SEVA: A Systems Engineer's Virtual Assistant

Jitin Krishnan
Department of Computer Science
George Mason University
jkrishn2@gmu.edu

Patrick Coronado
Instrument Development Center
NASA Goddard Space Flight Center
patrick.l.coronado@nasa.gov

Trevor Reed
Robot Operations
NASA Jet Propulsion Laboratory
trevor.reed@jpl.nasa.gov

Abstract

A Systems Engineer's Virtual Assistant (SEVA) is a novel attempt to bridge the gap between Natural Language Processing (NLP), Knowledge Base (KB) Construction research, and NASA's Systems Engineering domain. In this work, we propose the design of an explainable, human-in-the-loop, and interactive personal assistant system. The assistant will help a Systems Engineer in their daily work environment through complex information management and high-level question-answering to augment their problem-solving abilities. We describe the fundamental characteristics of the assistant by understanding operational, functional, and system requirements from Systems Engineers and NASA's Systems Engineering Handbook. The assistant is designed to act as a workbench to manage dynamic information about projects and analyze hypothetical scenarios. It is also designed to make logical inferences and perform temporal reasoning by handling domain information and information related to schedule and resources. In addition, the system learns new information over time by interacting with its user and can perform case-based reasoning from previous experiences. The knowledge base design describes a novel hybrid approach to build a domain-independent common-sense framework with which domain-specific engineers can attune it and build their projects. Using these specific objectives and constraints, the architecture of a personal assistant is proposed. Main contributions of this design paper are Systems Engineering (SE) domain analysis, a survey of existing research, preliminary experiments using the state-of-the-art systems to explore the feasibility, a proposal of a complete architecture with component level detail, and identification of areas that require further research and development.

Keywords: Natural Language Processing, Knowledge Base Construction, Question-Answering, Intelligent Agents, Explainable AI, Systems Engineering, Ontology

INTRODUCTION

Intelligent personal assistants have drawn attention in recent years due to their ubiquitous nature of making human lives easier. However, practical use of such personal assistants by individual scientists or engineers, such as those at NASA, is still very limited due to the risk-averse nature of engineering projects, users' steep learning curves in studying the unnecessary details of a new system, and the assistant's inability to capture the relevant complexity of the domain. In addition, explainability of answers plays a key role in such AI systems where trust in the answers is paramount. It is a dream for most scientists and engineers to have a personal assistant who takes care of the tedious book-keeping aspects and assists them in creative problem solving. Systems Engineers (SE) deal with large amounts of information in their everyday work. Their role in technical project planning requires handling this information and keeping track of diverse requirements, changing variables, resources, and schedules. This extensive information assimilation is both tedious and error-prone. The ability of SEs could be greatly enhanced by a system that could handle such tasks. SEVA is being developed with this goal in mind: to assist SEs and enhance their problem-solving abilities by keeping track of the large amounts of information of a NASA specific project and using the information to answer queries from the user.

Copyright held by the author(s). In A. Martin, K. Hinkelmann, A. Gerber, D. Lenat, F. van Harmelen, P. Clark (Eds.), Proceedings of the AAAI 2019 Spring Symposium on Combining Machine Learning with Knowledge Engineering (AAAI-MAKE 2019). Stanford University, Palo Alto, California, USA, March 25-27, 2019.

BACKGROUND AND MOTIVATION

NASA's state-of-the-art approach to computer-aided Systems Engineering projects is Model Based Systems Engineering (MBSE). It is a model-centric approach which supports all phases of SDLC (System Development Life Cycle) with which SEs can design their large scale projects. Two key problems that MBSE tackle are collaboration of different domains and conversion of documents to digital models. MBSE also has a similar goal in mind as SEVA: to facilitate understanding, aid decision-making, examine hypothetical scenarios, and explain, control, and predict events (Hart 2015). MBSE is expected to solve the problems of inadequacy in information-capture and dynamic nature of SE's design documentations such as System Block Diagrams, Electrical Interconnect List, and Mass Equipment List (Fosse et al. 2014). MBSE aims to tackle the problems of lack of a common language between different disciplines using a formal language called SysML. However, systems such as MBSE present a steep learning curve for new learners who will need to learn the system or a new language.

On the other end of the learning curve are popular intelli-

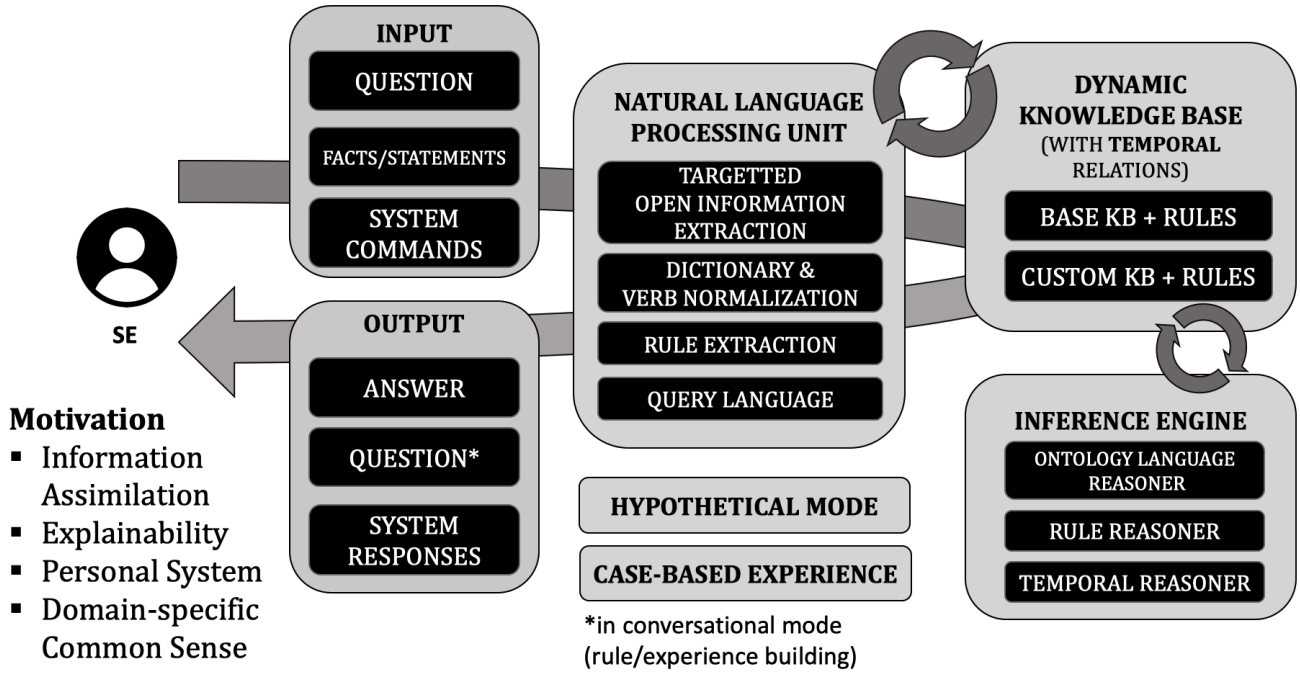

Figure 1: Architectural Components

gent assistants such as Siri providing proactive assistance on a very specific set of contextual tasks such as calling a friend, sending a message, or making a dinner reservation. These assistants are not knowledge systems which can assist a user in complex tasks such as engineering projects or disease diagnosis. There is little reason for such assistants to maintain large scale ontologies. For example, the rather small and active ontology maintained by Siri is sufficient enough to aid its task-based design (Gruber 2009). On the other hand, large scale knowledge systems such as IBM Watson perform statistical inferencing to answer a question by ingesting millions of documents. The main goal of Watson is to essentially answer a question. Watson is trained by QA sessions and answers are returned with probability or confidence (Lee et al. 2016). Watson's knowledge is a compilation of knowledge from various domain experts. Thousands of medical books can be ingested by Watson to create a large scale knowledge system for disease diagnosis. Since Watson used evidence gathering and scoring algorithms, ontologies were not essential in the architecture (Ferrucci et al. 2010; 2013). On the other hand, SEVA's domain is limited and needs a knowledge base or an ontology which is complete and can handle changing contextual facts. In addition, the ability to say *'I don't know'* is an essential characteristic of an explainable system.

