Semantics-based Graph Approach to Complex Question-Answering

Tomasz Jurczyk
Mathematics and Computer Science
Emory University
Atlanta, GA 30322, USA
tomasz.jurczyk@emory.edu

Jinho D. Choi
Mathematics and Computer Science
Emory University
Atlanta, GA 30322, USA
jinho.choi@emory.edu

Abstract

This paper suggests an architectural approach of representing knowledge graph for complex question-answering. There are four kinds of entity relations added to our knowledge graph: syntactic dependencies, semantic role labels, named entities, and coreference links, which can be effectively applied to answer complex questions. As a proof of concept, we demonstrate how our knowledge graph can be used to solve complex questions such as arithmetics. Our experiment shows a promising result on solving arithmetic questions, achieving the 3-folds cross-validation score of 71.75%.

1 Introduction

Question-answering has lately gained lots of interest from both academic and industrial research. Services such as Yahoo! Answers[1] or Quora[2] provide platforms for their users to ask questions to one another; however, answer accuracy or response rate of these services strongly depends on the users' willingness of sharing their knowledge, which is not always consistent. This kind of inconsistency has led many researchers to focus on developing question-answering systems that retrieve, analyze, and answer questions without much human engagement.

Although the task of question-answering has been been well-explored, several challenges still remain. One of such challenges concerns about architectural aspects of meaning representation. Thanks to years of research on statistical parsing, several tools are already available that provide rich syntactic and semantic structures from texts. Output of these tools, however, often needs to be post-processed into more complicated structures, such as graphs of knowledge, in order to retrieve answers for complex questions. These graphs consist of relations between entities found not only within a sentence, but also across sentences. Vertices and edges in these graphs represent linguistic units (e.g., words, phrases) and their syntactic or semantic relations, respectively.

Robustness of handling several types of questions is one of the key aspects about a question-answering system; yet most of previous work had focused on answering simple factoid questions (Yao et al., 2013; Yih et al., 2013). Recently, researchers started focusing on solving complex questions involving arithmetics or biological processes (Hosseini et al., 2014; Berant et al., 2014). A complex question can be described as a question requiring the collection and synthesis of information from multiple sentences (Chali and Joty, 2008). The more complex the questions become, the harder it is to build a structural model that is general enough to capture information for all different types of questions.

This paper suggests an architectural approach of representing entity relations as well as its application to complex question-answering. First, we present a systematic way of building a graph by merging four kinds of information: syntactic dependencies, semantic role labels, named entities, and coreference links, generated by existing tools (Section 3). We then demonstrate, how our graph can be coupled with statistical learning to solve complex questions such as arithmetic, which requires understanding of the entire

[1] https://answers.yahoo.com
[2] https://www.quora.com

Proceedings of NAACL-HLT 2015 Student Research Workshop (SRW), pages 140–146,
Denver, Colorado, June 1, 2015.

context (Section 4). Our experiments show that it is possible to retrieve document-level entity relations through our graph, providing enough information to handle such complex questions (Section 5).

2 Related Work

Punyakanok et al. (2004) presented a system using edit distance between question and potential answer pairs, measured by the number of required transformations of their dependency trees. Heilman and Smith (2010) presented a more sophisticated system finding the most efficient tree transformation using a greedy search. Cui et al. (2005) proposed a system utilizing fuzzy relation matching guided by statistical models. Yao et al. (2013) described an approach taking both an edit distance and sequence tagging for selecting finer-grained answers within answer candidates. All the work above leverages dependency tree matching similar to ours; however, our approach performs matching through semantic relations as well as coreference links, and also is designed for handling complex questions whereas the others mainly focused on factoid questions.

Kushman et al. (2014) described an approach for predicting sentence-to-equation alignments for solving arithmetic questions. Hosseini et al. (2014) presented a system predicting verb categories, and constructing equations from the context using these categories. Berant et al. (2014) proposed an approach that extracted structures from biological processes, and mapped each question to a query form. Our work is related to the first two work; however, it is distinguished in a way that our constructed graph is not designed to handle just arithmetic questions, but complex questions in general.

Our work is also related to research of aligning text into a set of entities and instances describing states of the world. Snyder and Barzilay (2007) presented an approach for solving text-to-database alignment as a structured multi-label classification. Vogel and Jurafsky (2010) presented a learning system that followed navigational paths based on natural language by utilizing apprenticeship from directions on the map paired with human language. Chambers and Jurafsky (2009) presented an unsupervised learning system for narrative schemas based on coreferent arguments in chains of verbs. Pourdamghani et al. (2014) and Pan et al. (2015) presented Abstract Meaning Representation (AMR) consisting of multi-layered relations for English sentences. Our semantics-based graph shares a similar idea with AMR; however, our graph is constructed from existing structures such as dependency trees and semantic roles, whereas AMR requires its won annotation, which could be manual intensive work for building statical parsing models.

