Natural Language Question Answering with Goal-directed Answer Set Programming

Kinjal Basu, Gopal Gupta

The University of Texas at Dallas, Richardson, Texas, USA

Abstract
Understanding the meaning of a text is a fundamental challenge of natural language understanding (NLU) research. An ideal NLU system should process a language in a way that is not exclusive to a single task or a dataset. To do so, knowledge driven generalized semantic representation for English text is utmost important for any NLU applications. Ideally, for any realistic (human like) NLU system, commonsense reasoning must be an integral part of it and goal directed answer-set-programming (ASP) is indispensable to do commonsense reasoning. Keeping all of these in mind, we have developed various NLU application ranging from visual question answering to a conversational agent. In contrast to existing purely machine learning-based methods for the same tasks, we have shown, our applications not only maintain high accuracy but also provides explanation for the answer it computes.

Keywords
Answer Set Programming, Natural Language Understanding, Question Answering Conversational Agent

1. Introduction

The long term goal of natural language understanding (NLU) research is to make applications, e.g., chatbots and visual/textual question answering (QA) systems, that act exactly like a human assistant. A human assistant will understand the user's intent and fulfill the task. The task can be answering questions about a story or an image, giving directions to a place, or reserving a table in a restaurant by knowing user's preferences. Human level understanding of natural language is needed for an NLU application that aspires to act exactly like a human. To understand the meaning of a natural language sentence, humans first process the syntactic structure of the sentence and then infer its meaning. Also, humans use commonsense knowledge to understand the often complex and ambiguous meaning of natural language sentences. Humans interpret a passage as a sequence of sentences and will normally process the events in the story in the same order as the sentences. Once humans understand the meaning of a passage, they can answer questions posed, along with an explanation for the answer. Similarly, for visual question answering, an image should be represented in human's mind, then it is able to answer natural language questions by understanding the intent. Moreover, by using commonsense, a human assistant understands the user's intended task and asks questions to the user about the required information to successfully carry-out the task. Also, to hold a goal oriented conversation, a human remembers all the details given in the past and most of the time performs non-monotonic reasoning to accomplish the assigned task. We believe that an automated QA system or a goal oriented closed domain chatbot should work in a similar way.

If we want to build AI systems that emulate humans, then understanding natural language sentences is the foremost priority for any NLU application. In an ideal scenario, an NLU application should map the sentence to the knowledge (semantics) it represents, augment it with commonsense knowledge related to the concepts involved–just as humans do—then use the combined knowledge to do the required reasoning. In this paper, we introduce to one of our algorithm [1] for automatically generating the semantics corresponding to each English sentence using the comprehensive verb-lexicon for English verbs - VerbNet [2]. For each English verb, VerbNet gives the syntactic and semantic patterns. The algorithm employs partial syntactic matching between parse-tree of a sentence and a verb's *frame syntax* from VerbNet to obtain the meaning of the sentence in terms of VerbNet's primitive predicates. This matching is motivated by denotational semantics of programming languages and can be thought of as mapping parse-trees of sentences to knowledge that is constructed out of semantics provided by VerbNet. The VerbNet semantics is expressed using a set of primitive predicates that can be thought of as the *semantic algebra* of the denotational semantics.

Answering questions about a given picture, or Visual Question Answering (VQA) can be processed similar to the textual QA. To answer questions about a picture, humans generally first recognize the objects in the picture, then they reason with the questions asked using their commonsense knowledge. To be effective, we believe

ICLP'21: International Conference on Logic Programming, September, 2021
Kinjal.Basu@utdallas.edu (K. Basu); gupta@utdallas.edu (G. Gupta)

a VQA system should work in a similar way. Thus, to perceive a picture, ideally, a system should have intuitive abilities like object and attribute recognition and understanding of spatial-relationships. To answer questions, it must use reasoning. Natural language questions are complex and ambiguous by nature, and also require commonsense knowledge for their interpretation. Most importantly, reasoning skills such as counting, inference, comparison, etc., are needed to answer these questions. Here, we present out VQA work — AQuA (ASP-based Visual Question Answering), that closely simulates the above described way of an ideal VQA [3].

