A Systems Engineer's Virtual Assistant (SEVA)

Jitin Krishnan
George Mason University
Department of Computer Science
Fairfax, Virginia 22030
jkrishn2@gmu.edu

Patrick Coronado
Instrument Development Center
NASA Goddard Space Flight Center
Greenbelt, Maryland 20771
patrick.l.coronado@nasa.gov

Trevor Reed
Robot Operations
NASA Jet Propulsion Laboratory
Pasadena, California 91109
trevor.reed@jpl.nasa.gov

ABSTRACT

A Systems Engineer's Virtual Assistant (SEVA) is a novel attempt towards a trustable, human-in-the-loop, interactive, and personal software system designed to assist a Systems Engineer in their daily work environment through complex information management and high-level query-answering to augment their problem-solving abilities. The work defines fundamental characteristics of the assistant by understanding operational, functional, and system requirements from Systems Engineers. Using these specific objectives and constraints, the architecture of a personal assistant is proposed using tools from Natural Language Processing, Knowledge Representation, and Reasoning. SEVA collects information by ingesting various types of discipline-specific documents and human interaction. It acts as a workbench to manage dynamic information about projects and analyzes hypothetical scenarios. It handles information relating to schedule and resources and performs temporal reasoning. It makes logical inferences and derives new information, when needed, in order to answer questions asked by the user. It also performs case-based reasoning using experiences learned from interacting with the user. The work addresses above tasks within a Systems Engineering context where trust in the answers is paramount. This requires the assistant to be able to explain the reasoning behind the answers or models learned with no ambiguity. In addition, the knowledge base design proposes a hybrid approach to build a domain-independent framework with which domain-specific users can attune it to their preferences: giving it freedom from traditionally structured knowledge representation paradigms.

KEYWORDS

Systems Engineering, Personal Assistant, Natural Language Processing, Knowledge Base Construction, Reasoning, QA

ACM Reference format:
Jitin Krishnan, Patrick Coronado, and Trevor Reed. 2018. A Systems Engineer's Virtual Assistant (SEVA). In *Proceedings of First Workshop on Knowledge Base Construction, Reasoning and Mining, Los Angeles, California USA, Feb 2018 (KBCOM),* 9 pages.

KBCOM, Los Angeles, California USA

INTRODUCTION

Intelligent personal assistants have drawn attention in recent years due to their ubiquitous nature of making human lives easier. However, 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. It is a dream for most scientists and engineers to have an assistant such as Iron Man's Jarvis [48], who takes care of the tedious book-keeping aspects and assists them in creative problem solving. Systems Engineers (SE) deal with large amounts of data 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 reasoning, recall, resource and schedule queries from the user.

Natural Language Understanding is a key aspect of SEVA. It performs information extraction to create logic and relationships from interacting with the SE. From this, the knowledge base is constructed using a novel hybrid approach which aims to build a domain independent architecture for a domain specific user. This means that the user, by interacting with the domain independent system over time, will make it user-specific or domain-specific; a framework that can be used to build any personal assistant system in the future. This can also solve the pervasive problem where engineers need to learn new software systems or languages for effective coordination on large shared projects. Instead of engineers or scientists trying to coordinate with each other, their assistants can coordinate. In this paper, we discuss the state-of-the-art systems in these contexts and how, in the long run, user-specific assistants such as SEVA can benefit large scale engineering projects.

In the following sections, we discuss background works and the research in artificial intelligence that share similar spirits. The Operations Concept section describes Systems Engineering overview. System Concept and Key Components present how the functional requirements are mapped in to the architecture and reasons for the design choices. Future works and conclusion present the current state of the project and the next steps toward achieving this vision.

BACKGROUND

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 tackles 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 [17]. For example, the success of Jet Propulsion Laboratory's Curiosity Mars Rover has also led NASA to identify the potential challenges in building its second rover MARS2020 [13]. MBSE is adopted to aid the design of its flight system model. 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 [13]. MBSE aims to tackle the problems of lack of a common language between different disciplines using a formal language called SysML. It creates a shared system by which information blocks caused due to change propagations and interdependent disciplines are prevented. Improvements to MBSE have been recently proposed to incorporate Ontologies to provide formal representations to the models and entities [12].

NASA's unique and cutting edge engineering processes require the SEs to attempt problems that have never been attempted before. What MBSE essentially tackles is this growing complexity of projects which can easily leave individual engineers error prone in information assimilation, create communication gaps, and lead to loss of knowledge. However, systems such as MBSE present a steep learning curve for new learners who will need to learn the system or a new language. Our work takes the opposite approach where the system learns the user's language. This work proposes the building blocks necessary to design such personal assistants in large scale engineering projects. Thus, conceptually, a SEVA (or multiple SEVAs coordinating together) with a well developed ontology and reasoning skills can be inclusive of all functionality of MBSE. Models in SEVA are created through a learning process over time, stored in a dynamic ontology, and poses a framework with the ability to communicate with outside models.

