D. Natolana Ganapathy and K. P. Joshi, "A Semantically Rich Framework to Automate Cloud Service Level Agreements," in *IEEE Transactions on Services Computing*, doi: 10.1109/TSC.2022.3140585.

https://doi.org/10.1109/TSC.2022.3140585

Preprint: IEEE Transactions on Services Computing, page(s): 1-12, ISSN: 1939-1374, DOI: 10.1109/TSC.2022.3140585

A Semantically Rich Framework to Automate Cloud Service Level Agreements

Divya Natolana Ganapathy
divya.natolana@gmail.com

Dr. Karuna Pande Joshi, Senior Member IEEE
karuna.joshi@umbc.edu

Abstract—Consumers evaluate and choose cloud-based services based on the Service Level Agreements (SLA). These agreements list the service terms and metrics to be agreed upon by the service providers and the customers. Current cloud SLAs are text documents that require significant manual effort to parse and determine if providers meet the SLAs. Moreover, due to the lack of standardization, providers differ in the way they define the terms and metrics, making it more difficult to compare different provider SLAs. We have developed a novel framework to significantly automate the process of extracting knowledge embedded in cloud SLAs and representing it in a semantically rich knowledge graph helping the user to make a calculated decision in choosing a provider. Our framework captures the key terms, measures, and deontic rules, in the form of obligations and permissions present in the cloud SLAs. In this paper, we discuss our framework, technique, and challenges in automating the cloud services agreement. We also describe our results and their validation against well-established standards.

Index Terms—Service Level Agreements, Semantic Cloud Services, Semantic Web, Deontic Logic, Actor determination, Natural Language Processing, and Text Mining.

1 Introduction

Organizations are increasingly migrating to Cloud-based services to take advantage of Cloud features like rapid provisioning, scalability, ease of use, cost savings, high availability, and platform independence. Cloud service is typically accompanied by a service level agreement (SLA) which defines the minimal guarantees that a provider offers to its customers [1]. Components of a service could be procured from multiple service providers; there could be primary and secondary service providers, who will need to agree on the service terms and conditions, thereby adding to the complexity of the SLAs. The SLA specifies service performance metrics like

- Availability timeframe of services,
- Scheduled maintenance times,
- Contingency or business continuity plans,
- Timeframes for notification and recovery following an unplanned service disruption or a security incident,
- Problem resolution and escalation procedures, etc.

Cloud service agreements are currently text documents and have to be manually parsed by the consumers to determine which service best suits their needs, this process is very time consuming and prone to errors. This makes consumers dependent on the cloud service provider's performance reports to gauge the value of the cloud service. Moreover, cloud contracts contain rules and policies that are not fully encapsulated by existing performance metrics that cloud providers track. There are no standard definitions for cloud metrics, different providers could have different definitions for the same measure. One of the large federal agencies found that their cloud service provider was tracking the service availability measure as only system availability and not user access. The provider insisted that their service was available and meeting SLA requirements given that it was 'up and running' even if not every valid end user could access it at the same time. As a result, the federal agency had to update the service contract to redefine 'service availability', the provider also had to modify the business process to track availability and user access statistics. Such situations can be overcome by using a semantic framework like the one we have built to analyze the service contract. A query to the SLA knowledge graph would help in a clear understanding of key terms and also help to compare these terms across different providers. The agency would then be able to pick another provider with the required services or ask for amends in the current contract.

As SLA standards are still in the nascent stage, different cloud providers construct their own rules and policies for the service offering and define them as clauses in the cloud legal document. The complexity increases when the documents are written by a third-party service provider.

The Research challenge in this problem is the lack of standardization, due to the lack of standards for cloud service performance, providers often construct their own rules for performance measures and metrics. They define them as 'clauses' in the cloud legal documents, such as TOS, SLAs, or privacy policy documents, that are part of the cloud contract. Reviewing all these cloud legal documents to ensure the cloud service is meeting the organizational requirements is a labor and time-intensive endeavor for consumers. Also, is often an afterthought when a cloud service fails to live up to its expectations. A critical step

- *D. N. Ganapathy is with the Computer Science and Electrical Engineering Department, University of Maryland, Baltimore County, Baltimore, MD, 212250.*
- *K. Joshi is with the Information Systems Department, University of Maryland, Baltimore County, Baltimore, MD, 21250.*

in automating cloud service selection and management is to make the cloud SLAs machine-understandable so that selection tools can interpret the policy rules and metrics defined in the service contracts. Another research challenge is co-referencing/cross-referencing, many legal documents tend to reference other documents or different sections within the document. There is a need to develop techniques to represent and reason over multiple legal documents that co-reference/cross-reference another set of documents and/or sections within the document.