3 Semantics-based Knowledge Approach

3.1 Motivation

Our motivation arises from both the complexity and the variety of questions and their relevant contexts. The complexity concerns with exploiting syntactic dependencies, semantic role labels, named entities, and coreference links all together for finding the best answers. For arithmetic questions, such complexity comes from the flow of entity relations across sentences and semantic polarities of verb predicates, which are required to transform the contexts in natural language into mathematical equations.

The variety concerns with robustly handling various types of questions. It is relatively easier to develop an architecture designated to handle just one type of questions (e.g., a system to extract answers for factoid questions) than many different types of questions (e.g., opinions, recommendations, commentaries). In this section, we present a semantic-based knowledge approach (constructed graph) that not only conveys relations from different layers or linguistic theories, but also is effective for finding answers for various types of questions.

3.2 Components

Given a *document*, our system first parses each sentences into a dependency tree, then finds predicate-argument structures on top of the dependency tree. Once sentences are parsed, coreference links are found for nodes across all trees. Finally, each dependency node gets turned into an *instance*, which can be linked to other related instances. Multiple instances can be grouped together as an *entity* if they are coreferent. Our graph is semantically driven because semantic predicate-argument relations take precedence over syntactic dependencies when both exist.

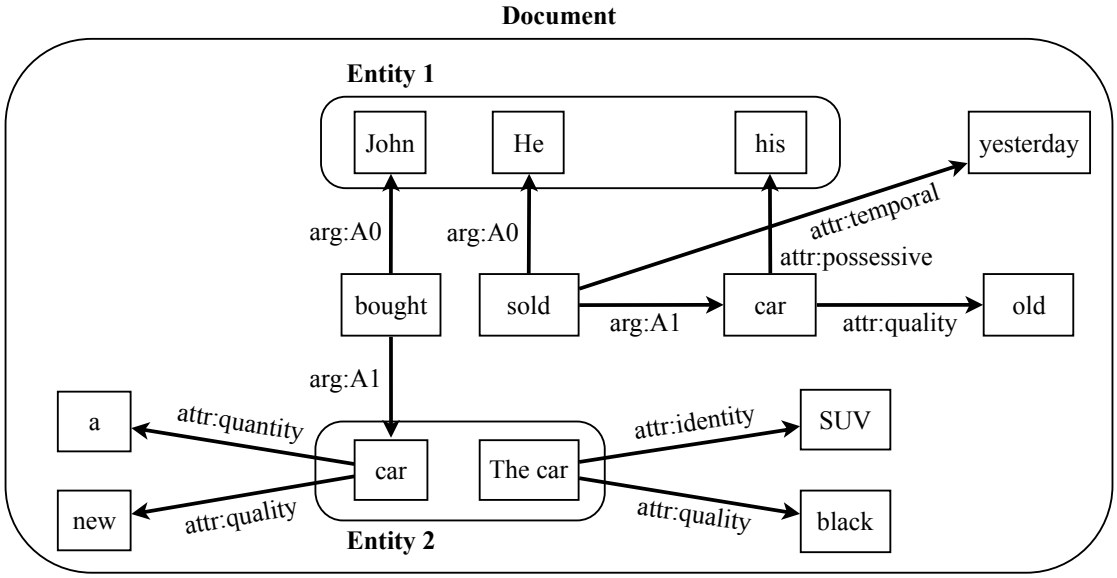


Figure 1: Example of our semantic-based graph given three sentences: *John bough a new car*, *The car was black SUV*, and *He sold his old car yesterday*.

Document

A *document* contains a graph consisting of a set of entities, instances, and relations between the instances (Figure 1). A document can be small as a microblog or big as the entire Wikipedia articles.

Entity

An *entity* can be described as a set of instances referring to the same object mostly found through coreference resolution. In Figure 1, although *John*, *He*, and *his* are recognized as individual instances, they are grouped into one entity because they all refer to *John*. Maintaining these relations is crucial for answering complex questions.

Instance

An *instance* is the atomic-level object in our graph that usually represents a word-token, but can also represent compound words (e.g., *New York*), multi-word expressions, etc. The instance is linked to other instances as a predicate, an argument, or an attribute.

Predicate & Argument

An instance is a *predicate* of another instance if it forms any argument structure (Palmer et al., 2005). Currently, our graph takes non-auxiliary verbs and a few eventive nouns as predicates provided by a semantic role labeler. An instance is an *argument* of another if it is required to complete the meaning of the other instance. In Figure 1, *John* and *car* are arguments of *bought* because they are necessary to give an understanding of *bought*. We plan to improve these relations through semantic parsing in the future.