The Cyc project tried to perform human-like reasoning by building a comprehensive system with common-sense knowledge such as *"water makes things wet"*, *"plants die eventually"*, etc (Panton et al. 2006). The idea was to create a system that can serve as a foundation to all future expert systems. However, millions of assertions are hand-coded to the system and the amount of knowledge required to be learned was one of the major criticisms of the project (Domingos 2015). In our work, we aim to construct a scalable domain specific common-sense knowledge base with SE facts.

The MIT Programmer's Apprentice Project (Rich and Waters 1987; Brooks 1997), which ran in 1970s and 80s, shares a similar spirit with SEVA. The goal was to study knowledge representation formalism and reasoning in the context of programming. The work realized the importance of personal assistants performing mundane tasks and incremental development of knowledge (Rich and Waters 1987) which is also relevant in SEVA. Our work combines a generalist (SE) with a specialist (SEVA) such that the specialist's knowledge base acts as a dynamic workbench for the generalist.

OPERATIONAL CONCEPT

SEVA has 3 main categories of questions from which its external interface requirements are based from: 1) **Relational**, 2) **Recall**, and 3) **Hypothetical**. *Relational* questions determine the existence of links between entities. In ontological terminology, links are verbs/predicates that connect subjects and objects. The answer can be 'Yes', 'No', or 'Unknown'. For example, questions such as "*Is Neon a noble gas?*" or "*Can Aerogel capture a Niacin molecule moving at 5km/s?*" fall into this category. *Recall*, the next type, determines what entity another entity is linked to. For example, questions such as "*What is the total mass of the spacecraft?*" or "*When is the vibration test for the flight Main Electronics Box?*" fall into this category. *Hypothetical* questions are "what-if" scenarios. For example, questions such as "*What will be the*

mass of Instrument X if component C is removed?".

Apart from direct recall of information and storage, four major capabilities of this virtual assistant are defined based on the various scenarios encountered by an SE: A) **perform reasoning tasks**, B) **handle temporal** (time-related) **information**, C) **answer hypothetical** (what-if) **questions**, and D) **learn from "experience"**.

Reasoning

An ontology describes concepts, properties, and relationships in formal languages such as first-order logic, description logic, or horn clauses. Using reasoning skills on its ontology, SEVA will be able to create new, indirect, or derived information that are dependent on the existing knowledge. Primary form of reasoning is related to the idea of taxonomic or *is-a* relationship between two concepts (Walton 2007). For example, assume SEVA has the following information in its knowledge base: {*Neon is a noble gas*} and {*Noble gases are odorless*}. From this information, an inference engine can deduce that {*Neon is odorless*}. SEVA's ontology consists of specific axioms from Description Logic and rules defining the complexity and explainability of the domain that is represented. The terms "reasoning engine" and "inference engine" are used interchangeably throughout this paper.

Time

SEVA handles three categories of time: 1) **time-tagging information**, 2) **processing time-related information**, and 3) **knowledge of tense**. Every information in the assistant's knowledge base has to be time-tagged. This helps the assistant in answering questions such as "*What was the mass of the nephelometer instrument three days ago?*" or "*When was the test schedule changed for the second stage cryocooler?*". The primary objective of time-tagging is to help the engineer to temporarily track the change log history of all elements of a project or task. Alternatively, processing time-related information simply means to understand temporal information in a sentence such as *06:23:43* for time, *07/11/2018* for date, and to understand intervals of time such as *'three hours ago'* or *'due in 5 hours'*. Understanding tense is an obvious capability necessary to perform the above two tasks, which means to know grammatical terms such as *is, was, ago, had, initial, etc.*

Hypothetical Mode

In hypothetical mode, an SE can ask 'what if' questions. By entering the hypothetical mode, the user can temporarily modify the information in the knowledge base. 'What-if' questions such as "*What is the Technology Readiness Level(TRL) of instrument X, if it has been to space?*" are a combination of two tasks: 1) **perform temporary updates on the knowledge base** and 2) **ask questions** as usual. In the example, the knowledge base will be temporarily updated with a new information: {*Instrument X has been to space*}. The SE then asks the question {*What is the TRL of instrument X?*} After exiting this mode, the knowledge base will be restored to its original form without the temporary updates. Hypothetical mode helps the SE in decision-making, design support, and in analyzing various scenarios or hypothetical models.

Experience

Two reasons for unsuccessful querying are 1) **the entities or concepts in the question are unknown to SEVA** and 2) **entities are known but there is not enough information to answer the question**. In the first case, SEVA asks the SE a series of questions to better understand the original question and to fill the knowledge base with necessary information to answer the question. For example, when the SE asks "*What is the mass of Neon?*" and SEVA's knowledge base has no information about 'Neon', it will respond by inquiring "*What is Neon?*". The second case is a logical problem that applies to relational questions where the ontology is missing the predicate link. Assume that the entities 'mass' and 'neon' are in SEVA's knowledge base. Suppose the SE asks "*What is the mass of instrument X?*" SEVA understands every entity in the question but has no link connecting them. There are two ways to provide the unknown information: 1) **help the system to derive the answer by creating a logical case through interaction** (eg: "*mass of an instrument is the sum of its components*") or 2) **provide the answer directly** (eg: "*mass of instrument X is 5kg*"). This type of user-assistant interaction constitutes an 'Experience' and can be used if similar cases appear in future question-answering sessions. A key aspect of SEVA's architecture is that an 'Experience' is context-driven; it is a function of questions, statements, and rules or logic from the user. The type of learning we aim to provide SEVA with is case-based contextual awareness.

Nature of Input & Querying

SEVA takes input from various sources including operations manuals and projects. In this work, we restrict to Natural Language text as input. Other types of inputs are left as future work. There are 4 types of interaction types in SEVA: **1)** information given to the assistant that undergoes natural language processing and subsequently added to its knowledge base, **2)** basic user commands to enter and exit hypothetical mode, undo an operation, etc., **3)** a successful query where the assistant responds with an answer, and **4)** an unsuccessful query (due to unknown concepts) which is tackled by using interactive dialogues and case-based reasoning from experience.

Common-Sense Knowledge

Creating a Common-Sense AI is an important and challenging task in AI research today. Several research have explored this topic, inspiring ones being projects such as Mosaic Common Sense Knowledge Graphs and Reasoning (Zellers et al. 2018) and Aristo System (AI2 Allen Institute for Artificial Intelligence). SEVA aims to construct a targeted common-sense knowledge base which is explainable, can be trusted, and is carefully populated from trusted sources. This limits the usage of big data on semantic web including the usage of DBpedia (Lehmann et al. 2015). In addition to domain specific entities, common-sense knowledge also includes verb usages, reasoning tasks, and rules

that are collected from Systems Engineers, Handbooks, and the Web (limited).

SYSTEM CONCEPT

Based on the defined operations concept, primary functional requirements are derived - ability to **1)** Ingest information as text, **2)** Store information in an ontology, **3)** Perform reasoning on the ontology, **4)** Respond to interactive queries, **5)** Enter hypothetical mode, **6)** Understand time and schedule, **7)** Manage a Dynamic Ontology, **8)** Save specific interactions as 'Experience', **9)** Connect knowledge base with endowed models outside the system.