There needs to be a common platform with a common understanding through which information exchange and querying can be done. In this work, we have built a novel framework to enable users to pick the preferred cloud service provider from a selection of providers.This is done by building a base ontology with classes like cloudSLA, Deontic statements etc using key term extraction and NIST standards. Once the ontology is in place, various techniques are used to populate this ontology; the measure of suitability and completeness of a provider's SLA is based on the population count of this ontology. The relationships are created using modal logic and dependency parsing. A combination of text mining and semantic web technologies were incorporated to extract various components of the SLA that work independently to extract components of NIST, making it a self-evaluating model adding to the novelty of the approach. Techniques used to extract different components are further shown in figure 2. Table extractions to obtain remedies for non -compliance. Definition extractor to extract standard definitions, deontic extraction to extract roles and responsibilities and using modal and logic and dependency parsing and similar context words extracted using skip gram. Once the required components are extracted they are auto-populated onto the knowledge graph which can then be queried using a SPARQL query. We have also implemented novel ways of evaluation by implementing a multi-layered evaluation system starting with knowledge graph population count evaluation to evaluate the completeness of each provider. Followed by actor evaluation by verifying the actors extracted with manual actor extraction Finally evaluation of results extracted by SPARQL query These various techniques and the engineering of their combination and the implementation speaks for the novelty of this research.

In this paper, we initially discuss the background and related work in this area. Then we present our approach towards automating service level agreements and describe the knowledge graph we have developed for the same. We present the results of our analysis in the section Results and Validation and end with conclusions and future work.

2 Related Work

2.1 SLA Standards

Baset [1] describes an SLA as consisting of a service guarantee that specifies the metrics which a provider strives to meet over a service guarantee time period. Failure to achieve those metrics will result in a service credit to the customer. Availability, incident response time and recovery, and fault resolution time are examples of service guarantees. Service guarantee granularity describes the resource scale on which a provider specifies a service guarantee. For example, the granularity can be on a per service, per data center, per instance, or per transaction basis. SLA also specifies the service guarantee exclusions. Service violation measurement and reporting describe how and who measures and reports the violation of service guarantee, respectively. In recent years, international bodies have been working towards defining standard definitions and terminologies for a Cloud SLA. The International Organization for Standardization's ISO/IEC DIS 19086 standard [2], European Commission's "Cloud Service Level Agreement Standardization Guidelines" and NIST's Special Publication 500-307 on Cloud Computing Service Metrics Description [3], are some of the publications that when finalized will help consumers mandate a fixed SLA format from various cloud providers. That will also help consumers compare and contrast the various cloud services based on their terms of service/SLA metrics. The committee responsible for the standardization and documentation of Service Level Agreements is ISO/IEC JTC 1, Information technology, Subcommittee SC 38, Cloud computing, and distributed platforms. The overview, fundamental concepts, and definitions for cloud SLA is in ISO/IEC 19086 which builds on the cloud computing concepts defined in ISO/IEC 17788 and ISO/IEC 17789.ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization [4]. There have been efforts made to build a Unified Service Description Language (USDL) [5], Cloud Service Description Model (CSDM) [6], and rSLA [7] to describe the features of cloud services and share across web. However, these are not based on the already existing standards and are building different standards on their own. hlWe overcome this in our framework by centering the whole experiment around NIST standards [8] making the base schema for knowledge graph, NIST specified. In this work we extract knowledge from various service providers and compare them against a common NIST standard, giving it a very well defined structure. This is done by first building the basic schema of the ontology with classes and relationships using NIST standards and then populating this ontology classes and relationships with information extracted from the different providers. This is done to build a standard, one for all solution for service level agreement extraction and evaluation while making sure that simplicity and ease of usage are maintained. We combine the aspects of standardization and generalization of SLA by creating a semantically rich system. According to ISO/IEC 19086 the cloud SLA should consist of the key characteristics of a cloud computing service and should provide a common understanding between cloud service consumers and cloud service providers. A cloud SLA framework should consist of Cloud Service Agreement (CSA), Cloud Service Level Agreement (SLA), Cloud Service Level Objectives (SLO), and Cloud Service Qualitative Objectives (SQO) [3].

Tasks in analyzing service level agreements:

1) Comprehending Cloud service agreements: Once a cloud service consumer has chosen a cloud service after analyzing the service characteristics and if its requirements are met, then this information is placed in the cloud SLA. Cloud SLA has Cloud service objectives. These objectives give a value to a cloud service char-

acteristic that is understood to have a meaning and a specific level of it needs to be met by the cloud service. A metric does not give specific values but gives a few rules on how to measure a particular characteristic of cloud service.

2) Metrics Measurement: The document gives a model for representing the metrics in a specific way so that measurement tools can be devised. The measured value can be compared to cloud service objective commitments made in the cloud service agreements.
3) Verification of cloud SLA: As the measurements are made with the same metrics that were used to define the characteristics of a service in the cloud SLA, it is possible to compare the measurement result to the cloud service objective commitment. If the values do not match then a remedy will be sought out.