Popular intelligent assistants such as Siri provide proactive assistance on a very specific set of contextual tasks such as calling a friend, sending a message, booking a flight, 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 [15]. Another feature is that they learn the users' actions and behaviors such that they can pro-actively help users in making decisions or performing tasks. One example is learning the users' language styles from keyboard usage. SEVA is neither proactive nor predictive, the two being cautionary factors for high-risk engineering missions. However, two key aspects of these assistants which appeal to SEVA are the idea of getting attuned to the individual using it and temporal awareness which we explore in this work. On the other hand, large scale knowledge systems such as IBM Watson perform statistical inferencing to answer a question by going through ingesting millions of documents. IBM Watson is a technology platform that uses natural language processing and machine learning to reveal insights from large amounts of unstructured and structured data [21]. The main idea behind Watson is to essentially answer a question. Watson is trained by QA sessions and answers are returned with probability or confidence [28]. 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 [11, 27]. On the other hand, SEVA performs ontology based answering and its design understands that its knowledge could be incomplete. SEVA's knowledge is contextual facts and if it knows an answer, that answer should have an accuracy of 100%. For example, "mass of instrument X is 5kg" is a true statement at time $t1$. An ontological knowledge representation is essential in such scenarios to produce low-risk results as it is essential for the knowledge base to dynamically change when the mass increases to 6kg at time $t2$, but still keeping track of the old mass. An expert system is typically created by instilling as much domain knowledge as possible. However, since SEVA is a personal assistant, the design only needs to have a platform onto which the user can build a user-specific or domain-specific system over time by interaction or ingesting information.

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 [34]. The idea was to create a system that can serve as a foundation to all future expert systems. However, millions of assertions are handcoded to the system and the amount of knowledge required to be learned was one of the major criticisms of the project [9]. The idea of common sense is important to SEVA. In order to create an elementary system, definitions of basic truths have to established from the systems engineering perspective. For example, $1 + 1 = 2$ and propositional logic are relevant common sense to a SE. More relevant common sense knowledge can be imparted to SEVA along the way just like a child developing common sense by interacting with the environment.

The MIT Programmer's Apprentice Project [5, 38], 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 [38] which is also relevant in SEVA. The apprentice project featured Plan Calculus and a hybrid apprentice system; one for design and one for requirements in programming[38].

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. Many existing systems either create a steep learning curve, do not capture the complexity of engineering projects, take the approach to create an all-knowing expert system, or require huge amount of labeled data sets. With its domain independent architecture, SEVA addresses such issues of intelligent personal assistants and identifies future research areas.

OPERATIONAL CONCEPT

SEVA has many uses for SEs in their daily work environment, such as during group meetings and individual-specific office work, the assistant will be able to store information and answer questions asked by the user. It will be able to recall information, perform inference, answer questions involving temporal information, solve hypothetical (what-if) questions, and handle a combination of tasks. The assistant learns new information from user-input and acquires new problem-solving skills through inference or through the '*experience*' gained from the question-answering session. Together, these functions improve the efficiency and problem-solving ability of an SE and mitigate faulty human memory and logic.

SEVA would be able to contribute during meetings in multiple ways. During a regular update meeting, SEVA could gather new information. For example, information such as "*deadline for the Near Infrared Camera Optics is August 4th*" or "*the vibration tests for the rocket boosters are complete*" can be stored by the assistant in a note-taking style. In a design meeting, SEVA would answer direct questions about the information that it has stored; such as "*When is the stress test for component A scheduled to be completed?*" or "*What is the development status of the flight Phase II software system?*". In problem-solving meetings, it would answer more complex questions such as "*What would the mass of the Spacecraft be if we removed the radar instrument?*" or "*Can titanium be used for soft x-ray shielding?*" Apart from meetings, SEVA can help SEs in their office work for recalling information from various engineering manuals or journals. It can assist them in conducting experiments by storing procedures, results, and analysis, provide design support by answering 'what-if' questions, and keep track of schedule and other time-related information.

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 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. The next type of questions (Recall) determine 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 MEB*[1] *?*" 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 some formal language 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. One form of reasoning is related to the idea of taxonomic or *is-a* relationship between two concepts [47]. 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 infer that {*Neon is odorless*}. SEVA should also have the ability to explain the steps by which it came to a derived information to the SE when queried. The term 'reasoning engine' and 'inference engine' is 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, *07/11/2016* 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 TRL* [2] *of instrument X, if it has been to space?*" is 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 in the question are unknown to SEVA** and 2) **entities are known but there is not enough information to answer the question**. In case one, 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 reply by asking "*What is Neon?*". Case two is a logical problem that applies to relational questions where the ontology is missing the predicate link. Assume 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

[1] Main Electronics Box [33]

[2] Technology Readiness Level: measures the maturity of a technology [29]

are two ways to provide the unknown information: 1) **make it derive the answer by creating a logical case through interaction** (eg: "*mass of an instrument is the sum of its components*") or 2) **give 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 should take 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 input**: Information given to the assistant to undergo natural language processing and subsequently added to its knowledge base
(2) **Commands**: Basic user commands to enter and exit hypothetical mode, undo an operation
(3) **Successful Querying**: Ask a question to the assistant and the assistant responds with an answer
(4) **Unsuccessful Querying due to Unknown Elements**: Unsuccessful querying problem is tackled by using interactive dialogues and Case-Based reasoning from Experience.