Requirements 1-8 will be described along with experiments in the upcoming sections. 9 is outside the scope of this paper, and for this purpose, we define three generic components: 1) Monitor, 2) Logicker, and 3) Inter-Module Communication Protocol. *'Monitor'* encompasses methodologies that can oversee, monitor, and assist in debugging the rest of the architecture. It acts as the central link between the system and the endowed models located outside. Architectural components for SEVA are depicted in *Figure 2*.

Inter-module Communication Protocol (ICP) in *Figure 2* depicts the underlying API (Application Program Interface), the methodology by which each module will communicate with one another. For example, it addresses the type and the format in which the NLP module should produce output in order to pass it to the ontology module.

The endowed models represent knowledge that are not innate to SEVA. The design does not aim to include all possible knowledge but rather to have the ability to access any outside knowledge when needed. Monitor encompasses a special component called *'Logicker'* that makes this knowledge transfer possible. This means that SEVA may not know how to solve 'orbital mechanics' or 'flight dynamics' problems until **1)** an experience occurs, **2)** the user teaches it or **3)** it endows the knowledge base with an orbital mechanics or flight dynamics module through Logicker. It is key to separate such functionalities from essential capabilities such as elementary math functions (arithmetic, logic, or relational operations) which will be included in SEVA's fundamental capability.

The architecture based on the aforementioned functional requirements is shown in Figure 1. Individual components and experiments will be described in the following sections.

NATURAL LANGUAGE PROCESSING (NLP Module)

The NLP module converts raw text into information that can be represented in the form of an ontology. The NLP functions are divided into 4 tasks: 1) analyze the linguistic structure of the domain, 2) extract entities and relations, 3) extract contextual rules, and 4) interact with the ontology to perform entity linking and verb normalization.

Triple extraction or relation extraction is a well known AI technique to populate knowledge bases. In recent years, Open Information Extraction (Open IE) has received significant attention as compared to traditional IE systems due to the ability to extract domain independent triples on a large scale. In addition, extractions are learned by using hand-crafted rules or automatically created training data without using any annotated treebanks. Extractors such as AI2 Open IE or Open IE 4.x (Etzioni et al. 2011), Stanford Open IE (Angeli, Premkumar, and Manning 2015), and ClausIE (Del Corro and Gemulla 2013) have shown significant progress in Open IE research and quality in their extractions. Open IE is particularly important in extracting relations for common-sense knowledge bases as shown in works such as (Mishra, Tandon, and Clark 2017) to perform question-answering on elementary science knowledge which is a part of Aristo (AI2 Allen Institute for Artificial Intelligence), an intelligent system, to perform QA about science. Our goal is to target it to Systems Engineering domain and address the challenges. An existing challenge for Open IE is unavailability of labeled corpus for evaluation which is addressed in works like (Stanovsky and Dagan 2016) and (Stanovsky et al. 2018). We create a baseline rule-based extractor and evaluate our system based on a manually constructed corpus.

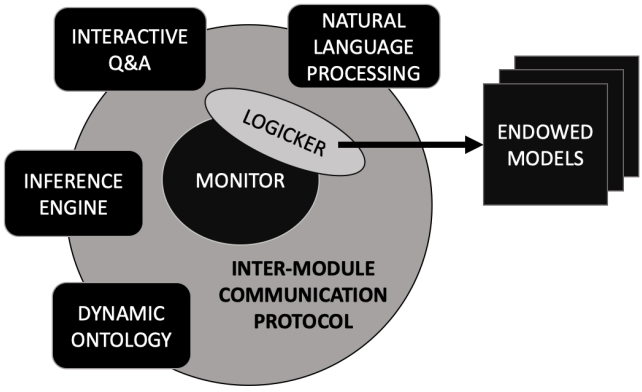

Figure 2: System Concept

Systems Engineering Domain Analysis

Question-Answer Types: QA pairs are collected from domain experts to understand the type of questions and answers that matter for individual systems engineers throughout a mission's life cycle. For an elementary knowledge represented as a [*subject-predicate-object*] triple, we categorize questions as follows:

1. **[S-P-?]**: [*subject-predicate-?*] questions that are looking for objects in sentences. Example: 'What is the mass of STI?'
2. **[S-?-O]**: [*subject-?-object*] questions that have *Yes/No* answers showing whether there exists the given predicate link between the subject and the object. Example: 'Is STI an instrument?'

Refer to Table 2 to see sample SE questions about SEVA Testing Instrument (STI).

Assumptions on Grammatical Constructs: We assume that the SE text is free of grammatical errors and that engineers converse with the system in grammatically correct

Input Sentence	Stanford Open IE	ClausIE	AI2 Open IE	SEVA-TOIE
STI, an instrument, has a 2500 pixel CCD detector	("STI" "has" "2500 pixel CCD detector") *incomplete/ missing information*: "STI is an instrument"	("STI" "is" "an instrument") ("STI" "has" "a 2500 pixel CCD detector")	(STI; has; a 2500 pixel CCD detector) (STI; [is]; an instrument)	(STI; has; CCD detector) (STI; is-a; instrument) (CCD detector; has-property; 2500 pixel)
STI is an instrument with a TRL value of 5	("STI" "is" "instrument") ("STI" "is instrument with" "TRL value of 5") ("instrument" "is with" "TRL value of 5") ("STI" "is instrument with" "TRL value")	("STI" "is" "an instrument with a TRL value of 5") ("STI" "is" "an instrument")	(STI; is; an instrument with a TRL value of 5) (STI; is an instrument with; a TRL value)	(STI; is; instrument) (instrument; has-property; TRL value) (TRL value; has-value; 5)
STI is scheduled for acoustic testing on July 3, 2015 from 2:00PM to 6:00PM.	produced 13 triples ("STI" "is" "scheduled") ("STI" "is scheduled for" X) X = various combinations of remaining sentence	("STI" "is scheduled" "for acoustic testing on July 3 2015 from 2:00 PM") ("STI" "is scheduled" "for acoustic testing on July 3 2015 to 6:00 PM") ("STI" "is scheduled" "for acoustic testing on July 3 2015")	(STI; is scheduled; for acoustic testing) *incomplete/ missing information*: "on July 3, 2015 from 2:00PM to 6:00PM"	(STI; is; scheduled) (scheduled; for; testing) (testing; has-property; acoustic) (testing; has-value; 3 2015 July) (scheduled; from; 2:00 PM) (scheduled; to; 6:00 PM)]

Table 1: Comparison of output from state-of-the-art Open Information Extractors and SEVA-TOIE. ***Red*** colored extractions are incomplete, incorrect, or noisy; ***Blue*** colored extractions need to be further granularized for ontology population.

SE Question	Answer
What is the mass of STI?	56kg
When is acoustic testing scheduled for STI?	07-10-2018 10:00AM
Where is STI located?	GSFC Building 28
Is STI an instrument?	Yes
Can Aerogel capture a Niacin molecule moving at 5km/s	No

Table 2: Sample SE QA

English. We analyze the grammatical constructs of SE text from the handbook (2017) and domain expert. We expect that engineering manuals, in general, are less likely to use pronouns such as *I, YOU, HE, SHE, WE, YOU, WHO,* and their object pronouns. However, pronouns such as *IT, ITS, THERE, THAT, THEY, THESE, THOSE, ONE, ONES, ITSELF, WHAT,* and *WHICH* are prevalent. Coreference resolution, associating these pronouns to its subject/object, is an essential task of the NLP module. However, we do not address this task in the evaluation and is left as future work. In a sentence, we focus on nouns, verbs, adjectives, and adverbials to study the phrase structure of the SE text. Our evaluation consists of only simple independent sentences in SE domain.