2.2 Semantic Web

In cloud-based service environments, there is an exchange of information, queries, and requests between the user and the cloud service provider. This information exchange could be for data or policies followed by the cloud service providers. The handling of heterogeneous policies is usually not present in a closed and/or centralized environment but is an issue in the open cloud. The inter-operability requirement is not just for the data itself, but even for describing services, their service level agreements, quality-related measures, and their policies for sharing data.

One way to solve this would be to use semantic web techniques for modeling service related information. We have used this to automate the selection of cloud service provider [9] [10] [11]. The data is annotated with machine-understandable meta-data, which could be queried and used in the correct context in an automated manner. Semantic web technology uses simple languages RDF [12] (Resource Description Language) and OWL [13] for describing the objects and relations, define ontologies, and describing the meta-data using these ontologies. Our framework uses the novel approach of combining the semantic knowledge extracted using the neural network skip gram model to get semantically similar terms in different providers to populate classes set up on NIST standards.

Another approach uses Web Services Definition Language (WSDL) [14] which is a W3C standard for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. However, WSDL does not allow a means to express the policies that the service supports or adheres to. Hence additional proposals like WS-Policy and WSLA have been made to allow for the expression of additional nonfunctional attributes. Turner et al. [15] have proposed a service technology layer for the creation and deployment of web services. They have compared the existing protocols and technology available to implement web services and have also noted gaps that need to be researched.

Oldham et al. [16] have proposed semantically rich extensions to extend the WS-Agreement. Aiello et al. [17] have proposed a formal definition of 'Agreement' based on WS-Agreement. WS-Negotiation is proposed as an extension of WS-Agreement to allow negotiation capabilities. WS-Negotiation consists of three parts - negotiation message, negotiation protocol, and negotiation decision. In this basic model for WS-Negotiation, the emphasis is on how to negotiate and what to negotiate. It does not demonstrate how a requester's enterprise policies can be used to automate the negotiation process. Bui and Gachet [18] have described a broker capable of offering negotiation and bargaining support to facilitate the matching of web services. The negotiation protocol and decision making mechanisms for negotiation have not been described. Our proposed knowledge graph is flexible enough to capture knowledge in terms of existing and future SLA standards. This knowledge graph allows for a clear comparison of different providers' language, usage of terms, the value of performance by simple queries helping the customer make a strategic decision.

2.3 Text Extraction

Researchers have applied Natural Language Processing (NLP) techniques to extract information from text documents. In Rusu et. al. [19] the authors suggest an approach to extract subject-predicate-object triplets. They generate Parse Trees from English sentences and extract triplets from the parse trees. Etzioni et al. [20] developed the KNOWITALL system to automate the process of extracting large collections of facts from the web in an unsupervised, domain-independent, and scalable manner. Etzioni et al. [20] used Pattern Learning to address this challenge.

Natural language technique uses Noun Phrase extraction for information extraction to create triplets by considering 'Noun Phrase' which would be obtained from part-of-speech taggers. Barker et al. [21] extract key-phrases from documents and show that the noun phrase-based system performs roughly as well as a state-of-the-art, corpus-trained key-phrase extractor.In this paper, we have taken this approach a few steps further by not only extracting 'Noun Phrase' and subject-predicate-object triplets to create classes and relationships but also to get deeper into the structure of phrases. This helps to understand the dependency of the terms in a phrase, determine the type of the statements and who the actors are in those statements, and also determines which party, the service provider, or the customer gets directly affected by the phrases. This information could strongly affect the service contract negotiation which is explained later in the paper.

Documents consist not only of a huge number of unstructured texts but also a vast amount of valuable structured data in the form of tables. Extracting knowledge from structured tables is an ongoing research problem with multiple solutions proposed to handle both general and domain-specific tables. Mulwad et al. [22] proposed a framework that assigns a class to table columns and links table cells to entities and inferred relations between columns to properties. Bhagavatula et al. [23] created a system called TabEL which works by extracting content using entity linking. We use a modified version of it to improve our knowledge extraction system. Researchers have explored the automated techniques for extracting permissions and obligations from legal documents using text mining and semantic techniques, Kagal et al. [24] formed, proposed a semantic web-based

policy framework to model conversation specifications and policies using obligations and permissions. In this paper, we have created a comprehensive framework for knowledge extraction from SLAs by combining and improving the techniques mentioned above. Along with which we have created new techniques, piecing out the different components of the SLA into the most granular machine-readable format. We use 'Noun Phrase' and 'subject-predicate-object' for Key Term extraction and Definitions, We discovered the usage of Table extraction for getting the Remedy for non-compliance, we built neural networks for extracting the Period of performance terms written in different forms. In addition to extracting direct and indirect actors from clear and ambiguous obligation and permission statements by identifying the term dependency in phrases.