2. Background

Answer Set Programming (ASP): An answer set program is a collection of rules of the form -

$$l_0 \leftarrow l_1,\ \ldots,\ l_m,\ not\, l_{m+1},\ \ldots,\ not\, l_n.$$

Classical logic denotes each l_i is a literal [4]. In an ASP rule, the left hand side is called the *head* and the right-hand side is the *body*. Constraints are ASP rules without *head*, whereas facts are without *body*. The variables start with an uppercase letter, while the predicates and the constants begin with a lowercase. We will follow this convention throughout the paper. The semantics of ASP is based on the stable model semantics of logic programming [5]. ASP supports *negation as failure* [4], allowing it to elegantly model common sense reasoning, default rules with exceptions, etc., and serves as the secret sauce for AQuA's sophistication.

s(CASP) System: s(CASP) [6] is a query-driven, goal-directed implementation of ASP that includes constraint solving over reals. Goal-directed execution of s(CASP) is indispensable for automating commonsense reasoning, as traditional grounding and SAT-solver based implementations of ASP may not be scalable. There are three major advantages of using the s(CASP) system: (i) s(CASP) does not ground the program, which makes our framework scalable, (ii) it only explores the parts of the knowledge base that are needed to answer a query, and (iii) it provides natural language justification (proof tree) for an answer [7].

Denotational Semantics: In programming language research, denotational semantics is a widely used approach to formalize the meaning of a programming language in terms of mathematical objects (called *domains*, such as integers, truth-values, tuple of values, and, mathematical functions) [8]. Denotational semantics of a programming language has three components [8]:

1. **Syntax**: specified as abstract syntax trees.

NP V NP	
Example	***"She grabbed the rail"***
Syntax	**Agent V Theme**
Semantics	**Continue(E,Theme)**,**Cause(Agent,E)** **Contact(During(E),Agent,Theme)**

Figure 1: VerbNet frame instance for the verb class *grab*

2. **Semantic Algebra**: these are the basic domains along with the associated operations; meaning of a program is expressed in terms of these basic operations applied to the elements in the domain.

3. **Valuation Function**: these are mappings from abstract syntax trees (and possibly the semantic algebra) to values in the semantic algebra.

Given a program **P** written in language **L**, **P**'s denotation (meaning), expressed in terms of the semantic algebra, is obtained by applying the valuation function of **L** to program **P**'s syntax tree. Details can be found elsewhere [8].

VerbNet: Inspired by Beth Levin's classification of verbs and their syntactic alternations [9], VerbNet [2] is the largest online network of English verbs. A verb class in VerbNet is mainly expressed by *syntactic frames*, *thematic roles*, and *semantic representation*. The VerbNet lexicon identifies thematic roles and syntactic patterns of each verb class and infers the common syntactic structure and semantic relations for all the member verbs. Figure 1 shows an example of a VerbNet frame of the verb class *grab*.

3. Commonsense Reasoning with Default Theories

As mentioned earlier, a realistic socialbot should be able to understand and reason like a human. In human to human conversations, we do not always tell every detail, we expect the listener to fill gaps through their commonsense knowledge and commonsense reasoning. Thus, to obtain a conversational bot, we need to automate commonsense reasoning, i.e., automate the human thought process. The human thought process is flexible and *non-monotonic* in nature, which means *"what we believe today may become false in the future with new knowledge"*. We can model commonsense reasoning with (i) default rules, (ii) exceptions to defaults, (iii) preferences over multiple defaults [5], and (iv) modeling *multiple worlds* [4, 10].

Much of human knowledge consists of default rules, for example, the rule: *Normally, birds fly.* However, there are exceptions to defaults, for example, *penguins are exceptional birds that do not fly.* Reasoning with default rules is non-monotonic, as a conclusion drawn using a default rule may have to be withdrawn if more knowledge becomes available and the exceptional case applies. For example, if we are told that Tweety is a bird, we will conclude it flies. Later, knowing that Tweety is a penguin will cause us to withdraw our earlier conclusion.

Humans often make inferences in the absence of complete information. Such an inference may be revised later as more information becomes available. This human-style reasoning is elegantly captured by default rules and exceptions. Preferences are needed when there are multiple default rules, in which case additional information gleaned from the context is used to resolve which rule is applicable. One could argue that expert knowledge amounts to learning defaults, exceptions and preferences in the field that a person is an expert in.

Also, humans can naturally deal with *multiple worlds*. These worlds may be consistent with each other in some parts, but inconsistent in other parts. For example, animals don't talk like humans in the real world, however, in the cartoon world, animals do talk like humans. So, a fish called Nemo, may be able to swim in both the real world and the cartoon world, but can only talk in the cartoon world. Humans have no trouble separating cartoon world from real world and switching between the two as the situation demands. Default reasoning augmented with the ability to operate in multiple worlds, allows one to closely represent the human thought process. Default rules with exceptions and preferences and multiple worlds can be elegantly realized with answer set programming [4, 10] and the s(CASP) system [6].