SEVA-TOIE

SEVA Targeted Open Information Extractor (TOIE) extracts simple essential relations from SE Text. A Systems Engineer was tasked with constructing a set of project specific

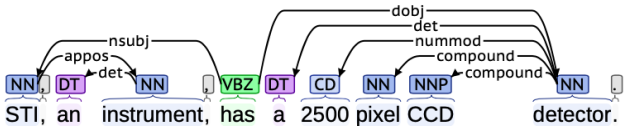

Figure 3: Dependencies produced by Stanford CoreNLP

sentences. The context in which these sentences are produced are instrument descriptions and instrument development meetings. Compound sentences are broken down and complex sentences are excluded. Our data set includes only independent sentences that do not require co-reference resolution. A snippet is shown in Table 3.

Sample Sentences in the Data Set
STI is SEVA Test Instrument. STI, an instrument, has a 2500 pixel CCD detector. STI has a length of 200 cm. STI is an instrument with a TRL value of 5. The spacecraft shall provide a direct Earth entry capability for 11500 m/s. The system shall have a 1.4 factor of safety. STI is scheduled for acoustic testing on July 3, 2015 from 2:00PM to 6:00PM.

Table 3: Data Set Snippet

The relations extracted are of type: $\{is-a, transitive-verb, has-property, has-value\}$. This is implemented by pattern matching on the dependency tree produced by the Stanford dependency parser, phrase chunking, and occasionally splitting at prepositions. Apart from subjects

(*nsubj*) and objects (*dobj*), the algorithm primarily focuses on a subset of universal dependencies: *case, nmod, compound, amod* (De Marneffe and Manning 2008). The implementation uses NLTK(Bird, Klein, and Loper 2009), Stanford Core NLP(Manning et al. 2014), and Stanford POS Tagger(Toutanova et al. 2003). The results from the base line model are used to evaluate powerful state-of-the-art OpenIE systems such as Open IE Standalone by AI2 (Michael Schmitz 2017), Stanford Open IE (Angeli, Premkumar, and Manning 2015), and ClausIE (Corro and Gemulla 2013). A Systems Engineer performed the evaluation based on three criteria: meaningfulness, completeness, and minimality of the extracted triples.

Basic dependencies produced by Stanford CoreNLP (Manning et al. 2014) for the sentence *"STI, a satellite, has a 2500 pixel CCD detector"* is shown in Figure 3. Pattern matching based on the dependency tree triples with the help of noun/verb phrase chunking produces SEVA-TOIE triples [*(STI; has; CCD detector), (STI; is-a; satellite), (CCD detector; has-property; 2500 pixel)*] that are granular enough to populate SEVA's knowledge base. The *'appos'* dependency defines the subject *'STI'* producing the "is-a" relationship. The object *"detector"* accompanied with compound modifier nouns and the cardinal number are broken down accordingly to produce the *"has-property"* triple. "is-a" relationship is produced from *cop* and *appos*. *[S-P-O]* triple is produced from *nsubj-VerbPhrase-dobj* pattern. Both *has-property* and *has-value* relations are produced from *amod* and *nmod-case*. *S-P* triple is produced from *nsubj/NN - CD* pattern. Compound words can be both split and joined accordingly to make the triple granular using *compound/nmod/CD*. Noun and verb phrases can be extracted using simple chunkers such as $NP: <DT>?<JJ.*>*(<NN.*>+<IN>)?<NN.*>+$ and $VP: <VB.*>+(<.*>?(<IN>|<VBG>))?<TO>?<VB.*>*$

Results & Research Challenge: Although powerful for large scale extractions, Table 1 shows that the extractions produced by the state-of-the-art systems can be noisy, incomplete, and need to be granularized in order to be put to practical knowledge base construction. Stanford Open IE uses distant supervision to train a classifier that generate clauses and uses hand-crafted patterns to produce triples. The extractor performed well at breaking down triples but sometimes produced triples in large quantities including noisy and incomplete ones. ClausIE uses hand-crafted rules for both clause generation and clause-type detection. The extractor often produced correct results. However, minimality of triples was the biggest concern. Open IE 4.x uses bootstrapping approach and training data to learn patterns. This system maintained a good balance between the number of extractions, minimality, correctness, and completeness. The task identified for future research is to extract high accuracy targeted triples on a large scale that are granular enough to be used to populate a knowledge base using the SE in the loop. This will make use the existing systems in addition to SEVA's internal extraction methodology.

Open IE can be customized to the SE domain by identifying the linguistic structure of the domain language which can be accompanied by tasks such as parts-of-speech tagging and semantic role labeling. A sentence represented using a constituency parser, such as (Zhu et al. 2013), can be used to extract sub-phrases from sentences or a dependency parser, such as Stanford Neural-network parser (Chen and Manning 2014), can be used to see the relation between words. A semi-supervised grammar induction can be used to address domain specific idioms, common-sense knowledge, and abbreviations. (Klein 2005) and (Spitkovsky 2013) are some relevant research that address unsupervised parsing and grammar induction.

Types of Input

We define three types of input informantion that is represented by SEVA's knowledge base: **1.** Project Specific Knowledge, **2.** Common-Sense knowledge, and **3.** Relational knowledge. Project specific knowledge comes from the engineer, meetings, and project documents. Common-Sense knowledge comes from the SE Handbook and other trusted sources. Relational knowledge consists of relational phrases, verbs, and their axiomatic relationships. This is closely connected to the ABox, TBox, and RBox knowledge population which will be described in the ontology section. Various sources of knowledge in SEVA are shown Figure 4.

Verbs, Vocabulary, and Relations

For successful ontological reasoning, the relations and the axioms in the ontology need to be well defined. For example, automatically constructing axiomatic relationships such as *inverse-of* (eg: ***partOf*** $\equiv hasComponent^{-}$) requires semantic understanding of words and phrases. Currently, these axiomatic relationships are constructed manually in the ontology. In this context, our future work will focus on using WordNet (Miller 1995) for synonyms and vector models for word and phrase representation such as Word2Vec (Mikolov et al. 2013) to extract deeper understanding of relational phrases, their synonyms, antonyms, and other common-sense domain specific semantic relationships.

Rule Extraction

Extracting rules in desirable forms such as horn clauses or description logic from text or triples is another essential NLP task for Ontological reasoning. Currently, rules are added manually using Semantic Web Rule Language (SWRL) (Horrocks et al. 2004) to SEVA's OWL ontology. Our goal is to assemble the grammatical structure from parsing, concepts, and relationships into clauses or rule-like structure. Some relations in the knowledge base need to be more restricted and better formulated as rules. Rules also fit better for Closed-World Assumption scenarios. Logic extraction from text requires unsupervised deep semantic parsing mentioned in works such as (Poon and Domingos 2009; Schoenmackers et al. 2010; Jha and Finegan-Dollak 2011). *Example Text*: Aerogel can capture a niacin molecule that has speed less than 5m/s. A rule that is ideally extracted from the given text: $hasSpeed(X,Y) \wedge (Y < 5m/s) \wedge NiacinMolecule(X) \wedge Aerogel(Z) \Rightarrow canCapture(Z,X)$