In our previous work, we have used topic modeling to extract key terms with moderate results. We intend on extending it to small text topic modeling as the traditional methods do not yield great results because of the size of the SLA. We have also described a preliminary knowledge extraction system for cloud SLA documents. In this paper, we extend our framework to include extraction of information from tables and rules in the form of obligations and permissions from cloud SLA documents. which is then automatically populated to the knowledge graph and queried later for results.

3 Knowledge graph for Cloud Service Level Agreement

3.1 Introduction

The cloud legal documents fall into three broad categories. Firstly, the Service Contract documents, which describes the rules and clauses that specify service functionality, quality and performance metrics. Secondly, the Privacy and Security Data Document, that describes the data security and privacy policies for the service and thirdly, the Regulatory Compliance documents that specifies the set of rules, policies or standards formulated by regulatory agencies or standards organizations. The main goal is to make the SLA machine-readable. To achieve this, we collect the key terms and relationships extracted using various techniques, from the SLA, which is used to develop a semantically rich knowledge graph. The knowledge graph is based on knowledge representation language, OWL (Web knowledge graph Language). The knowledge graph could be queried using SPARQL to obtain the stored relationship. This base of the knowledge graph was built by the combination of two tasks, Key-term and relationship extraction from the text and matching the obtained terms with NIST [3] suggested standards. According to [8] the figure 1 shows that the cloud service level agreements should contain, standards, definitions, roles and responsibilities, period of performance, etc. The knowledge graph we build would contain classes and relationships as mentioned by NIST [8].

3.2 Classes

1) ***Class: CloudSLA***: This is the main class containing subclasses like service name, service description, service delivery, etc.

Agreements, regardless of type, should specify the following:

- Explicit definitions of both the organization's roles and responsibilities and the service provider's roles and responsibilities (including level of clearance or background investigation needed for staff)
- Description of the service environment, including locations, facility security requirements, and policies, procedures, and standards; and, agreements and licenses
- Defined service levels and service level costs. The service level section of the service agreement may stipulate various service levels for different types of customers or price levels and it may stipulate different service levels for various periods of performance, e.g., year 1 may demand a higher service level than year 2 of the contract.
- Defined process regarding how the managers will assess the service provider's compliance with the service level and due date targets, rules, and other terms of the agreement
- Specific remedies (e.g., financial, legal) for noncompliance or harm caused by the service provider
- Period of performance and/or deliverable due dates

Fig. 1. NIST Guidelines for SLA [3]

2) ***Class:DeonticStatements*** :The class contains subclasses "Permission", and "Obligation", . The SLA statements that indicate permission are the DataProperty values of the class Permission. The statements which are Obligation statements in the SLA are the DataProperty values of the class Obligation and so on.
3) ***Class: ServiceStandards*** : The class ServiceStandards has the standards as defined by the service provider. These are the terms that are mentioned with or without the title 'Definitions' in the SLA document. The data property of this class has frequently used terms and their definitions.
4) ***Class:TermsandAdjustments*** : This class has subclasses like 'ServiceCost', 'PeriodofPerformance' and 'RemedyforNonCompliance'. The class 'RemedyforNonCompliance' contains the values of credits assured to the consumer by the service provider if the service provider is unable to provide the promised service.

3.3 Relationship

Relationships in the knowledge graph are the relationships between two classes which is called the ObjectProperty or the relationship between a class or an instance of a class and value of that class or instance called the DataProperty.

1) ***hasDeonticStatements***: This DataProperty represents the relationship between various Service Level Agreements and their deontic statements.
2) ***hasObligationStatements***: This relationship relates the instances of class Obligation to their SLA obligation statements.
3) ***hasPermissionStatements***: This relationship relates the instances of class Permission to their SLA permission statements.
4) ***hasActors***: This relationship relates the instances of class Obligation and Permission to the actors in the Obligation and Permission statements.
5) ***hasRemedyforNonCompliance***: This relates the instances of class RemedyforNonCompliance with the values of credits returned in case of non-compliance.
6) ***hasServiceStandards***: This relates the instances of class ServiceStandards with the standardized terms as defined by the SLA.

4 Methodology

Through this research, we aim to enable the consumer to select from multiple cloud services. The architectural elements

and components of the framework are as shown in figure 2. We start by web scraping various cloud providers' SLA documents. On top of this text mining and semantic web techniques are applied. Once the text prepossessing is done, different components of the framework work independently to extract different components from the SLA as suggested by NIST [8]. We have the Key Term Extractor component to extract classes and relations which is then compared with NIST standards, then we have the Table Extraction to extract Remedies for non-compliance, the Skip Gram Model to extract Period of Performance, Definition Extractor to extract the standards, followed by Deontic Extractor to extract the Roles and Responsibilities. Once these are extracted they are populated into the classes and sub classes in the knowledge graph, which is then queried to answer various user requirements helping them to select the best SLA.