4. Visual Question Answering

Our work — AQuA (ASP-based Question Answering) is an Answer Set Programming (ASP) based visual question answering framework that truly "understands" an input picture and answers natural language questions about that picture [3]. This framework achieves 93.7% accuracy on CLEVR dataset, which exceeds human baseline performance. What is significant is that AQuA translates a question into an ASP query without requiring any training. AQuA replicates a human's VQA behavior by incorporating commonsense knowledge and using ASP for reasoning. VQA in the AQuA framework employs the following sources of knowledge: (i) knowledge about objects extracted using the YOLO algorithm [11], (ii) semantic relations extracted from the question, (iii) query generated from the question, and (iv) commonsense knowledge. AQuA runs on the query-driven, scalable s(CASP) [6] answer set programming system that can provide a proof tree as a justification for the query being processed.

AQuA processes and reasons over raw textual questions and does not need any annotation or generation of function units such as what is employed by several approaches proposed for the CLEVR dataset [12, 13, 14]. Also, instead of predicting an answer, AQuA augments the parsed question with commonsense knowledge to truly understand it and to compute the correct answer (e.g., it understands that *block* means *cube*, or *shiny object* means *metal object*).

4.1. Technical Approach

AQuA represents knowledge using ASP paradigm and it is made up of five modules that perform the following tasks: (i) object detection and feature extraction using the YOLO algorithm [11], (ii) preprocessing of the natural language question, (iii) semantic relation extraction from the question, (iv) Query generation based on semantic analysis, and (v) commonsense knowledge representation. AQuA runs on the query-driven, scalable s(CASP) [6] answer set programming system that can provide a proof tree as a justification for a query being processed. Figure 2 shows AQuA's architecture. The five modules are labeled, respectively, YOLO, Preprocessor, Semantic Relation Extractor (SRE), Query Generator, and Commonsense Knowledge.

Preprocessor module extracts information from the question by using Stanford CoreNLP parts-of-speech (POS) tagger and dependency graph generator. The output of the Preprocessing module will be consumed by the Query Generator and the Semantic Relation Extraction (SRE) modules. AQuA transforms natural language questions to a logical representation before feeding it to the ASP engine. The logical representation module is inspired by Neo-Davidsonian formalism [15], where every event is recognized with a unique identifier. Next, the *semantic relation labeling* is the process of assigning relationship labels to two different phrases in a sentence

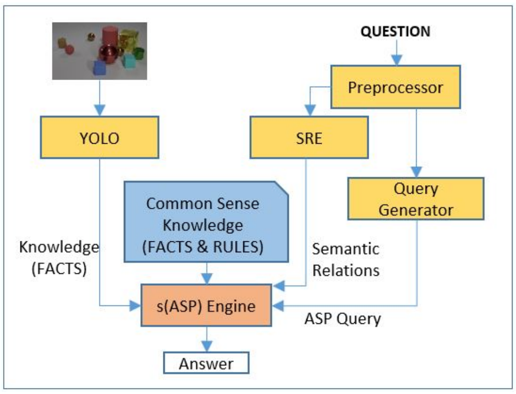

Figure 2: AQuA System Architecture

Question Type		*Accuracy (%)*	
Exist		**96**	
Count		**91.7**	
Compare Value	Shape	87.42	**92.89**
	Color	94.32	
	Size	92.17	
	Material	96.14	
Compare Integer	Less Than	97.7	**98.05**
	Greater Than	98.6	
	Equal	NA*	
Query Attribute	Shape	94.01	**94.39**
	Color	94.87	
	Size	93.82	
	Material	94.75	

Table 1
AQuA Performance Results
* Equality questions are minuscule in number so currently ignored.

based on the context. To understand the CLEVR dataset questions, AQuA requires two types of semantic relations (i.e., *quantification* and *property*) to be extracted (if they exists) from the questions. Based on the knowledge from a question, AQuA generates a list of ASP clauses with the query, which runs on the s(CASP) engine to find the answer. In general, questions with one-word answer are categorized into: (i) yes/no questions, and (ii) attribute/value questions. Similar to a human, AQuA requires commonsense knowledge to correctly compute answers to questions. For the CLEVR dataset questions, AQuA needs to have commonsense knowledge about properties (e.g., color, size, material), directions (e.g., left, front), and shapes (e.g., cube, sphere). AQuA will not be able to understand question phrases such as *'... red metal cube ...'*, unless it knows *red* is a color, *metal* is a material, and *cube* is a shape. Finally, the ASP engine is the brain of our system. All the knowledge (image representation,commonsense knowledge, semantic relations) and the query in ASP syntax are executed using the query-driven s(CASP) system

4.2. Experiments and Results

We tested our AQuA framework on the CLEVR dataset [16] and we got accuracy of **93.7%** with 42,314 correct answers out of 45,157 questions. This performance is beyond the average human accuracy. Quantitative results for each question type are summarized in Table 1.

