.

[3] B. Katz, G. Borchardt, and S. Felshin. Syntactic and Semantic Decomposition Strategies for Question Answering from Multiple Sources. In *Proceedings of the AAAI Workshop on Inference for Textual Question Answering*, pages 35–41, Pittsburgh, PA, July 2005.

[4] F. Lacatusu, A. Hickl, and S. Harabagiu. The Impact of Question Decomposition on the Quality of Answer Summaries. In *Proceedings of LREC-5*, Italy, 2006.

[5] Q. McNemar. Note on the Sampling Error of the Difference Between Correlated Proportions or Percentages. *Psychometrika*, 12(2):153–157, 1947.

[6] E. Saquete, P. Martínez-Barco, R. Muñoz, and J. Vicedo. Splitting Complex Temporal Questions for Question Answering Systems. In *Proceedings of the 42nd ACL*, Spain, July 2004.

[7] I. Witten and E. Frank. *Data Mining—Practical Machine Learning Tools and Techniques.* Morgan–Kaufmann, San Francisco, CA, 2000.