SYSTEM CONCEPT

Based on the defined operations concept, primary functional requirements were derived. SEVA has the capability 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) Hold a Dynamic Ontology
(8) Save specific interactions as 'Experience'
(9) Connect knowledge base with endowed models outside the system

SEVA has a component called *'Monitor'* which 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 1*.

Inter-module Communication Protocol (ICP) in *Figure 1* 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 would not know how to solve orbital mechanics problems or probability theory until **1)** an experience occurs, **2)** the user teaches it OR **3)** it endows the knowledge base with an orbital mechanics or probability module through Logicker. However, essential capabilities such as elementary math functions which includes addition, subtraction, and comparison need to be part of SEVA's fundamental capability.

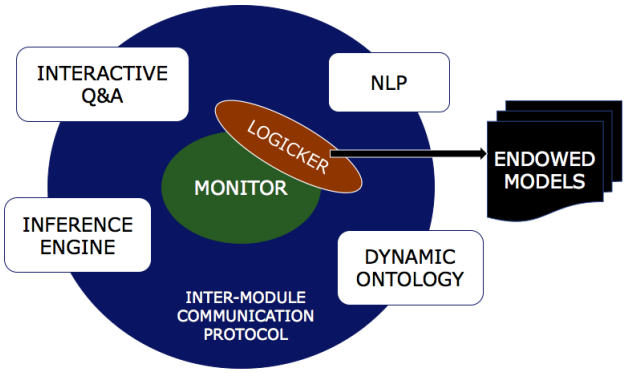

Figure 1: SEVA's Architectural components

More about Monitor, Logicker, and the endowed models are left as future work. Next section and *Figure 2* describes the architectural components and the functional diagram of SEVA's design.

KEY COMPONENTS

Natural Language Processing (NLP)

The NLP module converts NL text into information that can easily be represented in the form of an Ontology. The NLP functions are divided into 3 tasks: 1) determine the linguistic structure, 2) extract relations, and 3) extract rules.

Task 1 is to learn the linguistic (grammatical) structure of English. This involves syntactic or semantic parsing techniques and sub-tasks such as parts-of-speech tagging or semantic role labeling. The parse tree of a sentence can be represented as constituency-based or dependency-based. A constituency parser, such as [51], is used to extract sub-phrases from sentences while a dependency parser, such as Stanford Neural-network parser [6], is used to see the relation between words. Research focus of this step is to develop an unsupervised version of parsing to induce grammar. A semi-supervised version can be used to address domain specific idioms, common sense knowledge, and abbreviations. [22] and [44] are some relevant works that address unsupervised parsing and grammar induction.

Task 2 is the process by which relationships are created through triples or n-tuples. They allow for chunks of information to be represented and stored in an understandable and usable form. Triples are usually represented in [Subject - Predicate - Object] form. Extracting this predicate or relation between subjects and objects is the goal of Task 2. SEVA's design uses unsupervised shallow semantic parsing approach to the triple extraction problem such that the framework is domain independent. We propose an integrated heuristic/rule-based Information Extractor for SEVA using Open Information Extractors (OIE) addressed in works such as ClausIE