We have extensively studied the 2,843 questions that produced erroneous results. Our manual analysis showed that mismatch happens mostly because of errors caused by the YOLO module: failing to detect a partially visible object, wrongly detecting a shadow as an object, wrongly detecting two overlapping objects as one, etc. Other reasons for wrong answers are wrong parsing or oversimplified spatial reasoning.

5. Textual Question Answering

Unlike programming languages, the denotation of a natural language can be quite ambiguous. English is no exception and the meaning of a word or sentence may depend on the context. The generation of correct knowledge from a sentence, hence, is quite hard. We have developed a VerbNet based algorithm for semantic generation of English text. In this section, we present a novel approach to automatically map parse trees of simple English sentences to their denotations, i.e., knowledge they represent [17]. We applied this approach to construct two NLU applications that we present here: SQuARE (Semantic-based Question Answering and Reasoning Engine) and StaCACK (Stateful Conversational Agent using Commonsense Knowledge).

5.1. Semantics-driven ASP Code Generation

Similar to the denotational approach for meaning representation of a programming language, an ideal NLU system should use denotational semantics to compositionally map text syntax to its meaning. Knowledge primitives should be represented using the *semantic algebra* [8] of well understood concepts. Then the semantics along with the commonsense knowledge represented using the same semantic algebra can be used to construct different NLU applications, such as QA system, chatbot, information extraction system, text summarization, etc. The ambiguous nature of natural language is the main hurdle in treating it as a programming language. English is no exception and the meaning of an English word or sentence may depend on the context. The algorithm we present takes the syntactic parse tree of an English sentence and uses VerbNet to automatically map the parse tree to its denotation, i.e., the knowledge it represents.

An English sentence that consists of an action verb (i.e., not a *be* verb) always describes an event. The verb also constrains the relation among the event participants. VerbNet encapsulates all of this information using verb classes that represent a verb set with similar meanings. So each verb is a part of one or more classes. For each class, it provides the skeletal parse tree (frame syntax) for different usage of the verb class and the respective semantics (frame semantic). The semantic definition of each frame uses pre-defined predicates of VerbNet that have thematic-roles (AGENT, THEME, etc.) as arguments. Thus, we can imagine VerbNet as a very large valuation (semantic) function that maps syntax tree patterns to their respective meanings. As we use ASP to represent the knowledge, the algorithm generates the sentence's

Algorithm 1 Semantic Knowledge Generation

Input: p_t: constituency parse tree of a sentence
Output: *semantics*: sentence semantics

procedure GetSentenceSemantics(p_t)
  verbs ← *getVerbs(p_t)* ▷ returns list of verbs present in the sentence
  semantics ← *{}* ▷ initialization
  for each v ∈ *verbs* **do**
    classes ← *getVNClasses(v)* ▷ get the VerbNet classes of the verb
    for each c ∈ *classes* **do**
      frames ← *getVNFrames(c)* ▷ get the VerbNet frames of the class
      for each f ∈ *frames* **do**
        thematicRoles ← *getThematicRoles(p_t, f.syntax, v)* ▷ see Algorithm 2
        semantics ← *semantics* ∪ *getSemantics(thematicRoles, f.semantics)*
        ▷ map the thematic roles into the frame semantics
      end for
    end for
  end for
  return semantics
end procedure

Algorithm 2 Partial Tree Matching

Input: p_t: constituency parse tree of a sentence; *s*: frame syntax; *v*: verb
Output: *tr*: thematic role set or *empty-set*: {}

procedure GetThematicRoles(p_t, s, v)
  root ← *getSubTree(node(v), p_t)* ▷ returns the sub-tree from the parent of the verb node
  while root **do**
    $tr \leftarrow getMatching(root, s)$ ▷ if s matches the tree return thematic-roles, else {}
    if $tr \neq$ *{}* **then return** tr
    end if
    root ← *getSubTree(root, p_t)* ▷ returns *false* if root equals p_t
  end while
  return {}
end procedure

semantic definition in ASP. *Our goal is to find the partial matching between the sentence parse tree and the VerbNet frame syntax and ground the thematic-role variables so that we can get the semantics of the sentence from the frame semantics and represent it in ASP.*

The illustration of the process of semantic knowledge generation from a sentence is described in the Figure 3. We have used Stanford's CoreNLP parser [18] to generate the parse tree, p_t, of an English sentence. The semantic generator component consists of the valuation function to map the p_t to its meaning. To accomplish this, we have introduced Semantic Knowledge Generation algorithm (Algorithm 1). First, the algorithm collects the list of verbs mentioned in the sentence and for each verb it accumulates all the syntactic (frame syntax) and corresponding semantic information (thematic roles and predicates) from VerbNet using the verb-class of the verb. The algorithm finds the grounded thematic-role variables by doing a partial tree matching (described in Algorithm 2) between each gathered frame syntax and p_t. From the verb node of p_t, the partial tree matching algorithm performs a bottom-up search and, at each level through a depth-first traversal, it tries to match the skeletal parse tree of the frame syntax. If the algorithm finds an exact or a partial match (by skipping words, e.g., prepositions), it returns the thematic roles to the parent Algorithm 1. Finally, Algorithm 1 grounds the pre-defined predicate with the values of thematic roles and generates ASP code.