4.1 Data-set Used for the Experiment

Data used is from publicly available Service Level Agreement documents of popular cloud service providers like VMware vCloud Air [25], Amazon Web Services(AWS) [26], Microsoft Azure cloud [27], Google Cloud Platform [28] and IBM Softlayer [29] was used to build and train the framework. After the model was built it is tested for correctness using on Rackspace [30], SAP [31], Alibaba [32] and Oracle [33]. Most of these are compute SLA, however, the framework works fairly well even on other SLA which follows the NIST guidelines and has arranged the documents according to the extractor in the framework.

4.2 Text Extraction

The text from the SLA documents are extracted using multiple techniques based on their application (here, the class that they would get populated into). Below is the detailed component-wise extraction technique.

4.2.1 Definition Extraction/ Standards Extraction

Legal documents define the terms and standards as established that are repeatedly used in them. This is done to achieve clarity without repetition. Some rules that are followed are that the legal terms should not be defined if used just once or if they conflict with the accepted meaning. Also, they should be placed where they are most easily found, preferably quoted before they are used. SLAs contain definitions related to the provider's operation, the framework helps in extracting those terms making it easier for decision making.

To extract these definitions we tokenize the document into sentences, these sentences are parsed using the CMU Link parser [34] and regular expression. Here the sentences would be split into noun phrase X and verb phrase Y similar to Noun-phrase in [35]. Y could even be a complex sentence on its own. X and Y are connected by a few filler words like 'is', 'means' etc. Pattern matching is used to match the sentences with this required pattern. If a sentence matches this pattern it gets picked up by the extractor, an example sentence would be *"A "Service Credit" is a dollar credit, calculated as set forth above, that we may credit back to an eligible account"*. Here in this sentence *'Service credit'* would be the X in the sentence and *'a dollar credit, calculated as set forth above, that we may credit back to an eligible account'* would be the Y. We then add this sentence to the RDF with the obtained relation between them, which is that X is a key term that is defined as Y. This forms the dataProperty value of the class 'Standards' as these are the standards set by the SLAs.

4.2.2 Extraction of Remedies for Non-Compliance

On various occasions, the cloud service providers fail to provide the promised service duration or uptime. A good SLA should have the means to deal with this non-compliance. The Service provider should have clear information on how it would provide a remedy for this situation by giving back credits or extra service or some other remedy useful to the consumer. For example, AWS in their [26] as shown in figure 3 specifies the credits that would be given back to the consumer in case AWS fails to provide the promised service. This high-quality information is usually within a table. This is very convenient for humans to understand. Unfortunately, its equally hard for it to be machine-understandable. To deal with this we use the BeautifulSoup library in python. BeautifulSoup is a python library for pulling data out of HTML and XML files. It works with your favorite parser to provide various ways of navigating, searching, and modifying the parse tree. We extract the table by extracting contents under the table tag in HTML. After this parsing of the table, we end up with a dictionary of elements which is the contents of the table cells. To keep the framework more generic we have used the approach of extracting remedies for non-compliance only from tables. The remedies for non-compliance class is not populated for a few SLAs in the knowledge graph, that does not mean that they do not have remedies of non-compliance, it only means that they do not have in a clear table format which is followed by several popular cloud provider SLA.

This extracted data is encoded as RDF statements. The RDF statements are then added to the knowledge base. These details are populated as the dataProperty of the class 'RemedyforNonCompliance', which is a subclass of class 'TermsandAdjustments'. Once we get the dictionary with data in it, the table headers are used to associate semantics to different cell data. We strove to keep the system reusable by not using any data or documents from a local copy but instead use real-time hyperlinks to service providers. This is done so that the same prototype could be reused even when the providers change their SLA content as long as the update is on the same link.

4.2.3 Extraction of Word Context / Skip-gram model

While dealing with a text document like the SLA, we often come across different terms being used to refer to the same concept across different service providers. It is relatively easy for humans to understand that the term 'availability' used by AWS is the same as term 'uptime' used by GCP. However, it is not that trivial for the machine to understand this. To make the machine understand these words as similar words, we build a 3 layer neural network, which uses skip-gram model as shown in figure 4 to understand the context of the words. The input to this model is one-hot encoded-words from the SLA. The output layer gives the probability of the label, here the neighboring words or the context words. Cross entropy(loss-function) is applied to the