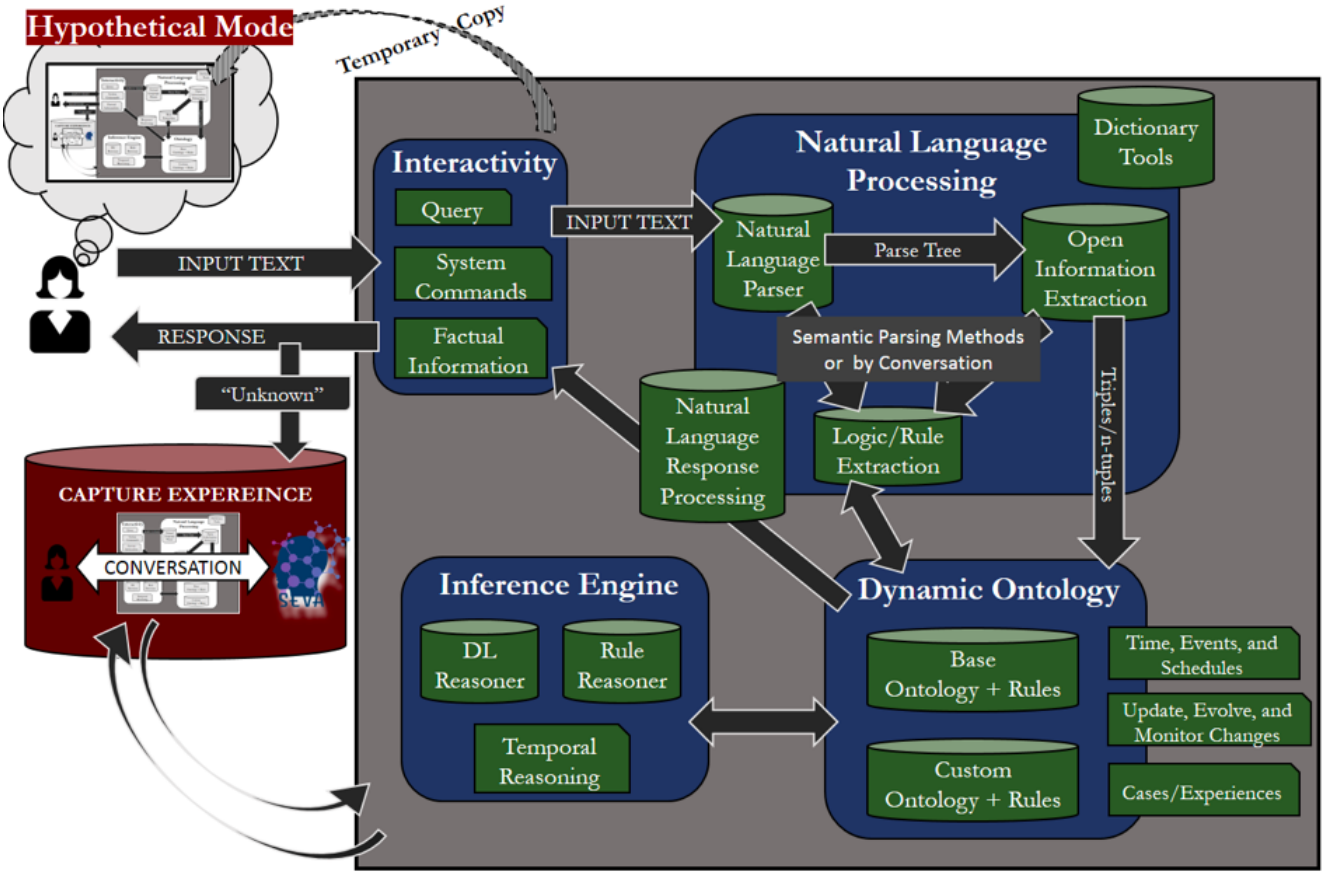

Figure 2: SEVA's Functional Diagram

[7], Open IE 4.0 [10], CSD-IE [3], and Stanford Open IE [2]. In this approach, extractors are learned by using hand-crafted rules or automatically created training data without using any annotated treebanks. These relationships can then be used to build SEVA's Ontology. Refer *Table 1* for extraction results from state-of-the-art OIE tools. ***Challenge:*** The extractor results shown in *Table 1* are promising. However, they are not fully accurate extractions of the input text. SEVA's goal is to solve this problem using a verb mediated heuristics/rule-based extraction improving upon the existing work.

Task 3 extracts rules in desirable forms such as horn clauses or description logic from text or triples. The goal is to assemble the grammatical structure from parsing, concepts, and relationships into clauses or rule-like structure. This is because 1) some relations needs to be more restricted or better formulated as rules, 2) rules fit better for Closed World Assumption scenarios, and 3) logic extraction from text requires unsupervised deep semantic parsing mentioned in works such as [20, 37, 40]. Example Text: Aerogel can capture a niacin molecule that has speed less than 5m/s. Example of a rule that is ideally extracted from the text:

$$hasSpeed(X, Y) \wedge (Y < 5m/s) \wedge NiacinMolecule(X)$$

$$\wedge\, Aerogel(Z) \Rightarrow canCapture(Z, X)$$

Dictionary/Thesaurus. Training the language parser with domain specific sentences will increase the accuracy. This is done to make the system familiar with domain specific idioms such as the usage of "heat pipe" or "cold case" by SEs. We can also create a domain specific or domain independent set of relations to relate synonyms and abbreviations to concepts or predicates. This will help in scenarios where the user asks the same question in different ways or the SE uses abbreviations frequently or a predicate is the same as another.

Knowledge Base (Ontology)

The Ontology module converts the output from the NLP module to represent information in the form of an Ontology. This work chooses a Semantic Web based approach to building SEVA's ontology. This provides extensibility and access to vast knowledge of information when needed in a format that SEVA, or specifically, the Logicker, will understand.

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 (latest version) built-in set of specifications called Resource Description Framework (RDF) [23]; essentially a set of RDF triples. We are interested in representing our information in an appropriately

Table 1: Outputs from Open Information Extractors

Input Sentence	Stanford Open IE	ClausIE	Open IE 4.0
If the instrument has been in space, then its TRL is 7	(instrument; has in; has space) (its TRL; is in; has space) (7; is in; has space)	(the instrument; has been; in space) (its; has; TRL) (its TRL; is ; 7 if the instrument has been in space then) (its TRL; is; 7)	(the instrument; has been; in space)
Aerogel can capture a Niacin molecule if the speed of the molecule is less than 5km/s	(Aerogel; can capture Niacin molecule; less than 5km/s) (speed; is; less) (speed; is less than; 5km/s) (Aerogel; can capture; Niacin molecule) (Aerogel; can capture Niacin molecule; less)	(Aerogel; can capture; a Niacin molecule if the speed of the molecule is less than 5km/s) (Aerogel; can capture; a Niacin molecule) (the speed of the molecule; is; less than 5km/s)	(Aerogel; can capture; a Niacin molecule) (the speed of the molecule; is; less than 5km/s)

expressive logic language. Thus we focus on Description Logic (DL), a decidable fragment of First-Order Logic, and rules which corresponds to OWL 2 DL and OWL 2 RL [32] respectively.

