Structure Mapping for Jeopardy! Clues

J. William Murdock

IBM T.J. Watson Research Center,
P.O. Box 704,
Yorktown Heights, NY 10598
murdockj@us.ibm.com

Abstract. The Jeopardy! television quiz show asks natural-language questions and requires natural-language answers. One useful source of information for answering Jeopardy! questions is text from written sources such as encyclopedias or news articles. A text passage may partially or fully indicate that some candidate answer is the correct answer to the question. Recognizing whether it does requires determining the extent to which what the passage is saying about the candidate answer is similar to what the question is saying about the desired answer. This paper describes how structure mapping [1] (an algorithm originally developed for analogical reasoning) is applied to determine similarity between content in questions and passages. That algorithm is one of many used in the Watson question answering system [2]. It contributes a significant amount to Watson's effectiveness.

1 Introduction

Watson is a question answering system built on a set of technologies known as DeepQA [2]. Watson has been customized and configured to compete at Jeopardy!, an American television quiz show. Watson takes in a question and produces a ranked list of answers with confidence scores attached to each of these answers.

One of the stages in the DeepQA question answering pipeline is deep evidence scoring. This stage receives as input a question and a candidate answer in the context of some supporting evidence (typically a passage containing that answer). Questions typically have a focus identified for them (i.e., the term in the question indicating the answer being sought). For example, a deep evidence scorer could be given a question like "*He was the first U.S. President*" and a passage like "*George Washington was the first U.S. President*." "*He*" in the question will be marked as the focus. If the candidate answer is "*George Washington*," each of the deep evidence scorers will attempt to determine the extent to which what the passage says about the "*George Washington*" addresses what the question asks about the "*He*". In this example, there is a perfect match, and all of Watson's deep evidence scoring mechanisms will conclude that this passage strongly supports the specified answer. However, other passages may answer the question less directly, or provide evidence for only a portion of what the question is asking for (e.g., that Washington was *a* president).

The examples above do not require any explicit analogy. One could envision passages that say (for example) that (a) Charles de Gaul was a great French general who fought for the liberation of France, (b) that Charles de Gaulle was the first president

A. Ram and N. Wiratunga (Eds.): ICCBR 2011, LNAI 6880, pp. 6–10, 2011.

of the fifth republic of France, and (c) that George Washington was a great American general who fought for the liberation of the U.S.; by analogy, one might suspect that George Washington was the first president of the U.S. DeepQA does not explicitly do reasoning of this sort, but may in future work.

This paper provides a detailed description of one of Watson's deep evidence scoring algorithms: the Logical Form Answer Candidate Scorer (LFACS). LFACS uses a logical form of a question (e.g., a Jeopardy! clue) containing a specified focus term and the logical form of a passage containing a specified candidate. LFACS employs a structure mapping algorithm similar to the one described in [1]. LFACS embodies a variety of specializations of structure mapping that are driven by the nature of its task. For example, LFACS is *pragmatic* in the sense described in [3], because it has a specific inference it is intended to draw: the extent to which the candidate answer in the passage corresponds to the answer in the clue.

2 Role in the Architecture

The DeepQA architecture is described in [2]; this section provides a minimal description of the architecture to explain the context in which LFACS is used. DeepQA begins with **question analysis**, which applies a variety of natural-language processing algorithms to the question text. These algorithms include general purpose text processing such as parsing and semantic relation detection. They also include processing that is specific to analyzing questions, e.g., determining the focus. In Jeopardy! a question focus is often denoted by a pronoun with no anaphor or a common noun with the word "*this*" as a determiner. For example, in the question "*Ambrose Bierce penned this sardonic reference work in 1906*," the focus is "*work*." The focus is defined to be the term in the question that would correspond to the answer in a corresponding assertion.

Question analysis in DeepQA is followed by **primary search and candidate generation**, which finds candidate answers in variety of sources. Some of those sources are natural-language text while others are structured sources such as knowledge bases. Answers are subjected to preliminary scoring (including answer typing, etc.), and those answers that seem poor (i.e., have a confidence score below a fixed threshold, according to a statistical model) are filtered out.