The *predicate* and *argument* relations represent both semantic and syntactic relations between instances in the document. Semantic role labels in (Palmer et al., 2005) and dependency labels in (Choi and Palmer, 2012) are used to represent semantic and syntactic relations in our graph. Our experiments show that these relations play a crucial role in answering arithmetic questions (Section 5).

Attribute

An instance is an *attribute* of another if it is not an argument but gives extra information about the other instance. While an argument completes the meaning of its predicate, an attribute augments the meaning with specific information. In Figure 1, *new* is not an argument but an attribute of *car* because this information is not required for understanding *car*, but provides finer-grained information about the car.

Attributes can be shared among instances within the same entity. In Figure 1, the attributes *new* and *black* are shared between instances *car* and *the car*. This is particularly useful for questions requiring information scattered across sentences. Table 1 shows the types of attributes that we have specified so far.

This list will be continuously updated as we add more question types to our system.

Type	Description
Locative	Geographical or relative location information (e.g., *New York*, *near my house*).
Temporal	Absolute or relative temporal information (e.g., *tomorrow noon*, *2 years ago*).
Possessive	Possessor of this instance (e.g., *his*, of *Mary*).
Quantity	Absolute or relative quantity information (e.g., *two* books, *few* books).
Quality	Every other kind of attributes.

Table 1: List of attributes used in our graph.

3.3 Graph construction

Algorithm 1 shows a pseudo-code for constructing our graph given a dependency tree, consisting of syntactic and semantic relations, and coreference links.

Input: D: a dependency tree,
C: a set of coreference links.
Output: G: Graph.
foreach node N in D **do**
  if N.skip() **then**
    continue;
  else if N.isArgument() **then**
    $P \leftarrow N$.getPredicate();
    $L \leftarrow N$.getArgumentLabel();
    G.addArgument(P, N, L);
  else if N.isAttribute() **then**
    $A \leftarrow N$.getAttributeHead();
    $L \leftarrow N$.getAttributeType();
    G.addAttribute(A, N, L);
  else
    $H \leftarrow N$.getSyntacticHead();
    $L \leftarrow N$.getSyntacticLabel();
    G.addArgument(H, N, L);
  end
  if C.hasEntityFor(N) **then**
    $E \leftarrow C$.getEntityFor(N)
    G.addToEntity(E, N);
end

Algorithm 1: Graph constructing algorithm.

Every node in the dependency tree has exactly one syntactic head and can be a semantic argument of zero to many predicates. For each node, it first checks if this node should be added to the graph (i.e., auxiliary verbs are not added). If it should, it checks it is a semantic argument of some predicate. If not, it checks if it is an attribute of some instance. By default, it becomes an argument of its syntactic head. Finally, it gets added to an entity if it is coreferent to some other instance. Moreover, our graph is also designed to support weights of vertices and edges. Now, we assign a value of 1 as a weight for every element, but we plan to extend our work by determining the importance of different weights for specific semantic relations. We believe that an intelligent weighting system will improve the overall accuracy of the system by enhancing the matching process.

4 Case Study

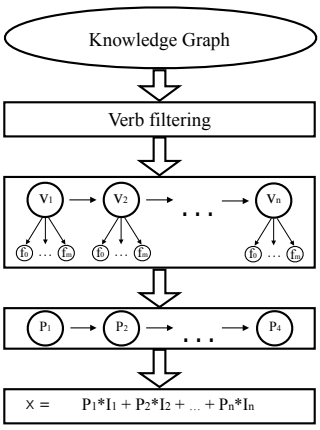
4.1 Arithmetic questions

This section demonstrates our approach to the application of complex question-answering, targeted on arithmetic questions. The purpose of this section is to show a proof of concept that our graph can be effectively applied to answer such questions. For our experiments, we take a set of arithmetic questions used for elementary and middle school students. These questions consist of simple arithmetic operations such as addition and subtraction. Table 2 shows a sample of these questions.