Description Logics (DLs). This section uses the ideas of DLs from [26] to incorporate into SEVA. Descriptions logics provide the formal semantics for designing ontologies in OWL. DLs consist of concepts, roles, and individuals. Concepts represent a set or a class. Examples are instruments, spacecrafts, and launch vehicles. Individuals are specific instances of a set such as SLS, which is a launch vehicle, or Mass Spectrometer, an instrument. Roles are relationships between concepts or individuals. Examples are *partOf* and *hasMass*. Their usage will be as *partOf(Thrusters, Spacecraft)* or *hasMass(DiscoveryShuttle, 75000kg)*. Readers familiar with Object Oriented Programming can relate this to objects, their properties, actions, and the concept of inheritance.

Based on these three components, knowledge in a DL language is categorized into three: Assertion Box (ABox), Terminological Box(TBox), and Relational Box (RBox). ABox contains assertions with individuals such *Spacecraft(DiscoveryShuttle)* implying that "Discovery Shuttle is an instance of Spacecraft" or *partOf(StarTracker, DiscoveryShuttle)* implying that "StarTracker is a part of Discovery Shuttle". TBox consists of relationships between concepts. For example, *Conduit* ≡ *Pipe* implying "same-as" relationship or *MassSpectrometer* ⊑ *Spectrometer* implying "sub-class" relationship. RBox consists of relationships between roles *partOf* ∘ *partOf* ⊑ *partOf* representing transitive property of the role. SEVA's design uses a popular fragment of DL called *SROIQ* [18] Ontology.

Hybrid Ontology. SEVA Ontology is divided into Base Ontology and Custom Ontology. Custom Ontology consists of domain specific concepts and instantiations, mostly the TBox and ABox axioms. Base Ontology contains the essential ingredients or building blocks of a domain independent initial ontology upon which a SE can build their customized ontology. It consists of the very basic axioms and rules that governs the ontology. It predominantly describes relationships and properties of predicates or RBox axioms. Some examples are *partOf* ∘ *partOf* ⊑ *partOf* representing transitive property and $partOf \equiv hasComponent^{-}$ describing inverse relationship. Some TBox and ABox axioms can optionally be stored in the Base Ontology if they are common sense knowledge, synonyms, or abbreviations. Some examples are *Conduit* ≡ *Tube* and *isAbbreviationFor(SLS, SpaceLaunchSystem)*. For a comprehensive knowledge in Synonyms or other English language structures, in addition to hand-crafted database, tools such as WordNet or DBpedia can be used. Over a period of usage, the Base Ontology by itself will constitute a fundamental domain independent and transferable relation-base for the SEs.

Challenge: Research focus of this step will be to automatically understand verbs and their properties, and connect them to other verbs in the base ontology using a dictionary or thesaurus tools. Assume the SE enters the sentence: *"X partOf Y"*. While the custom ontology is entered with the axiom *partOf(X, Y)*, the base ontology should be populated with the axiom *partOf* ∘ *partOf* ⊑ *partOf* implying transitive property. In addition, if *"hasComponent"* is present in the ontology, an inverse relationship has to be established between those two. Refer *Figure 3* for an example.

Rule-like Knowledge. Rule-like knowledge represents information in the form of horn clauses with a head and a body. Consider the example below:
can-Capture(X,Y) :-Aerogel(X), NiacinMolecule(Y), hasThickness(X,Z), greaterThan(Z,5cm), hasSpeed(Y,S), lessThan(S,10m/s).

The above rule says that an *Aerogel* with thickness greater than *5cm* can capture a Niacin molecule moving at speed less than *10m/s*. Although the same logic can be represented in Description Logic(DL), this requires NLP to convert text to DL logic which can demand deep semantic parsing approaches. SEVA's design uses rules as an addition to Description Logic. The task of 'Rule Extractor' component of NLP module is to extract rules from triples. This is also important in extracting information from nested triples that are dependent on each other. A simple example of this is an n-tuple shown below:
T1 : $[X,$ equals, Y, if **T2**$]$
T2 : $[Z,$ has value, $10]$