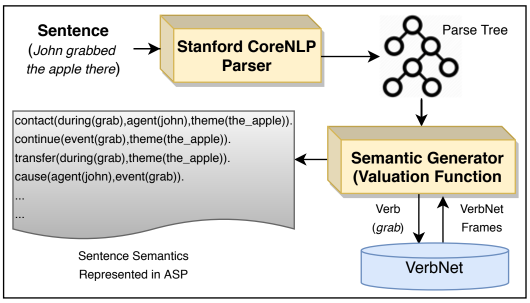

Figure 3: English to ASP translation process

The ASP code generated by the above mentioned approach represents the meaning of a sentence comprised of an action verb. Since VerbNet does not cover the semantics of the *'be'* verbs (i.e., *am, is, are, have,* etc.), for sentences containing *'be'* verbs, the *semantic generator* uses pre-defined handcrafted mapping of the parsed information (i.e., syntactic parse tree, dependency graph, etc.) to its semantics. Also, this semantics is represented as ASP code. The generated ASP code can now be used in various applications, such as natural language QA, summarization, information extraction, Conversational Agents (CA), etc.

5.2. SQuARE

Question answering system for reading comprehension is a challenging task for the NLU research community. In recent times with the advancement of ML applied to NLU, researchers have created more advanced QA systems that show outstanding performance in QA for reading-comprehension tasks. However, for these high performing neural-networks based agents, the question rises whether they really *"understand"* the text or not. These systems are outstanding in learning data patterns and then predicting the answers that require shallow

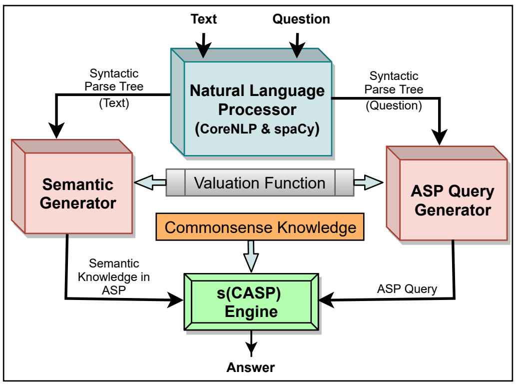

Figure 4: SQuARE Framework

or no reasoning capabilities. Moreover, for some QA tasks, if a system claims that it performs equal or better than a human in terms of accuracy, then the system must also show human level intelligence in explaining its answers. Taking all this into account, we have created our SQuARE QA system that uses ML based parser to generate the syntax tree and uses Algorithm 1 to translate a sentence into its knowledge in ASP. By using the ASP-coded knowledge along with pre-defined generic commonsense knowledge, SQuARE outperforms other ML based systems by achieving 100% accuracy in 18 tasks (99.9% accuracy in all 20 tasks) of the bAbI QA dataset (note that the 0.01% inaccuracy is due to the dataset's flaw, not of our system). SQuARE is also capable of generating English justification of its answers.

SQuARE is composed of two main sub systems: the *semantic generator* and the *ASP query generator*. Both subsystems inside the SQuARE architecture (illustrated in Figure 4) share the common valuation function.

Example: To demonstrate the power of the SQuARE system, we next discuss a full-fledged example showing the data-flow and the intermediate results.

Story: A customized segment of a story from the bAbI QA dataset about *counting objects* (Task-7) is taken.

```
1 John moved to the bedroom.
2 John got the football there.
3 John grabbed the apple there.
4 John picked up the milk there.
5 John gave the apple to Mary.
6 John left the football.
```

Parsed Output: CoreNLP and spaCy parsers parse each sentence of the story and pass the parsed information to the semantic generator. Details are omitted due to lack of space, however, parsing can be easily done at `https://corenlp.run/`.