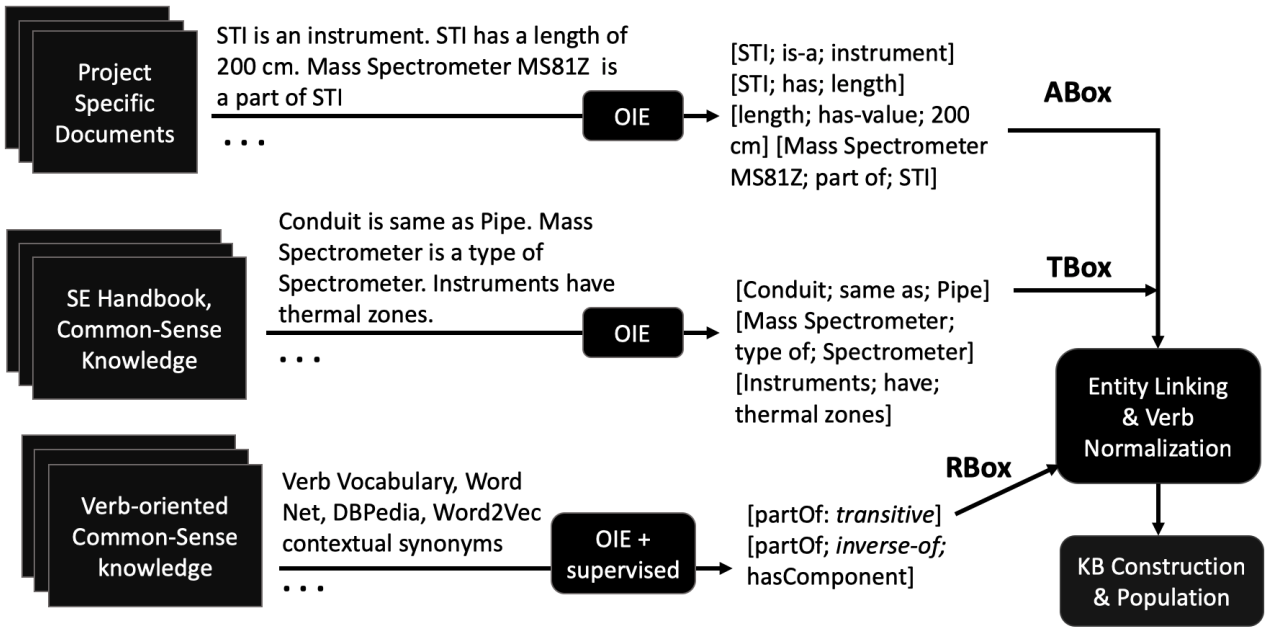

Figure 4: Types of Input Knowledge

Table 4: SEVA DL Example

Assertion Box (ABox)	Terminology Box (TBox)	Relational Box (RBox)
Spacecraft(DiscoveryShuttle) implies that "Discovery Shuttle is an instance of Spacecraft" ***partOf(StarTracker, DiscoveryShuttle)***) implies that "StarTracker is a part of Discovery Shuttle"	***Conduit*** $\equiv$ ***Pipe*** implies "same-as" relationship ***MassSpectrometer*** $\sqsubseteq$ ***Spectrometer*** implies "sub-class" relationship	***partOf*** $\circ$ ***partOf*** $\sqsubseteq$ ***partOf*** represents transitive property of the role ***partOf*** $\equiv hasComponent^{-}$ represents inverse property of two roles

Time Information

SEVA gives special consideration for time entities as they are significant in scheduling and maintaining a dynamic knowledge base with temporal reasoning. The result of passing the sentences *"STI is scheduled for acoustic testing on July 3, 2015 from 2:00PM to 6:00PM. The instrument is scheduled for vibration testing on July 3, 2015 at 4:30PM for 3 hours."* to the off-the-shelf 7-class Stanford Named Entity Tagger (Surdeanu et al. 2011) produces time and date tags shown in Figure 5. However, it failed to tag *"2:00PM"* and *"6:00PM"*. In addition, *"3 hours"* also need to be semantically understood in context. Both supervised or unsupervised approaches to construct a model that is trained on the SE text to detect various time entities are proposed. This includes making use of the Stanford NER to improve its results in tagging time entities in the SE domain.

"STI is scheduled for acoustic testing on July 3, 2015 from 2:00PM to 6:00PM.
The instrument is scheduled for vibration testing on July 3, 2015 at 4:30PM for 3 hours."

Figure 5: 7-Class Stanford Named Entity Tagger used for Date/Time

KNOWLEDGE BASE (ONTOLOGY Module)

With the help of the NLP module, the ontology module links the subjects and objects identified to the concepts and instances already present in the ontology. Relations are created by normalizing the verbs to its root form. For our baseline, we choose a Semantic Web based approach to building SEVA's ontology. This provides extensibility and access to vast knowledge of information. Usage of resources such as DBpedia will be replaced with a SE specific common-sense ontology in the future.

Semantic Web and Web Ontology Language (OWL)

The idea of semantic web framework is to establish a common representation of data on the web that can be used universally. SEVA uses OWL 2's built-in set of specifications called Resource Description Framework (RDF) (Klyne and Carroll 2003); essentially a set of RDF triples. The knowledge representation language is Description Logic (DL), a

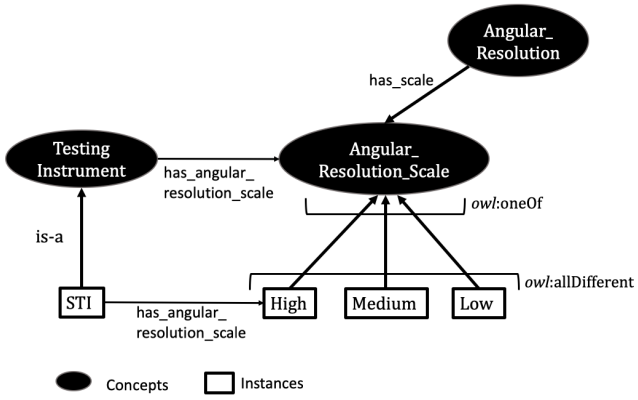

Figure 6: Ontology snippet for *"STI has high angular resolution"*

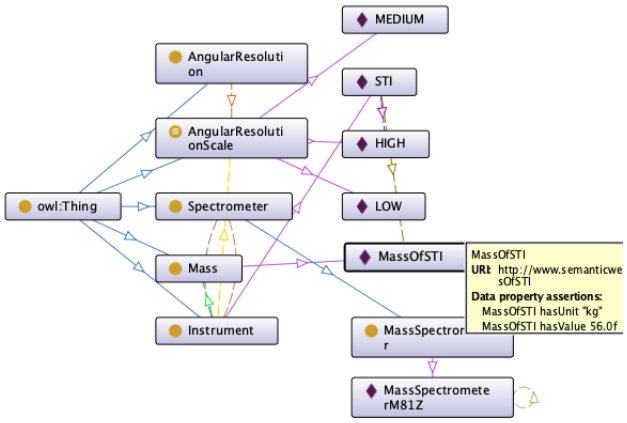

Figure 7: Sample OWL Ontology in Protege

decidable fragment of First-Order Logic, and rules which corresponds to OWL 2 DL and OWL 2 RL (Motik et al. 2009) respectively. Description Logic (Krötzsch, Simancik, and Horrocks 2012) provides the formal semantics for designing ontologies in OWL. DLs consist of concepts, roles, and individuals. Ontological axioms define the expressiveness and the types of knowledge it can represent. SEVA's design uses a popular expressive fragment of DL called *SROIQ* (Horrocks, Kutz, and Sattler 2006) ontology as it fits the types of knowledge SEVA likes to store and reason from in its knowledge base. An snippet of SEVA's knowledge base is shown in Table 4.

Knowledge created in the ABox is instantiations of concepts in the TBox or relations between already constructed instances. This is typically received from project specific documents. On the other hand, TBox consists of concepts and their relationships. This knowledge is received from both project specific documents as well as guidelines, SE Handbook, and common-sense knowledge. RBox focuses on axiomatic relationships between relations and is responsible for ontological reasoning along with *is-a* relationships. This information cannot be found in documents and need to be constructed automatically or manually. This verb-oriented knowledge is considered common-sense knowledge which is constructed specifically for the SE domain.