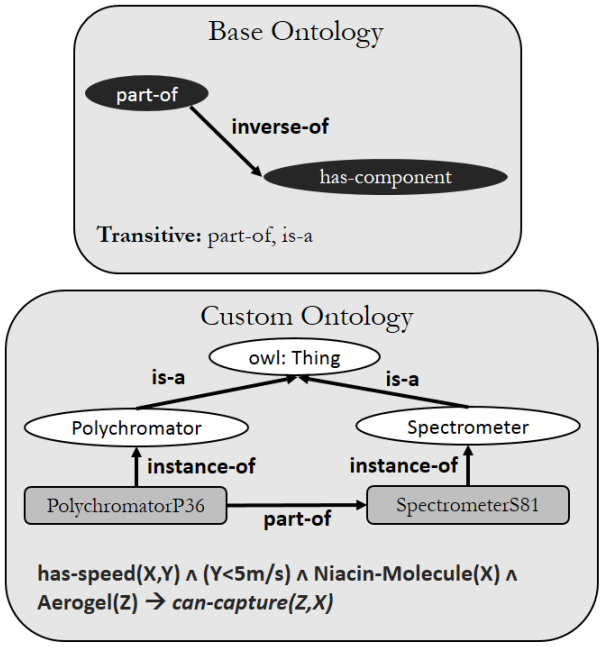

Figure 3: Hybrid Ontology Model

Another source of horn-like rules are 'experiences' from the interactive sessions. The user can teach SEVA rules to do a certain task through a conversation.

Open/Closed World Assumption (OWA/CWA). An open world does not assume complete information thus responding with an "I don't know" answer, while a closed world assumes that if something is not known to be true, then it is false. Formal definitions of these two assumptions can be found in [39]. 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", it is reasonable to assume they are not. This means that SEVA's knowledge base should support an open world assumption in general but certain predicates would be allowed to have closed world property. Works such as [8] and [24] study this type of integration. Base ontology in *Figure 3* 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 support CWA while Description Logic formalisms support OWA.

Dynamic Nature. SE deals with constantly changing information including time, events, and schedule. New information may need to be added or old information may need to be updated. Ontology consistency needs to be checked on each update. SEVA also needs to maintain versions of Ontology which are required to perform temporal reasoning, scheduling, and undo operation. Guidelines for a dynamic ontology is presented in works such as [35, 36, 45, 49, 50]. To aid user friendliness, SEVA's design follows that newer information overrides older conflicting information. The user will be notified only if the conflicting information contain a Base Ontology entity or the user has made some specifications. For example, the user is notified only if there is logic issue but not a mass update. Representing time in ontologies and intelligent systems have been studied by works such as [1, 4]. ***Challenge:*** There is little research available on knowledge bases requiring a dynamic ontology that can perform temporal reasoning on a hybrid representation and a hybrid world assumption.

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 [47]. 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 [30], ELLY [42], Jena [19], and Logic Programming approaches to RL using SWI-Prolog [47].
(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++ [46], RACERPRO [16], Pellet [43], and Hermit [41]. An example of this kind of implementation is DLEJena which uses Pellet for TBox reasoning and Apache Jena for ABox reasoning [31].
(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 [14].

Interactive Query-Answering

SEVA uses the knowledge-based paradigm [21] 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 [47] using the same OIE paradigm.

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 can be a) an answer which is contextually correct according to SEVA, b) a logical interaction to understand the missing link or predicate, or c) an interaction or instructions to the user to teach SEVA about a missing concept. Refer *Figure 4* for query and response types.

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 a 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 user. The following

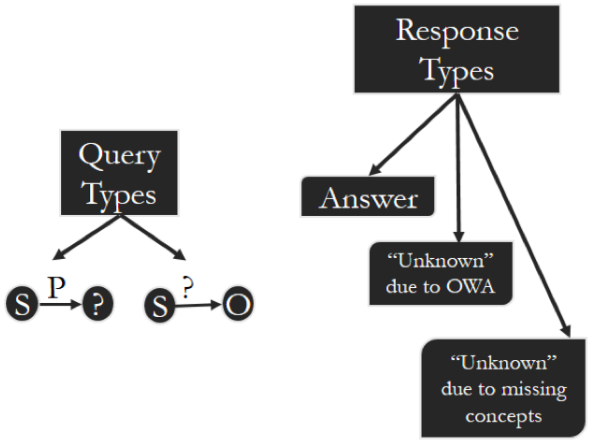

Figure 4: Queries and Responses
[S-P-O] represents [Subject - Predicate - Object] . OWA represents Open World Assumption

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: "Unknown"
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 their own rules. Creating contextual awareness is done through identifying structure of queries and applying case-based reasoning such as in [25] 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.

FUTURE WORK

This work is an introduction to SEVA's architectural design - the big picture, motivation, and its technical plausibility. Implementation and technical evaluation of each module, monitor, ICP, and computational complexity are currently in progress and left as future work.

Although SEVA is designed for a SE, the idea is extensible to all domains where personal assistants, that can grow alongside with the user, are needed. Thus, extending the idea to other domains is a future direction. Interconnectedness of different domains and different engineers makes a seemingly simple question "Is it possible to add one more instrument to the spacecraft?" challenging. We envision multiple SEVAs each belonging to a SE filling the gap of information assimilation and distributed coordination. Application of assistants in such collaborative settings is another area of future work.