All candidate answers that pass through the filter are then processed by **supporting evidence retrieval**. That component conducts a search for passages that contain the candidate answer and as many other terms from the question as possible. This retrieval step provides a set of potentially relevant passages for each answer, regardless of where it was originally found (text, knowledge bases, etc.). Candidate answers that were found in text will also have one or more passages from the primary search. Passages from both types of search are used as supporting passages.

The supporting passages are analyzed in **deep evidence scoring**, in which a variety of algorithms assess the degree to which the passage provides evidence in support of some candidate answer. LFACS, described below, is one of these deep evidence scoring components.

The **final merging and ranking** step combines equivalent candidate answers (e.g., "*Richard Nixon*" and "*Richard M. Nixon*") and determines the confidence that each answer is correct. It ranks the answers by their confidence scores. The final merging and ranking component uses statistical machine learning; the features used to compute a confidence for each answer come from algorithms throughout the pipeline. LFACS is one source of features used by this component.

3 Syntactic-Semantic Graphs

LFACS reasons over syntactic-semantic graphs of both the question and the passage. In these graphs, nodes are terms in the clue (e.g., a word or a proper name) and edges encode syntactic and/or semantic relations among those terms. The syntactic portions of the graph are derived from an English-Slot Grammar (ESG) parse [4]. The semantic portions of the graph are derived from pattern-based relation detectors. Syntactic relations are useful for identifying similarity when questions and passages have a similar structure (e.g., "He wrote *Utopia*" – "Thomas More wrote *Utopia*"). Semantic relations are useful when passages use different structures with equivalent meaning (e.g., "He wrote *Utopia*" – "Thomas More, author of *Utopia*"). Relation detection is very challenging and the relation detection capabilities in DeepQA, while very precise, have only a moderate level of coverage. LFACS can be effective when content in passages have similar structure or when they have similar semantics that fall within the coverage of our relation detectors. Because syntactic and semantic relations are combined in a single graph, LFACS can combine insights from each. For example, consider the following actual Jeopardy! clue:

It's believed Shakespeare wrote part of a 1595 play about this "Utopia" author.

Some content in the clue is covered by semantic relations such as the one between an author and a work by that author. However, there are other key relationships in this clue such as the one between a play and the person that the play is about. DeepQA does not have recognizers for this relationship, but is able to parse the text. Consider the following (made-up) sample passage:

We saw a 16th century play about Thomas More, who wrote Utopia*.*

The syntactic-semantic graph for this passage a semantic (*authorOf*) edge between *Thomas More* and *Utopia*; that edge matches the corresponding semantic edge in the graph of the clue. In addition, passage has syntactic edges that correspond to syntactic edges in the clue. *Thomas More* in the passage is the object of the preposition *about*, while the focus of the clue is the object of the preposition *about* in the clue. As a result the matching algorithm (see next section) is able to align the following terms in the clue to terms in the passage using semantic and/or syntactic edges: *1595*, *play*, *about*, *Utopia*, *author*. There are still some important terms in the clue that are not covered by this passage (e.g., *Shakespeare*). Our algorithm assesses the quantity and importance of the terms that it is able to align and asserts a numerical value for how strong it considers the match to be; that numerical value is used by the DeepQA final merger as one of the features that influences the evaluation of answers.

4 Algorithm

LFACS performs a form of structure mapping. The algorithm is similar to the one described in [1], with customization to reflect the nature of the content (extracted NLP results), the fact that LFACS has a single pre-specified inference to draw: Specifically, LFACS is trying to judge whether the passage provides support for a specific, designated candidate answer. Below are the key steps in structure mapping that are defined in [1], with descriptions of how those steps are realized in LFACS:

- Local Match Construction: LFACS matches both edges and nodes. Edges are matched using a formal ontology, e.g., the *authorOf* relation is a subrelation of the *creatorOfWork* relation. Nodes are matched using a variety of resources for determining equivalent terms, e.g., WordNet [5], Wikipedia redirects, and has specialized logic for matching dates, numbers, etc.
- Global Map Construction: Unlike [1], LFACS is only concerned with global matches that align the focus to the specified candidate answer. Thus global map construction begins with the focus and candidate answer and search outward from those nodes through the space of local matches. As in [1], the global match construction process ensures consistency of global maps, requiring that no single node in the question map to multiple nodes in the passage.
- Candidate Inference Construction: LFACS omits this step because the inference to be drawn is implied by its inputs (aligning the focus to the candidate answer).
- Match Evaluation: As in [1], the total score for a match in LFACS is the sum of the match scores for the local match hypotheses included in the maximal consistent global map. Local match scores in LFACS are computed using inverse-document frequency (IDF) from our text corpus. Terms with high IDF scores occur rarely in the corpus so the fact that they align with the clue is less likely to be a coincidence and thus more likely to imply that the answer is correct.

In using this algorithm, we have encountered a wide variety of technical issues that are specific to natural-language. For example, some concepts can be expressed as either a verb or a noun (e.g., *destroy-destruction*). We address those issues through some combination of graph preprocessing (e.g., adding edges to indicate the logical subject of *destruction* during relation detection) and specialized logic that is internal to the local match construction (e.g., allowing the *destroy* to match *destruction*).

Our approach to generating local match hypotheses mostly focuses on determining equivalence (or at least rough equivalence) between nodes. This focus reflects the fact that we are interested in similarity, but not analogy *per se*. If we were to try to address examples like the Charles de Gaul analogy in the introduction of this paper, we would need to relax those restrictions and adjust the confidence in our conclusions accordingly. This may be extremely important in domains where there is less direct evidence involving the candidate answers.

5 Evaluation and Conclusions

Detailed evaluations of deep evidence scoring components will be presented in a future publication. LFACS has statistically significant impact on question answering

accuracy when included in either a simple baseline DeepQA question answering system or to the complete Watson question answering system that competed with human grand champions. This impact, while significant, is small: less than half of one percent in the full system; the full system has an enormous number of answer scoring components and there is a great deal of overlap in the signal they provide. Other deep evidence scoring components in DeepQA (e.g., counting term matches, comparing word order) are more aggressive in what they consider to be a match. These aggressive components have the disadvantage that they do not draw on the full richness of the syntactic and semantic structure but the advantage that they can draw evidence from passages that have little structural similarity to the question.

The impact of LFACS when added to the simple baseline was smaller than that of the more aggressive components. However, in the complete system (containing many more features), the impact of LFACS (while small in an absolute sense) is larger than the impact of those components. The effect of ablating *all* of the deep evidence scoring components in the full system is much bigger than the effects of ablating any of them. These results have important implications for developers of question answering (or similar) technology. Simple, aggressive approaches are well-suited to quickly and easily attaining moderate effectiveness. However, as a system becomes more sophisticated, the opportunities for components of that sort to have impact becomes very limitted. In those cases, more algorithms such as LFACS that make effective use of syntatic and/or semantic structure can further enhance the effectiveness of a question answering system. As a result, additional and improved algorithms of this sort that draw on the full richness of our deep syntatic and semantic analysis are an important area for future research.

References

1. Falkenhainer, B., Forbus, K., Gentner, D.: The Structure Mapping Engine: Algorithm and examples. Artificial Intelligence 41, 1–63 (1989)
2. Ferrucci, D., Brown, E., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A., Lally, A., Murdock, J.W., Nyberg, E., Prager, J., Schlaefer, N., Welty, C.: Building Watson: An Overview of the DeepQA Project. AI Magazine 31(3), 59–79 (2010)
3. Forbus, K., Oblinger, D.: Making SME greedy and pragmatic. In: Proceedings of the Cognitive Science Society (1990)
4. McCord, M.C.: Slot Grammar: A System for Simpler Construction of Practical Natural Language Grammars. In: Studer, R. (ed.) Natural Language and Logic. LNCS, vol. 459, pp. 118–145. Springer, Heidelberg (1990)
5. Miller, G.A.: WordNet: A Lexical Database for English. Communications of the ACM 38(11), 39–41 (1995)