Knowledge Representation Example

Next, we describe a representation example that shows the complexity of our knowledge base construction and population process. Consider a seemingly simple example *"STI has high angular resolution"*. SEVA-TOIE produces two triples *[('STI', 'has', 'angular resolution'), ('angular resolution', 'has-property', 'high')]*.

Knowledge Base Construction: This phase gives context by constructing the TBox concepts and relations, and necessary instantiations in the ABox. This needs to be performed prior to passing the aforementioned example sentence. The knowledge *"STI is an instrument"* or ***Instrument(STI)*** must pre-exist along with the concept of *"angular resolution"* such that the entities can be linked.

Knowledge Base Population: This phase populates the ABox. When there are only predefined named entities and relations in the TBox, this step can be generally replaced by the methodologies of Named Entity Recognition customizing to the domain and Relation Extraction from templates and by training a model (Surdeanu et al. 2011). However, Open IE extracts triples with no pre-specified templates or vocabulary from arbitrary text. Refer to Figure 6 which shows the snippet of ontology structure for the example sentence. A sample OWL Ontology constructed is shown in Figure 7 using Protege OntoGraf visualizer. (Noy et al. 2003; Falconer 2010). Current implementation considers only a few hand picked structures. Our goal is to automatically, and through human-in-the-loop process, construct such domain specific and common-sense ontology structures which will later help in the creation of a dynamic and explainable knowledge base.

Ontology Compartments

SEVA's ontology is divided into Base ontology and Custom ontology. Custom ontology consists of domain specific concepts and instantiations, mostly the TBox and ABox axioms. ***Instrument(STI)*** and ***partOf(MassSpectrometerMS81Z,STI)*** fall in to custom ontology. Population of a custom ontology is easily performed as long as a solid base ontology is available. This means that if a framework of concepts and relations is available, their instances and relations can be added without any ambiguity. Base ontology contains the essential ingredients or building blocks for a common-sense ontology upon which a SE can build their customized ontology. It consists of basic axioms and rules that govern the ontology. It predominantly describes relationships and properties of predicates or RBox axioms such as $partOf \circ partOf \sqsubseteq partOf$ representing transitive property or $partOf \equiv hasComponent^{-}$ describing inverse relationship. It also contains the TBox and ABox information that are common-sense knowledge such as SE guidelines, synonyms, and abbreviations. *Example*: *Conduit* $\equiv$ *Tube* and *isAbbreviationFor(SLS, SpaceLaunchSystem)*.

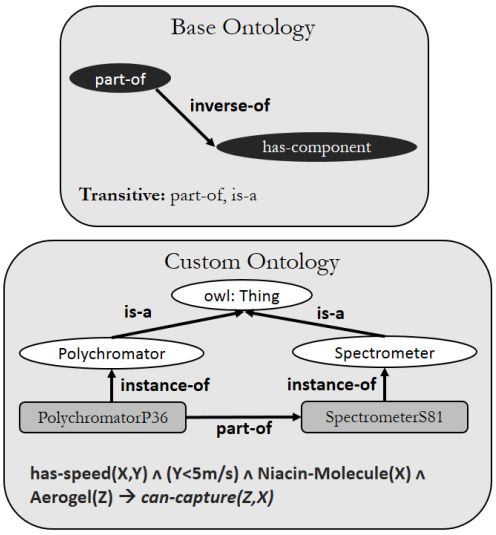

Figure 8: Ontology Compartments

Research Challenge identified for this step is to automatically construct the base ontology. A challenging aspect of this step, in order to have an explainable system, is to define what constitutes the input for the base ontology. Using the massive information available on the web is neither a scalable nor a trustable solution. Using the NASA SE Handbooks is a good first step. However, common-sense science and math knowledge is assumed which needs to be learned as a prerequisite. Essentially, we need to scale and define what common-sense knowledge is for the SE domain and identify the sources such that the knowledge is complete and scoped for a personal assistant's knowledge base.

Dynamic Knowledge

SEs deals with constantly changing information including project specifications, time, events, and schedule. New information will need to be added and old information will need to be updated. Ontology consistency needs to be checked on each update. Multiple versions of the ontology need to be maintained in order to perform temporal reasoning, scheduling, and undo operation. SEVA is designed as a dynamic work bench for the SE. Guidelines for a dynamic ontology is presented in works such as (Pittet, Nicolle, and Cruz 2012; Plessers, De Troyer, and Casteleyn 2007). To aid user friendliness, SEVA's design follows that newer information overrides older conflicting information. SE will be called in the loop only if the conflicting information contains a Base ontology entity or if the user has made some pre-existing specifications. For example, the user is notified only if there is a conflict in logic or reasoning process but not a mass update. The dynamic nature is closely associate with the ability to handle time. Representing time in ontology and intelligent systems have been studied by works such as (Allen 1991; Batsakis, Stravoskoufos, and Petrakis 2011). ***Research Challenege*** identified for this step is to construct a knowledge base that handles dynamic information and can perform temporal reasoning.

Recent developments in Knowledge Graph Identification research by jointly performing Entity Resolution, Node Labelling, and Link Prediction while enforcing ontological constraints in works such as (Pujara and Getoor 2014; Pujara et al. 2013; Choudhury et al. 2017) using Probabilistic Soft Logic is another direction for SEVA's Dynamic Knowledge Base Construction. These works show the importance of identifying the facts that may require revision due to the varying degrees of confidence in the extractions produced by the Open IE systems.

REASONING ENGINE

The primary function of this module is to perform inference on the ontology. TBox inferences include subsumption, consistency, satisfiability, equivalence, and disjoint checking. Whereas ABox inferences include instance checking and retrieval (Walton 2007). SEVA's design considers multiple options for reasoners depending on the design choices for Knowledge Base construction:

1. *Using an OWL 2 RL Reasoner throughout*: This option suggests the use of an RL reasoner for both TBox and ABox reasoning. This is less expressive but runs in polynomial time. Some example reasoners are BaseVISor (Matheus et al. 2006), ELLY (Siorpaes and Winkler 2010), Jena (Jena 2014), and Logic Programming approaches to RL using SWI-Prolog (Walton 2007).
2. *Using a DL Reasoner for TBox reasoning and an RL Reasoner for ABox*: TBox has more of static information. So using more expressive and computationally complex DL reasoner for TBox reasoning gives extensibility to SEVA in the future. Some DL reasoners are FaCT++ (Tsarkov and Horrocks 2006), RACERPRO (Haarslev et al. 2012), Pellet (Sirin et al. 2007), and Hermit (Shearer, Motik, and Horrocks 2008). An example of this kind of implementation is DLEJena which uses Pellet for TBox reasoning and Apache Jena for ABox reasoning (Meditskos and Bassiliades 2010).
3. *Using a DL Reasoner for TBox and non-OWL rule-based reasoner for ABox*: A conversion from OWL to rules tailored to the rule-engine is required in this case. For example, a conversion of OWL to Jess rules is required when using Jess rule engine (Friedman-Hill and others 2008).

The choices can be evaluated only after the dynamic nature of the knowledge base is addressed. For the baseline implementation in OWL and Protege, we use off-the-shelf Pellet reasoner.

Open/Close World Assumption & Explainability

If something is not known to be true, an open-world assumption (OWA) assumes incomplete information, and thus the assistant responds with "I don't know" as the answer, while a closed-world assumption (CWA) assumes complete information, thus the assistant responding with "false" as the answer. Formal definitions of these two assumptions can be found in (Russell and Norvig 2003). OWA is best seen in large systems such as semantic web where it is not feasible to incorporate all possible information in the ontology. However, some predicates are better represented by the CWA.