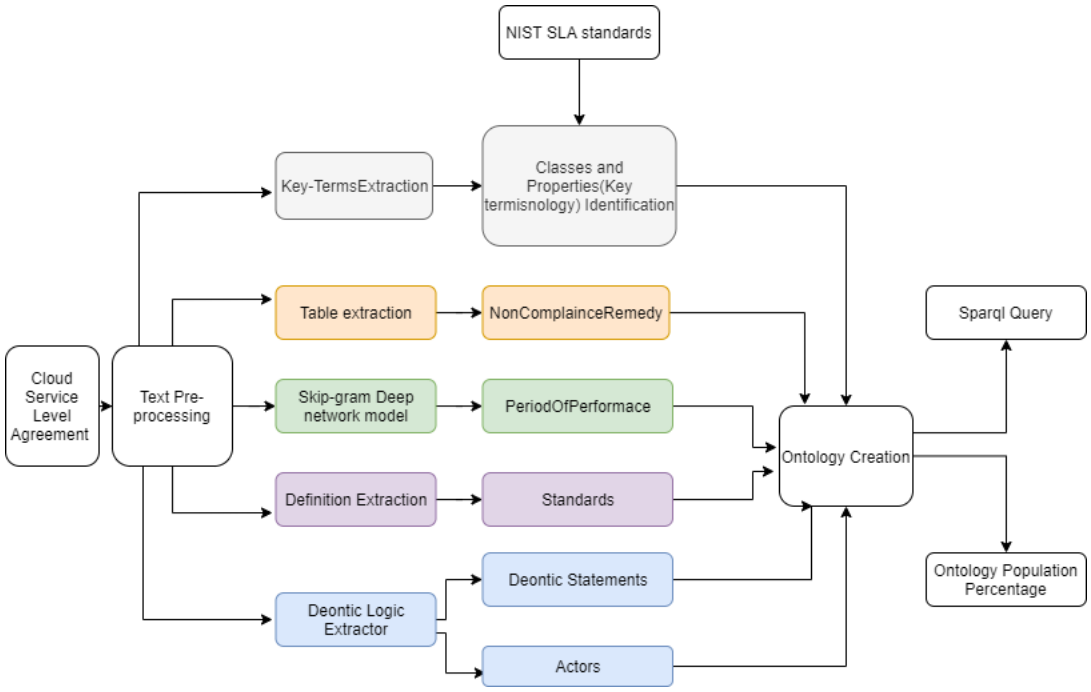

Fig. 2. Complete Architecture of Framework to extract knowledge from cloud SLA

Monthly Uptime Percentage	Service Credit Percentage
Less than 99.99% but equal to or greater than 99.0%	10%
Less than 99.0% but equal to or greater than 95.0%	30%
Less than 95.0%	100%

Fig. 3. Remedy for Non-Compliance Table in AWS [26]

output to get the one-hot encoding values of predicted context words. The hidden layer in the neural network model gives the n-dimensional word embedding of every word (output of interest in this application). The error function is minimized by gradient descent, in our application that means reducing the error in predicting the context words for a given input word. Once the model is trained we get the word embeddings. These embeddings can be plotted on to a graph. The distance between the embeddings for each input word gives the similarity or the dissimilarity between those words. When the embeddings are plotted on a 2d plot we see that the values for the word 'availability' and 'uptime' are quite close to each other. Hence, the definition of these words can be considered to belong to the same context. The input vocabulary can be easily changed to adapt and categorize new words coined for the concept. This kind of knowledge will help the machine understand the different terms for the same usage across various SLAs.

4.2.4 *Extraction of Roles and Responsibilities from SLA*

Extraction of Responsibilities:

Researchers have used deontic principles in the past to analyze policy and legal documents. The work by Travis D.Breaux [36] utilized semantic web and text mining approaches to extract obligations and permissions from privacy policies. We also used similar techniques to extract deontic rules from cloud SLA documents. We explored the use

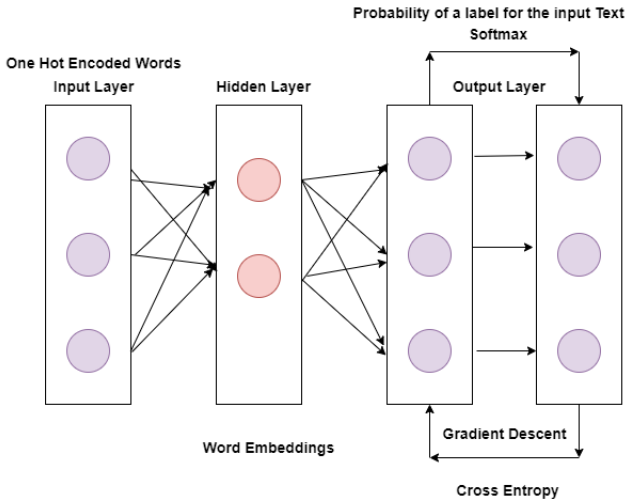

Fig. 4. Neural Network to Extract Word Context

<Noun / Pronoun> <modal verb> <verb>

Fig. 5. Grammar rule to extract responsibilities from SLA

of Modal Logic to extract rules or responsibilities present in the cloud SLA documents. Deontic logic or modal logic is a field of philosophical logic that deals with obligation, permission, and related concepts.

1) Permissions/Rights: Permissions are expressions that describe rights or authorizations for an entity/actor.
2) Dispensations: Dispensations describe optional or non-mandatory statements.
3) Obligations: Obligations define the responsibilities that an entity/actor must perform.