Semantics: From the parsed information, the *semantic generator* generates the semantic knowledge in ASP. We only give a snippet of knowledge (due to space constraint) generated from the third sentence of the story (VerbNet details of the verb - *grab* is given in Figure 1).

```
1 contact(t3,during(grab),agent(john),
              theme(the_apple)).
2 cause(t3,agent(john),event(grab)).
3 transfer(t3,during(grab),
              theme(the_apple)).
```

Question and ASP Query: For the question - *"How many objects is John carrying?"*, the *ASP query generator* generates a generic query-rule and the specific ASP query (it uses the process template for counting).

```
count_object(T,Per,Count) :-
 findall(O,property(possession,T,Per,O),Os),
 set(Os,Objects),list_length(Objects,Count).
?- count_object(t6,john,Count).
```

Answer: The s(CASP) system finds the correct answer - *1.*

Justification: The s(CASP) system generated justification for this answer is shown in Figure 5.

```
The total count of all the objects that john is possessing at time t6 is 1, because
 [the_milk] is the list of all the objects that are possessed by john at time t6,
                                                                         because
   the_milk is possessed by john at time t6, because
     time t6 comes after time t5, and
     the_milk is possessed by john at time t5, because
       time t5 comes after time t4, and
       the_milk is possessed by john at time t4, and
       there is no evidence that the_milk is not possessed by john at time t5.
     there is no evidence that the_milk is not possessed by john at time t6.
 The list [the_milk] is generated after removing duplicates from the list [the_milk],
                                                                         because
   The list [] is generated after removing duplicates from the list [].
 1 is the length of the list [the_milk], because
   0 is the length of the list [].
```

Figure 5: Natural language justification

5.3. StaCACK

Conversational AI has been an active area of research, starting from a rule-based system, such as ELIZA [19] and PARRY [20], to the recent open domain, data-driven CAs like Amazon's Alexa, Google Assistant, or Apple's Siri. Early rule-based bots were based on just syntax analysis, while the main challenge of modern ML based chat-bots is the lack of "understanding" of the conversation. A realistic socialbot should be able to understand and reason like a human. In human to human conversations, we do not always tell every detail, we expect the listener to fill gaps through their commonsense knowledge. Also, our thinking process is flexible and non-monotonic in nature, which means *"what we believe today may become false in the future with new knowledge"*. We can model this human thinking process with (i) default rules, (ii) exceptions to defaults, and (iii) preferences over multiple defaults [4].

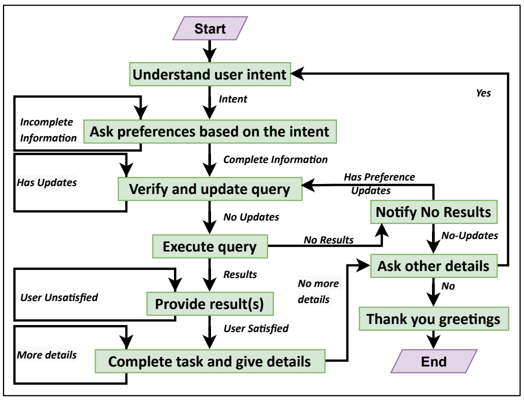

Figure 6: FSM for StaCACK framework

Following the discussion above, we have created StaCACK, a general closed-domain chatbot framework. StaCACK is a stateful framework that maintains states by remembering every past dialog between the user and itself. The main difference between StaCACK and the other stateful or stateless chatbot models is the use of commonsense knowledge for understanding user utterances and generating responses. Moreover, it is capable of doing non-monotonic reasoning by using defaults with exceptions and preferences in ASP.

Model / Tasks	MemNN (AM+ NG+ NL)	Mitra et al.	SQuARE
Single Supporting Fact	100	100	100
Two Supporting Facts	98	100	100
Three Supporting Facts	95	100	100
Two Arg. Relation	100	100	100
Three Arg. Relation	99	100	99.8
Yes/No Questions	100	100	100
Counting	97	100	100
Lists/Sets	97	100	100
Simple Negation	100	100	100
Indefinite Knowledge	98	100	98.2
Basic Coreference	100	100	100
Conjunction	100	100	100
Compound Coreference	100	100	100
Time Reasoning	100	100	100
Basic Deduction	100	100	100
Basic Induction	99	93.6	100
Positional Reasoning	60	100	100
Size Reasoning	95	100	100
Path Finding	35	100	100
Agent's Motivations	100	100	100
MEAN ACCURACY	**94**	**100**	**100**

Table 2
SQuARE accuracy (%) comparison

Task-specific CAs follow a certain scheme in their inquiry that can be modeled as a finite state machine (FSM). The FSM is illustrated in Figure 6. However, the tasks in each state transition are not simple as in every level it requires different types of (commonsense) reasoning.

StaCACK achieves 100% accuracy on the Facebook bAbI dialog dataset suit [21] (including OOV: out-of-vocabulary datasets) of five tasks created for a restaurant reservation dialog system. In addition, StaCACK can answer questions that ML chatbots cannot without proper training (details are given in following sections). We focus on agents that are designed for a specific tasks (e.g., restaurant reservation).