CONCLUSION

This work introduced SEVA - a vision of a Systems Engineer's personal assistant. 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 bookkeeping and potentially error-prone information assimilation. SEVA is a trustable system with a domain independent 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, inferencing, learning, and query-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.

REFERENCES

[1] James F Allen. 1991. Time and time again: The many ways to represent time. *International Journal of Intelligent Systems* 6, 4 (1991), 341–355.
[2] Gabor Angeli, Melvin J. Premkumar, and Christopher D. Manning. 2015. Leveraging Linguistic Structure For Open Domain Information Extraction. *In Proceedings of the Association of Computational Linguistics (ACL)* (2015). https://doi.org/10.3115/v1/P15-1034
[3] Hannah Bast and Elmar Haussmann. 2013. Open Information Extraction via Contextual Sentence Decomposition. *IEEE Seventh International Conference on Semantic Computing* (2013), 154–159.
[4] Sotiris Batsakis, Kostas Stravoskoufos, and Euripides GM Petrakis. 2011. Temporal reasoning for supporting temporal queries in OWL 2.0. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*. Springer, 558–567.
[5] Rodney A Brooks. 1997. The intelligent room project. In *Cognitive Technology, 1997. Humanizing the Information Age. Proceedings., Second International Conference on*. IEEE, 271–278.
[6] Danqi Chen and Christopher D Manning. 2014. A Fast and Accurate Dependency Parser using Neural Networks. *Proceedings of EMNLP* (2014). https://doi.org/10.3115/v1/D14-1082
[7] Luciano Del Corro and Rainer Gemulla. 2013. Open Information Extraction via Contextual Sentence Decomposition. *WWW Proceedings of the 22nd international conference on World Wide Web* (2013), 355–366. https://doi.org/10.1145/2488388.2488420
[8] Carlos Viegas Damásio, Anastasia Analyti, Grigoris Antoniou, and Gerd Wagner. 2006. Supporting open and closed world reasoning on the web. In *PPSWR*, Vol. 4187. Springer, 149–163.
[9] Pedro Domingos. 2015. *The master algorithm: How the quest for the ultimate learning machine will remake our world.* Basic Books.
[10] Oren Etzioniz, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam. 2011. Open Information Extraction: the Second Generation. *IJCAI Proceedings of the Twenty-Second international joint conference on Artificial Intelligence* 1 (2011), 3–10. https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-012
[11] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. 2010. Building Watson: An overview of the DeepQA project. *AI magazine* 31, 3 (2010), 59–79.
[12] Elyse Fosse. 2014. Model-based Systems Engineering (MBSE) 101. https://trs.jpl.nasa.gov/bitstream/handle/2014/43989/13-0392_A1b.pdf?sequence=1. (2014).