4) Prohibitions: Prohibitions specify the conditions or actions which an entity is not permitted to perform

For example, the sentences containing the words "must", "should", "will" impose obligations on the actor in the sentence.

To extract these statements we followed the below steps:

1) The SLA documents' text are first sentence tokenized
2) Every sentence is passed through a Stanford POS tagger [37]. This Parts of Speech tagger tags every word present in every sentence into a part of speech
3) Every word gets tagged an NN if it is a noun, MO if it is a modal verb, a VB if it is a verb, CD for cardinal digit, and so on.
4) We will make use of this tag to form a general grammar rule to extract statements as shown in figure 5.

This gives us the permissions and obligations statements. This method of analyzing will help the customers to understand their rights and obligations towards the services that they buy and use. It helps in understanding the document by giving it a structure to look at.

The categorization of sentences from SLA into the above categories resulted in the below findings

Example Sentence 1: *Google **will** make a determination in good faith based on its system logs, monitoring reports, configuration records, and other available information.* Type: **Obligation**

Such sentences are extracted from the SLA with the help of the modal words and then populated under the respective class in the cloud SLA knowledge graph, the statement in this example would get populated as a DataProperty value of the subclass Obligation under the class DeonticStatements.

Extraction of Roles :

Once the responsibilities are extracted, we extract the actors or the roles of these responsibilities. It is necessary to know who those permissions or obligations are on. Here we have two options for actors, the service provider and the customer. There are also statements which are more like definitions and hence do not contain either of the actors.

If the actor in a sentence is the service provider like for example, *"Amazon **will** provide 99.8 % availability"* then the obligations are on the service provider. Thinking grammatically and looking at the structure of the sentence the first approach is to extract the nouns, pronouns from the statements, the one's with tags 'NNP' and 'PRP' as the actors in that statement. This would work perfectly in a statement like the one shown in Example sentence 1, the actor would be 'Google'. However, if the statement has more than one actor like the one in Example sentence 2 then the above method of extracting nouns, pronouns as actors would not work. Example Sentence 2: *At our discretion, **we may** issue the Service Credit to the credit card **you** used to pay for the billing cycle in which the Unavailability occurred.*

Actor: CUSTOMER

Actor: SERVICE PROVIDER

Both the pronouns ('we' and 'you') would be captured as actors.Here the actors are service provider and customer. To extract the direct actor in such statements,we need a deeper structural knowledge of the statement and hence need more information about their dependencies in the statement. To achieve this we extract the dependencies of words on each in a statement using dependency parsing [38] on the extracted deontic statements

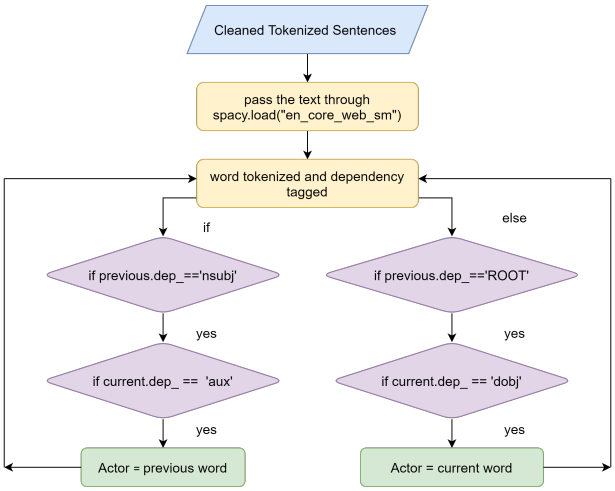

Fig. 6. Actor extraction from dependency parsed statements

"Dependency parsing is the task of extracting a dependency parse of a sentence that represents its grammatical structure and defines the relationships between "head" words and words, which modify those heads [39]." Dependency parsing can be used to extract sentences of both the types, Example sentence 1 and Example sentence 2 using the steps in Figure 6

Steps for dependency parsing for Type 1 statements (Example Sentence 1):

1) The sentences with modal words are first determined.
2) From these sentences the 'ROOT' words like 'apply','comply' etc,. are captured.
3) If the previous word is the 'ROOT' word and the current word has dependency 'dobj' (direct object) [40], extract the direct object ('dobj') .
4) Tag the current word as the 'Actor' in the statement.