For example, "is X partOf Y"? If the user has never mentioned that "thrusters are part of the spacecraft", is it reasonable to assume they are not? This means that SEVA's knowledge base should support an open world assumption in general but certain predicates should also be allowed to have closed world property. Works such as (Damásio et al. 2006) and (Knorr 2011) study this type of integration. Base ontology in *Figure 8* will integrate OWA and CWA to the predicates. This will in turn impact the choice of reasoning engines for SEVA as most rule-like knowledge formalisms supports CWA while Description Logic support OWA.

For every NASA Engineer who deals with mission critical information, their personal assistant needs to be highly trustable in its information assimilation task. Completeness of the knowledge base and the type of assumptions mentioned above play an important role in explaining how the assistant came to an answer. Probability of any answer should be ideally one and the system should be able to show the reasoning steps by which the answer is obtained.

QUESTION-ANSWERING

SEVA adopts the knowledge-based paradigm (Jurafsky and Martin 2017) for question answering by building a semantic representation of the question. NLP module is needed for parsing, extracting relations from the query, or extracting rules in some form of query language such as SPARQL (Walton 2007) using the same OIE paradigm. An example of the DL query in the ontology shown in Figure 7 is [*hasAngularResolutionScale* value *HIGH*] producing STI as the result.

Querying on an ontology link which consists of Subject-Predicate-Object can be of two forms: a) asking for the object and b) asking for the predicate. Possible responses with an Open World assumption can be a) an answer which is contextually correct according to SEVA, b) a logical conversation with the user to understand the missing link or predicate, or c) an interaction or instructions to the user to teach SEVA about a missing concept.

Capturing Experience

Capturing Experience is a process by which the assistant is taught logically how to arrive at the conclusion or can be viewed as an algorithm being taught to the assistant for case-based reasoning. It can use an existing algorithm to solve a similarly appearing problem with the guidance of the SE. The following scenario represents the "Unknown due to OWA" answer type where SEVA linguistically understands the question being asked, however does not poses enough information to answer it. It then interacts with the user to learn logic. The conversation between the user and SEVA shows an example of how an experience is captured into a case.

User: Can Aerogel capture a Niacin molecule moving at $3km/s$?
SEVA: "I don't know"
User: Start a rule.
SEVA: Enter the necessary conditions.
User: Depth of Aerogel times density of Aerogel divided by density of niacin molecule should be greater than $8mm$.
Speed of niacin molecule should be less than $5km/s$.
SEVA: Rule Saved! No, Aerogel cannot capture a niacin molecule moving at $3km/s$.

Through such an interactive session, SEVA lets user create custom rules. Creating contextual awareness is done through identifying structure of queries and applying case-based reasoning such as in (Kolodner 1992) with the SE's assistance at each learning step. For example, from the query - *Can Aerogel capture a Niacin molecule traveling at speed 100m/s?*, a model can be extracted and can be applied to a different but structural similar question *Can Titanium capture an Inositol molecule moving at 2 km/s?*. The user is required in the process to confirm whether SEVA can use the same logic (or reasoning) to answer the question and also to teach SEVA about Titanium if needed.

CONCLUSION AND FUTURE WORK

This work introduced SEVA - a vision of a Systems Engineer's personal assistant. It described SEVA's architectural design - the big picture, motivation, and its technical plausibility. SEVA is a single-user system designed to assist SEs in their day-to-day activities. The work designed the architecture for a framework with specific goals and constraints within the context of a NASA SE who deals with risk-averse complex engineering projects. SEVA makes it easier for the SE to focus on the creative problem solving by taking care of all the tedious book-keeping and potentially error-prone information assimilation. SEVA is a trustable and explainable system with a domain independent SE framework that grows by human-in-the-loop learning and becomes user-specific or domain-specific over time. The work discussed how SEVA performs natural language processing, information management, reasoning, learning, and question-answering. We described what SEVA is and what it is not, as well as the tools with which such an implementation is feasible and the areas that require further research and development.

We designed the overall architecture and implemented a baseline Open Information Extractor and a Knowledge Base Module for SEVA. Several sub-components are provided with high-level descriptions including the research challenges and implementation directions. Detailed implementation and technical evaluation of these modules are currently in progress and are left as future work.

Although SEVA is designed for a SE, the idea is extensible to all domains where personal assistants (ones which can grow alongside the user) are needed. Thus, extending the idea to other domains is a future direction. Interconnectedness of different domains and different engineers add complexity to any project. We envision multiple SEVAs each belonging to a specific engineer filling the gap of information assimilation and distributed coordination. Application of assistants in such collaborative settings is another area of future work.

References

AI2 Allen Institute for Artificial Intelligence. Aristo: An intelligent system that reads, learns, and reasons about science. https://allenai.org/aristo/. Accessed: 2018-08-12.

Allen, J. F. 1991. Time and time again: The many ways to represent time. *International Journal of Intelligent Systems* 6(4):341–355.

Angeli, G.; Premkumar, M. J.; and Manning, C. D. 2015. Leveraging linguistic structure for open domain information extraction. *Proceedings of the ACL*.

Batsakis, S.; Stravoskoufos, K.; and Petrakis, E. G. 2011. Temporal reasoning for supporting temporal queries in owl 2.0. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, 558–567. Springer.

Bird, S.; Klein, E.; and Loper, E. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Brooks, R. A. 1997. The intelligent room project. In *Cognitive Technology, 1997. Humanizing the Information Age. Proceedings., Second International Conference on*, 271–278. IEEE.

Chen, D., and Manning, C. D. 2014. A fast and accurate dependency parser using neural networks. *Proceedings of EMNLP*.

Choudhury, S.; Agarwal, K.; Purohit, S.; Zhang, B.; Pirrung, M.; Smith, W.; and Thomas, M. 2017. Nous: Construction and querying of dynamic knowledge graphs. In *Data Engineering (ICDE), 2017 IEEE 33rd International Conference on*, 1563–1565. IEEE.

Corro, L. D., and Gemulla, R. 2013. Open information extraction via contextual sentence decomposition. *Proceedings of the 22nd international conference on World Wide Web* 355–366.

Damásio, C. V.; Analyti, A.; Antoniou, G.; and Wagner, G. 2006. Supporting open and closed world reasoning on the web. In *PPSWR*, volume 4187, 149–163. Springer.

De Marneffe, M.-C., and Manning, C. D. 2008. Stanford typed dependencies manual. Technical report.

Del Corro, L., and Gemulla, R. 2013. Clausie: clause-based open information extraction. In *Proceedings of the 22nd international conference on World Wide Web*, 355–366. ACM.

Domingos, P. 2015. *The master algorithm: How the quest for the ultimate learning machine will remake our world*. Basic Books.

Etzioni, O.; Fader, A.; Christensen, J.; Soderland, S.; and Mausam, M. 2011. Open information extraction: The second generation. In *IJCAI*, volume 11, 3–10.

Falconer, S. 2010. Ontograf protege plugin. *Place: Available at: http://protegewiki. stanford. edu/wiki/OntoGraf[Accessed: 21/03/2014]*.

Ferrucci, D.; Brown, E.; Chu-Carroll, J.; Fan, J.; Gondek, D.; Kalyanpur, A. A.; Lally, A.; Murdock, J. W.; Nyberg, E.; Prager, J.; et al. 2010. Building watson: An overview of the deepqa project. *AI magazine* 31(3):59–79.

Ferrucci, D.; Levas, A.; Bagchi, S.; Gondek, D.; and Mueller, E. T. 2013. Watson: beyond jeopardy! *Artificial Intelligence* 199:93–105.

Fosse, E.; Harmon, C.; Lefland, M.; Castillo, R.; and Devereaux, A. 2014. Inheriting curiosity: Leveraging mbse to build mars2020. https://trs.jpl.nasa.gov/handle/2014/45871.