Example: StaCACK is able to hold the conversation in a more natural way by using commonsense knowledge, which may not be possible with a rule-based system based on a monotonic logic. Following example shows how StaCACK can understand the cuisine preference of a user, just by performing reasoning over commonsense information about a cuisine (that *curry* is predominant in *Indian* and *Thai* cuisine).

```
User:    Good morning.
StaCACK: Hello what can i help you with
         today?
User:    Can you make a restaurant
         reservation in London in a cheap
         price range?
StaCACK: I'm on it.
User:    <SILENCE>
StaCACK: Any preference on a type of
         cuisine?
User:    Anything, except Lebanese food.
StaCACK: Do you want to have Chinese
         food?
User:    I want to have curry.
StaCACK: Do you like Indian or Thai?
User:    Thai.
```

5.4. Experiments and Results

The SQuARE and the StaCACK system have been tested on the bAbI QA [22] and the bAbI dialog dataset respectively [21]. With the aim of improving NLU research, Facebook researchers have created the bAbI datasets suit comprising of different NLU application-oriented simple task-based datasets. The datasets are designed in such a way that it becomes easy for human to reason and reach an answer with proper justification whereas difficult for machines due to the lack of understanding about the language. In the SQuARE system, the accuracy has been calculated by matching the generated answer with the actual answer given in the bAbI QA dataset. Whereas, StaCACK's accuracy is calculated on the basis of *per-response* as well as *per-dialog*. Table 2 and table 3 compares our results in terms of accuracy with the ex-

	Mem2Seq	BossNet	StaCACK
Task 1	100 (100)	100 (100)	100 (100)
Task 2	100 (100)	100 (100)	100 (100)
Task 3	94.7 (62.1)	95.2 (63.8)	100 (100)
Task 4	100 (100)	100 (100)	100 (100)
Task 5	97.9 (69.6)	97.3 (65.6)	100 (100)
Task 1 (OOV)	94.0 (62.2)	100 (100)	100 (100)
Task 2 (OOV)	86.5 (12.4)	100 (100)	100 (100)
Task 3 (OOV)	90.3 (38.7)	95.7 (66.6)	100 (100)
Task 4 (OOV)	100 (100)	100 (100)	100 (100)
Task 5 (OOV)	84.5 (2.3)	91.7 (18.5)	100 (100)

Table 3
StaCACK accuracy per response (per dialog) in %.

isting state-of-the-art results for SQuARE and StaCACK system respectively.

6. Social-Bot

Using the similar technology of the StaCACK system, We have designed and developed the CASPR system, a socialbot designed to compete in the Amazon Alexa Socialbot Challenge 4. CASPR's distinguishing characteristic is that it will use automated commonsense reasoning to truly *"understand"* dialogs, allowing it to converse like a human. Three main requirements of a socialbot are that it should be able to "understand" users' utterances, possess a strategy for holding a conversation, and be able to learn new knowledge. We developed techniques such as conversational knowledge template (CKT) to approximate commonsense reasoning needed to hold a conversation on specific topics.

Our philosophy is to design a socialbot that emulates, as much as possible, the way humans conduct social conversations. Humans employ both learned-pattern matching (e.g., recognizing user sentiments) and commonsense reasoning (e.g., if a user starts talking about having seen the Eiffel Tower, we infer that they must have traveled to France in the past) during a conversation. Thus, ideally, a socialbot should make use of both machine learning as well as commonsense reasoning technologies. Our goal is to use the appropriate technology for a task, i.e., use machine learning and commonsense reasoning for respective tasks that they are good at. Machine learning is good for tasks such as parsing, topic modeling, and sentiment detection while commonsense reasoning is good for tasks such as generating a response to an utterance. In a nutshell, we should use machine learning for modeling *System 1* thinking and commonsense reasoning for modeling *System 2* thinking [23]. We strongly believe that intelligent systems that emulate human ability should follow this approach, especially, if we desire true understanding and explainability.

CASPR's conversation planning is centered around a loop in which it moves from topic to topic, and within a topic, it moves from one attribute of that topic to another. Thus, CASPR has an *outer conversation loop* to hold the conversation at the topmost level and an inner loop in which it moves from attribute to attribute of a topic. The logic of these loops is slightly involved, as a user may return to a topic or an attribute at any time, and CASPR must remember where the user left off in that topic or attribute. For the inner loops, CASPR uses a template, called conversational knowledge template (CKT), that can be used to automatically generate code that loops over the attributes of a topic, or loops through various dialogs (mini-CKT) that need to be spoken by CASPR for a given topic.