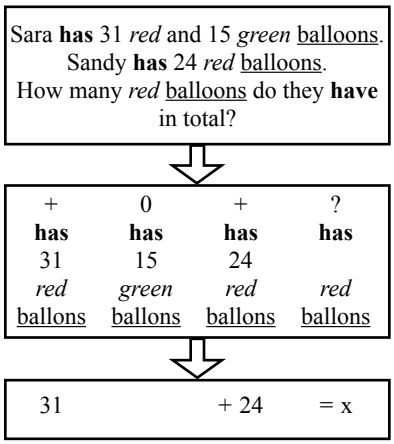
The main challenge of this task is mostly related to the contiguous representation of state changes. The question at the end concerns about either the start state, the transitions, or the end state of a specific theme (e.g., *pizza*, *kitten*). Therefore, simplistic string matching approaches, which would have worked well on factoid questions, would not perform well on this type of questions. Another challenge is found by coreference mentions in these questions. Arithmetic questions generally consist of multiple sentences such that coreference resolution plays a crucial role for getting high accuracy. These issues are further discussed in Section 5.4.

4.2 Verb polarity sequence classification

We turn the task of arithmetic question-answering into a sequence classification of verb polarities. We



(a) Flow of execution in our system for solving arithmetic questions. First, the verb filtering process is applied to select verbs in all sentences (V_i), which share the same semantic argument with the question. Given the selected verbs, their features (f_i) are extracted and the polarities (P_i) are predicted by a statistical model. Finally, the equation X is formed, where polarities are multiplied by the quantities of the arguments.



(b) Flow of execution for the example document. First, verbs are filtered and selected for the polarity selection. Next, all necessary information (numericals, themes etc.) is collected and organized into states. Finally, based on the verbs polarity, equation is being formed.

Figure 2: Flow of execution in general (a) and for an example document (b).

believe the verbs need to be classified in sequence because the same verb can convey different polarities in different contexts. Three types of verb polarities are used: +, −, and 0. Given the list of sentences in each question and the equation associated with it (Table 2), we map each verb with its polarity by comparing their quantities. '+' and '−' are assigned to verbs whose arguments show a plus sign or a minus sign in the equation, respectively. '0' is assigned to verbs whose arguments do not appear in the equation. This information is used to build a statistical model, which is used for decoding.

Arithmetic questions often contain verbs whose arguments are not relevant to the final question. For instance, in "*Jason has 43 blue and 16 red marbles. Tom has 24 blue marbles. How many blue marbles do they have in all?*", "*16 red marbles*" is more like a noise to answer this question. Our approach classifies such verbs as 0 so that they do not participate into the final equation. Once the equation is form, it is trivial to solve the problem using simple algebra.

Our approach is distinguished from some of the previous work where each verb is categorized into multiple classes (Hosseini et al., 2014) in a sense that our verb classes are automatically derived from the equations (no extra annotation is needed). Furthermore, our approach can be extended to more complicated operations such as multiplication and division as long as the correct equations are provided. The dataset used in Kushman et al. (2014) contains this type of questions and we plan to apply our approach on this dataset as the future work.

Question	Equation
A restaurant served 9 pizzas during lunch and 6 during dinner today. How many pizzas were served today?	x = 9 + 6
Tim's cat had kittens. He gave 3 to Jessica and 6 to Sara. He now has 9 kittens. How many kittens did he have to start with?	x = 3 + 6 + 9

Table 2: Sample of arithmetic questions.

5 Experiments

5.1 Data

For our experiments, we use the arithmetic dataset provided by the Allen Institute.[3] The dataset consists of 395 arithmetic questions together with their

[3] allenai.org/content/data/arithmeticquestions.pdf

equations and answers. We parsed all data using the dependency parser, the semantic role labeler, the named entity tagger, and the coreference resolution in ClearNLP (Choi and McCallum, 2013; Choi, 2012).[4] We then split the dataset into 3-folds for cross validation in a way that the polarity distributions are similar across different sets (Table 3).

5.2 Features

The following features are used for our experiments:

- Semantic role labels; especially numbered arguments as in PropBank (Palmer et al., 2005).
- Sequence of verbs and arguments whose semantic roles are recognized as 'themes'.
- Frequency of verbs and theme arguments in the current context.
- Similarity between verbs and theme arguments across sentences.
- Distance of the verb to the final question.

Given our graph, it was trivial to extract all features.

5.3 Machine learning

To build statistical models, we use a stochastic adaptive subgradient algorithm called ADAGRAD that uses per-coordinate learning rates to exploit rarely seen features while remaining scalable (Duchi et al., 2011). This is suitable for NLP tasks where rarely seen features often play an important role and training data consists of a large number of instances with high dimensional features. We use the implementation of ADAGRAD in ClearNLP using the hinge-loss, and take their default hyper-parameters (learning rate: $a = 0.01$, termination criterion: $r = 0.1$).

5.4 Evaluation

Table 3 shows the distributions of each fold and the accuracy of our system in answering arithmetic questions. Our cross-validation score is 71.75%, which is promising given how complex these questions are. Hosseini et al. (2014) were able to achieve 77.7% accuracy on the same dataset, which is higher than our result. However, our main goal for these experiments remains as to prove that our graph can be utilized to answer complex questions.