Friedman-Hill, E., et al. 2008. Jess, the rule engine for the java platform.

Gruber, T. R. 2009. Siri, a virtual personal assistantbringing intelligence to the interface.

Haarslev, V.; Hidde, K.; Möller, R.; and Wessel, M. 2012. The racerpro knowledge representation and reasoning system. *Semantic Web* 3(3):267–277.

Hart, L. E. 2015. Introduction to model-based system engineering (mbse) and sysml. http://www.incose.org/docs/default-source/delaware-valley/mbse-overview-incose-30-july-2015.pdf. Accessed: 11-09-2017.

Horrocks, I.; Patel-Schneider, P. F.; Boley, H.; Tabet, S.; Grosof, B.; Dean, M.; et al. 2004. Swrl: A semantic web rule language combining owl and ruleml. *W3C Member submission* 21:79.

Horrocks, I.; Kutz, O.; and Sattler, U. 2006. The even more irresistible sroiq.

Jena, A. 2014. the apache jena project.

Jha, R., and Finegan-Dollak, C. 2011. An unsupervised method for learning probabilistic first order logic models from unstructured clinical text. In *Icml Workshop on Learning from Unstructured Clinical Text*.

Jurafsky, D., and Martin, J. H. 2017. Chapter 28 question answering.

Klein, D. 2005. *The Unsupervised Learning of Natural Language Structure*. Ph.D. Dissertation, Stanford University.

Klyne, G., and Carroll, J. J. 2003. Resource description framework (rdf): Concepts and abstract syntax. http://www.w3.org/TR/2003/WD-rdf-concepts-20030123. Accessed: 08-03-2017.

Knorr, M. 2011. Combining open and closed world reasoning for the semantic web.

Kolodner, J. L. 1992. An introduction to case-based reasoning. *Artificial intelligence review* 6(1):3–34.

Krötzsch, M.; Simancik, F.; and Horrocks, I. 2012. A description logic primer. *arXiv preprint arXiv:1201.4089*.

Lee, J.; Kim, G.; Yoo, J.; Jung, C.; Kim, M.; and Yoon, S. 2016. Training IBM watson using automatically generated question-answer pairs. *CoRR* abs/1611.03932.

Lehmann, J.; Isele, R.; Jakob, M.; Jentzsch, A.; Kontokostas, D.; Mendes, P. N.; Hellmann, S.; Morsey, M.; Van Kleef, P.; Auer, S.; et al. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web* 6(2):167–195.

Manning, C.; Surdeanu, M.; Bauer, J.; Finkel, J.; Bethard, S.; and McClosky, D. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, 55–60.

Matheus, C.; Dionne, B.; Parent, D.; Baclawski, K.; and Kokar, M. 2006. Basevisor: A forward-chaining inference engine optimized for rdf/owl triples. In *5th International Semantic Web Conference*.

Meditskos, G., and Bassiliades, N. 2010. Dlejena: A practical forward-chaining owl 2 rl reasoner combining jena and pellet. *Web Semantics: Science, Services and Agents on the World Wide Web* 8(1):89–94.

Michael Schmitz, Harinder Pal, B. M. M. G. 2017. Open ie. `https://github.com/allenai/openie-standalone`.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.

Miller, G. A. 1995. Wordnet: a lexical database for english. *Communications of the ACM* 38(11):39–41.

Mishra, B. D.; Tandon, N.; and Clark, P. 2017. Domain-targeted, high precision knowledge extraction. *Transactions of the Association for Computational Linguistics* 5:233–246.

Motik, B.; Grau, B. C.; Horrocks, I.; Wu, Z.; Fokoue, A.; Lutz, C.; et al. 2009. Owl 2 web ontology language profiles. *W3C recommendation* 27:61.

Noy, N. F.; Crubézy, M.; Fergerson, R. W.; Knublauch, H.; Tu, S. W.; Vendetti, J.; and Musen, M. A. 2003. Protégé-2000: an open-source ontology-development and knowledge-acquisition environment. In *AMIA... Annual Symposium proceedings. AMIA Symposium*, volume 2003, 953–953. American Medical Informatics Association.

Panton, K.; Matuszek, C.; Lenat, D.; Schneider, D.; Witbrock, M.; Siegel, N.; and Shepard, B. 2006. *Common Sense Reasoning – From Cyc to Intelligent Assistant*. Berlin, Heidelberg: Springer Berlin Heidelberg. 1–31.

Pittet, P.; Nicolle, C.; and Cruz, C. 2012. Guidelines for a dynamic ontology-integrating tools of evolution and versioning in ontology. *arXiv preprint arXiv:1208.1750*.

Plessers, P.; De Troyer, O.; and Casteleyn, S. 2007. Understanding ontology evolution: A change detection approach. *Web Semantics: Science, Services and Agents on the World Wide Web* 5(1):39–49.

Poon, H., and Domingos, P. 2009. Unsupervised semantic parsing. In *Proceedings of the 2009 Conference on EMNLP: Volume 1*, 1–10. ACL.

Pujara, J., and Getoor, L. 2014. Building dynamic knowledge graphs. In *NIPS Workshop on Automated Knowledge Base Construction*, 9.

Pujara, J.; Miao, H.; Getoor, L.; and Cohen, W. 2013. Knowledge graph identification. In *International Semantic Web Conference*, 542–557. Springer.

Rich, C., and Waters, R. C. 1987. The programmers apprentice project: A research overview.

Russell, S. J., and Norvig, P. 2003. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition.

Schoenmackers, S.; Etzioni, O.; Weld, D. S.; and Davis, J. 2010. Learning first-order horn clauses from web text. In *Proceedings of the Conference on EMNLP*, 1088–1098.

2017. Nasa systems engineering handbook revision 2. `https://www.nasa.gov/connect/ebooks/nasa-systems-engineering-handbook`. Accessed: 2018-08-12.

Shearer, R.; Motik, B.; and Horrocks, I. 2008. Hermit: A highly-efficient owl reasoner. In *OWLED*, volume 432, 91.

Siorpaes, K., and Winkler, D. 2010. An elp reasoner.

Sirin, E.; Parsia, B.; Grau, B. C.; Kalyanpur, A.; and Katz, Y. 2007. Pellet: A practical owl-dl reasoner. *Web Semantics: science, services and agents on the World Wide Web* 5(2):51–53.

Spitkovsky, V. I. 2013. *Grammar Induction and Parsing with Dependency-and-Boundary Models*. Ph.D. Dissertation, Stanford University.

Stanovsky, G., and Dagan, I. 2016. Creating a large benchmark for open information extraction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2300–2305.

Stanovsky, G.; Michael, J.; Zettlemoyer, L.; and Dagan, I. 2018. Supervised open information extraction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, 885–895.

Surdeanu, M.; McClosky, D.; Smith, M. R.; Gusev, A.; and Manning, C. D. 2011. Customizing an information extraction system to a new domain. In *Proceedings of the ACL 2011 Workshop on Relational Models of Semantics*, 2–10. Association for Computational Linguistics.

Toutanova, K.; Klein, D.; Manning, C. D.; and Singer, Y. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, 173–180. Association for Computational Linguistics.

Tsarkov, D., and Horrocks, I. 2006. Fact++ description logic reasoner: System description. *Automated reasoning* 292–297.

Walton, C. 2007. *Agency and the semantic web*. Oxford University Press on Demand.

Zellers, R.; Bisk, Y.; Schwartz, R.; and Choi, Y. 2018. Swag: A large-scale adversarial dataset for grounded commonsense inference. *CoRR* abs/1808.05326.

Zhu, M.; Zhang, Y.; Chen, W.; Zhang, M.; and Zhu, J. 2013. Fast and accurate shift-reduce constituent parsing. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics* 434–443.