7. Conclusion and Future Work

In this paper, we discussed about our ASP based approaches to overcome the challenges of NLU. In the process of that we presented a visual question answering framework — AQuA. In the textual QA domain, we introduced to our novel semantics-driven English text to answer set program generator. Also, we showed how commonsense reasoning coded in ASP can be leveraged to develop advanced NLU applications, such as SQuARE and StaCACK. We make use of the s(CASP) engine, a query-driven implementation of ASP, to perform reasoning while generating a natural language explanation for any computed answer. At the end, we discussed about the design philosophy behind our social-bot CASPR and how we have qualified to participate in the Amazon Alexa Socialbot Challenge 4. As part of future work, we plan to extend the SQuARE system to handle more complex sentences and eventually handle complex stories. Our goal is also to develop an open-domain conversational AI chatbot based on automated commonsense reasoning that can "converse" with a human based on "truly understanding" that person's utterances.

References

[1] K. Basu, S. C. Varanasi, F. Shakerin, G. Gupta, Square: Semantics-based question answering and reasoning engine, arXiv preprint arXiv:2009.10239 (2020).

[2] K. Kipper, A. Korhonen, N. Ryant, M. Palmer, A large-scale classification of english verbs, Language Resources and Evaluation 42 (2008) 21–40. doi:10.1007/s10579-007-9048-2.

[3] K. Basu, F. Shakerin, G. Gupta, Aqua: Asp-based visual question answering, in: International Symposium on Practical Aspects of Declarative Languages, Springer, 2020, pp. 57–72.

[4] M. Gelfond, Y. Kahl, Knowledge representation, reasoning, and the design of intelligent agents: The answer-set programming approach, Cambridge University Press, 2014.

[5] M. Gelfond, V. Lifschitz, The stable model semantics for logic programming., in: ICLP/SLP, volume 88, 1988, pp. 1070–1080.

[6] J. Arias, M. Carro, E. Salazar, K. Marple, G. Gupta, Constraint answer set programming without grounding, TPLP 18 (2018) 337–354. doi:10.1017/S1471068418000285.

[7] J. Arias, M. Carro, Z. Chen, G. Gupta, Justifications for goal-directed constraint answer set programming, arXiv preprint arXiv:2009.10238 (2020).

[8] D. A. Schmidt, Denotational semantics: a methodology for language development, William C, Brown Publishers, Dubuque, IA, USA, 1986.

[9] B. Levin, English verb classes and alternations: A preliminary investigation, U. Chicago Press, 1993. doi:10.1075/fol.2.1.16noe.

[10] C. Baral, Knowledge representation, reasoning and declarative problem solving, Cambridge Uni. Press, 2003.

[11] J. Redmon, A. Farhadi, Yolov3: An incremental improvement, arXiv preprint arXiv:1804.02767 (2018).

[12] J. Johnson, et al., Inferring and executing programs for visual reasoning, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2989–2998.

[13] J. Suarez, J. Johnson, F.-F. Li, Ddrprog: A clevr differentiable dynamic reasoning programmer, arXiv preprint arXiv:1803.11361 (2018).

[14] K. Yi, et al., Neural-symbolic VQA: Disentangling reasoning from vision and language understanding, in: NIPS'18, 2018, pp. 1031–1042.

[15] D. Davidson, Inquiries into truth and interpretation: Philosophical essays, volume 2, Oxford University Press, 2001.

[16] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. Lawrence Zitnick, R. Girshick, Clevr: A diagnostic dataset for compositional language and elementary visual reasoning, in: IEEE CVPR'17, 2017, pp. 2901–2910.

[17] K. Basu, S. Varanasi, F. Shakerin, J. Arias, G. Gupta, Knowledge-driven natural language understanding of english text and its applications, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, 2021, pp. 12554–12563.

[18] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, D. McClosky, The Stanford CoreNLP NLP toolkit, in: ACL System Demonstrations, 2014, pp. 55–60. doi:10.3115/v1/P14-5010.

[19] J. Weizenbaum, ELIZA—a computer program for the study of natural language communication between man and machine, CACM 9 (1966) 36–45.

[20] K. M. Colby, S. Weber, F. D. Hilf, Artificial paranoia, Artificial Intelligence 2 (1971) 1–25.

[21] A. Bordes, Y.-L. Boureau, J. Weston, Learning end-to-end goal-oriented dialog, arXiv preprint arXiv:1605.07683 (2016).

[22] J. Weston, et al., Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks, arXiv preprint arXiv:1502.05698 (2015).

[23] D. Kahneman, Thinking, fast and slow, Macmillan, 2011.