[4] http://www.clearnlp.com

We also analyzed errors found in our experiment. The majority of errors were caused by errors from dependency parsing, semantic role labeling, or coreference resolution. For instance, verbs are not recognized correctly in some dependency trees, which becomes a major factor of decreasing accuracy. Also, semantic role labels sometimes were incorrectly assigned, which extremely influenced the accuracy of our system. As mentioned earlier, coreference resolution remains as one of the main challenges in handling complex questions. We will explore ways of improving these NLP tools, hoping to achieve higher accuracy for answering complex questions.

	1st fold	**2nd fold**	**3rd fold**
# of questions	118	118	118
# of verbs	418	423	420
# of + verbs	326	330	328
# of − verbs	51	51	51
# of 0 verbs	41	42	41
Accuracy	67.80	76.27	71.19

Table 3: Distributions and accuracies of all folds.

6 Conclusion and future work

This paper presents semantics-based knowledge approach for answering different types of complex questions. As a proof of concept, we demonstrate the application of our graph for arithmetic question-answering. By using the grounded knowledge in our graph, our system was able to extract appropriate features and build a statistical model for recognizing verb polarities that effectively solved arithmetic questions. Our system shows a promising result for answering arithmetic questions. Although we view the problem of solving arithmetic questions as a significant step towards complex question-answering, numerous challenges still remain, not only in the sub-domain of arithmetic questions, but also in other types of complex questions.

In the future, we plan to extend our work by exploring new features for the statistical model. Also, we plan to make improvement in dependency parsing, semantic role labeling, and coreference resolution through error analysis of our question-answering system. Finally, we will try to apply our knowledge approach to other types of complex questions.

References

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D. Manning. 2014. Modeling Biological Processes for Reading Comprehension. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, EMNLP'14, pages 1499–1510, Doha, Qatar, October. Association for Computational Linguistics.

Yllias Chali and Shafiq Joty. 2008. Selecting Sentences for Answering Complex Questions. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, EMNLP'08, pages 304–313, Honolulu, Hawaii, October.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised Learning of Narrative Schemas and their Participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, ACL'09, pages 602–610, Suntec, Singapore, August.

Jinho D. Choi and Andrew McCallum. 2013. Transition-based Dependency Parsing with Selectional Branching. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL'13, pages 1052–1062, Sofia, Bulgaria, August.

Jinho D. Choi and Martha Palmer. 2012. Guidelines for the Clear Style Constituent to Dependency Conversion. Technical report, Technical Report 01-12, University of Colorado at Boulder.

Jinho D. Choi. 2012. *Optimization of Natural Language Processing Components for Robustness and Scalability*. Ph.D. thesis, University of Colorado Boulder.

Hang Cui, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. 2005. Question Answering Passage Retrieval Using Dependency Relations. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 400–407. ACM.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *The Journal of Machine Learning Research*, 12(39):2121–2159.

Michael Heilman and Noah A Smith. 2010. Tree Edit Models for Recognizing Textual Entailments, Paraphrases, and Answers to Questions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL'10, pages 1011–1019.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to Solve Arithmetic Word Problems with Verb Categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, EMNLP'14, pages 523–533, Doha, Qatar, October.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to Automatically Solve Algebra Word Problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, ACL'14, pages 271–281, Baltimore, Maryland, June.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational linguistics*, 31(1):71–106.

Xiaoman Pan, Taylor Cassidy, Ulf Hermjakob, Heng Ji, and Kevin Knight. 2015. Unsupervised Entity Linking with Abstract Meaning Representation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies*, NAACL'15.

Nima Pourdamghani, Yang Gao, Ulf Hermjakob, and Kevin Knight. 2014. Aligning English Strings with Abstract Meaning Representation Graphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, EMNLP'14, pages 425–429, Doha, Qatar, October.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2004. Mapping Dependencies Trees: An Application to Question Answering. In *Proceedings of AI&Math*, pages 1–10.

Benjamin Snyder and Regina Barzilay. 2007. Database-Text Alignment via Structured Multilabel Classification. In *IJCAI*, pages 1713–1718.

Adam Vogel and Daniel Jurafsky. 2010. Learning to Follow Navigational Directions. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL'10, pages 806–814, Uppsala, Sweden, July.

Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013. Answer Extraction as Sequence Tagging with Tree Edit Distance. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL'13, pages 858–867, Atlanta, Georgia, June.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question Answering Using Enhanced Lexical Semantic Models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, EMNLP'13, pages 1744–1753, Sofia, Bulgaria, August.