Accessed: 11-09-2017.
[13] Elyse Fosse, Corey Harmon, Mallory Lefland, Robert Castillo, and Ann Devereaux. 2014. Inheriting Curiosity: Leveraging MBSE to Build Mars2020. https://trs.jpl.nasa.gov/bitstream/handle/2014/45871/15-3486_A1b.pdf?sequence=1. (2014). Accessed: 11-09-2017.
[14] Ernest Friedman-Hill et al. 2008. Jess, the rule engine for the java platform. (2008).
[15] Thomas R Gruber. 2009. Siri, a Virtual Personal Assistantfi!?Bringing Intelligence to the Interface. (2009).
[16] Volker Haarslev, Kay Hidde, Ralf Möller, and Michael Wessel. 2012. The RacerPro knowledge representation and reasoning system. *Semantic Web* 3, 3 (2012), 267–277.
[17] Laura E. Hart. 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. (2015). Accessed: 11-09-2017.
[18] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. 2006. The Even More Irresistible SROIQ. (2006).
[19] Apache Jena. 2014. the Apache Jena project. (2014).
[20] Rahul Jha and Catherine Finegan-Dollak. 2015. Education Professional Experience Publications Rahul Jha and Dragomir R. Radev. an Unsupervised Method for Learning Probabilistic First Order Logic Models from Unstructured Clinical Text. in Icml Workshop on Learning from Unstructured Clinical Text, 2011.
[21] Daniel Jurafsky and James H. Martin. 2017. Chapter 28 Question Answering. (2017). https://web.stanford.edu/~jurafsky/slp3/28.pdf
[22] Dan Klein. 2005. *The Unsupervised Learning of Natural Language Structure.* Ph.D. Dissertation. Stanford University.
[23] G Klyne and J. J Carroll. 2003. Resource Description Framework (RDF): Concepts and Abstract Syntax. http://www.w3.org/TR/2003/WD-rdf-concepts-20030123. (2003). Accessed: 08-03-2017.
[24] Matthias Knorr. 2011. Combining open and closed world reasoning for the semantic web. (2011).
[25] Janet L Kolodner. 1992. An introduction to case-based reasoning. *Artificial intelligence review* 6, 1 (1992), 3–34.
[26] Markus Krötzsch, Frantisek Simancik, and Ian Horrocks. 2012. A description logic primer. *arXiv preprint arXiv:1201.4089* (2012).
[27] Adam Lally. 2017. IBM Watson: Beyond Jeopardy! Q&A. https://learning.acm.org/webinar/lally.cfm. (2017). Accessed: 11-12-2017.
[28] Jangho Lee, Gyuwan Kim, Jaeyoon Yoo, Changwoo Jung, Minseok Kim, and Sungroh Yoon. 2016. Training IBM Watson using Automatically Generated Question-Answer Pairs. *CoRR* abs/1611.03932 (2016). arXiv:1611.03932 http://arxiv.org/abs/1611.03932
[29] Thuy Mai. 2012. Technology Readiness Level. https://www.nasa.gov/directorates/heo/scan/engineering/technology/txt_accordion1.html. (2012). Accessed: 07-07-2017.
[30] C Matheus, B Dionne, D Parent, Kenneth Baclawski, and M Kokar. [n. d.]. BaseVISor: A forward-chaining inference engine optimized for RDF/OWL triples.
[31] Georgios Meditskos and Nick Bassiliades. 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 (2010), 89–94.
[32] Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, Carsten Lutz, et al. 2009. OWL 2 web ontology language profiles. *W3C recommendation* 27 (2009), 61.
[33] Phil Newman. 2012. Acronyms and Glossary. https://asd.gsfc.nasa.gov/archive/hubble/overview/glossary.html. (2012). Accessed: 11-11-2017.
[34] Kathy Panton, Cynthia Matuszek, Douglas Lenat, Dave Schneider, Michael Witbrock, Nick Siegel, and Blake Shepard. 2006. *Common Sense Reasoning – From Cyc to Intelligent Assistant.* Springer Berlin Heidelberg, Berlin, Heidelberg, 1–31. https://doi.org/10.1007/11825890_1
[35] Perrine Pittet, Christophe Nicolle, and Christophe Cruz. 2012. Guidelines for a dynamic ontology-integrating tools of evolution and versioning in ontology. *arXiv preprint arXiv:1208.1750* (2012).
[36] Peter Plessers, Olga De Troyer, and Sven Casteleyn. 2007. Understanding ontology evolution: A change detection approach. *Web Semantics: Science, Services and Agents on the World Wide Web* 5, 1 (2007), 39–49.
[37] Hoifung Poon and Pedro Domingos. 2009. Unsupervised semantic parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1.* Association for Computational Linguistics, 1–10.
[38] Charles Rich and Richard C Waters. 1987. The Programmerfis Apprentice project: A research overview. (1987).
[39] Stuart J. Russell and Peter Norvig. 2003. *Artificial Intelligence: A Modern Approach* (2 ed.). Pearson Education.
[40] Stefan Schoenmackers, Oren Etzioni, Daniel S. Weld, and Jesse Davis. 2010. Learning First-order Horn Clauses from Web Text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP '10).* Association for Computational Linguistics, Stroudsburg, PA, USA, 1088–1098. http://dl.acm.org/citation.cfm?id=1870658.1870764
[41] Rob Shearer, Boris Motik, and Ian Horrocks. 2008. HermiT: A Highly-Efficient OWL Reasoner.. In *OWLED*, Vol. 432. 91.
[42] Katharina Siorpaes and Daniel Winkler. [n. d.]. An ELP Reasoner. ([n. d.]).
[43] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. 2007. Pellet: A practical owl-dl reasoner. *Web Semantics: science, services and agents on the World Wide Web* 5, 2 (2007), 51–53.
[44] Valentin I. Spitkovsky. 2013. *Grammar Induction and Parsing with Dependency-and-Boundary Models.* Ph.D. Dissertation. Stanford University.
[45] Ljiljana Stojanovic, Alexander Maedche, Boris Motik, and Nenad Stojanovic. 2002. User-driven ontology evolution management. *Knowledge engineering and knowledge management: ontologies and the semantic web* (2002), 133–140.
[46] Dmitry Tsarkov and Ian Horrocks. 2006. FaCT++ description logic reasoner: System description. *Automated reasoning* (2006), 292–297.
[47] Christopher Walton. 2007. *Agency and the semantic web.* Oxford University Press on Demand.
[48] wikia.com. [n. d.]. J.A.R.V.I.S. http://marvel-movies.wikia.com/wiki/J.A.R.V.I.S.. ([n. d.]). Accessed: 11-08-2017.
[49] Fouad Zablith. 2008. Dynamic ontology evolution. (2008).
[50] Fouad Zablith, Grigoris Antoniou, Mathieu d'Aquin, Giorgos Flouris, Haridimos Kondylakis, Enrico Motta, Dimitris Plexousakis, and Marta Sabou. 2015. Ontology evolution: a process-centric survey. *The knowledge engineering review* 30, 1 (2015), 45–75.
[51] Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and Accurate Shift-Reduce Constituent Parsing. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics* (2013), 434–443. https://doi.org/10.13140/2.1.2791.2961