Example Sentence: *Your failure to provide the request **will** disqualify **you** from receiving a Service Credit*

Dependency parsing: poss(failure-2, Your-1) nsubj(disqualify-8, failure-2) aux(provide-4, to-3) acl(failure-2, provide-4) det(request-6, the-5) dobj(provide-4, request-6) aux(disqualify-8, will-7) ROOT(disqualify-8, disqualify-8) **dobj(disqualify-8, you-9)** prep(disqualify-8, from-10) pcomp(from-10, receiving-11) det(Credit-14, a-12) compound(Credit-14, Service-13) dobj(receiving-11, Credit-14) punct(disqualify-8, .-15)

'ROOT' : 'disqualify'

'dobj' : 'you'

Actor: we (Service Provider)

Steps for dependency parsing for Type 2 statements (Example Sentence 2):

1) The sentences with modal words are first determined.
2) From these sentences the 'ROOT' words like 'apply','comply' etc,. are captured.
3) Check if the previous dependency is nominal subject (nsubj) [41] and the current word dependency is 'aux', (An aux (auxiliary) of a clause is a function word associated with a verbal predicate that expresses categories such as tense, mood, aspect, voice or evidentiality) [42]

4) If yes, then extract the nominal subject ('nsubj') [41] of the 'ROOT' word.
5) Tag the 'nsubj' as the 'Actor' in the statement (previous word).

Example Sentence:

*At our discretion, **we may** issue the Service Credit to the credit card **you** used to pay for the billing cycle in which the Unavailability occurred.*

Dependency parsing: *prep(issue-7, At-1) poss(discretion-3, our-2) pobj(At-1, discretion-3) punct(issue-7, ,-4) **nsubj(issue-7, we-5) aux(issue-7, may-6) ROOT(issue-7, issue-7)** det(Credit-10, the-8) compound(Credit-10, Service-9) dobj(issue-7, Credit-10) prep(issue-7, to-11) det(card-14, the-12) compound(card-14, credit-13) pobj(to-11, card-14) nsubj(used-16, you-15) relcl(card-14, used-16) aux(pay-18, to-17) xcomp(used-16, pay-18) prep(pay-18, for-19) det(cycle-22, the-20) compound(cycle-22, billing-21) pobj(for-19, cycle-22) prep(occurred-27, in-23) pobj(in-23, which-24) det(Unavailability-26, the-25) nsubj(occurred-27, Unavailability-26) relcl(cycle-22, occurred-27) punct(issue-7, .-28)*

'ROOT' : 'issue'
'aux' : 'may'
'nsubj' : 'we'
Actor: we (Service Provider)

Once the deontic statements and the actors for each statement are captured from every SLA under test. We populate it onto the knowledge graph as a data property 'hasactors' of the class 'Obligations' and 'Permissions'.

4.3 Knowledge Graph Auto-Population

A cloud SLA knowledge graph is built as shown in figure 7 with NIST standards as the base, once the schema is built, the extracted text from the above methods is populated onto this knowledge graph. This is done in the form of triplets, subject, object and predicate. The various aspects of the NIST standard become the main classes and subclasses of the knowledge graph as mentioned earlier, for example, Remedyfornoncompliance standard as mentioned by NIST forms a class in the knowledge graph. Each service provider is considered an instance of the above-built classes and subclasses. The value or the actual sentences corresponding to the NIST standard in each provider's SLA becomes the DataProperty value of a class/subclass and in turn of the instance, where the DataProperty would define the relationship between the class and the values. Once the SLA terms are identified we save them as an RDF graph which is machine-understandable. After which the key components can be queried from the knowledge graph using Apache Jena Fuseki. It could now be used by the user to compare various of SLAs, assisting in choosing the best one.

Adding these terms as classes and establishing the relationship between them in the graph could be done automatically with the help of our script that uses the library 'Owlready'(python library for modifying) owl files. The main classes in the knowledge graph are 'CloudSLA', 'DeonticStatements', 'SLAStandards', 'TermsandAdjustments', with subclasses. The service providers 'AWS', 'GCP', 'Azure', 'VMware' form the instances. Once the knowledge graph was completely built it was tested with other SLAs like 'IBMcloud', 'Alibaba Cloud', 'Oracle', 'Rackspace' etc. for checking the capturing capacity of the framework. The key steps followed to populate the knowledge graph are :

- Fetch and pre-process the SLA of a service provider
- Extract the key terms from the SLA using various Text Mining and Natural Language Processing techniques.
- Obtain the Noun Phrase and the Verb Phrase using the Stanford parser.
- Write the triples extracted, onto a JSON file as key-value pairs
- Then the JSON file is read and the key-value pairs are populated onto the knowledge graph. The terms now become the classes in the knowledge graph and the relationship fills in as Object Property and Data Property.

4.4 Query the knowledge graph

Once the knowledge graph is created we then query the knowledge graph using SPARQL [43]. SPARQL is a semantic query language with which we can query the data stored in the knowledge graph in the RDF format. We used the Apache Jena Framework [44] for running the SPARQL queries. One with basic SPARQL query knowledge would be able to extract the information from the knowledge graph. The extracted information is displayed in the form of text, graph or tables. The user can either select a cloud service name to analyze the properties or type in a natural language query. Using the result the user can compare the permissions of various service providers side by side, similarly, the obligations, definitions and non-compliance remedies and make a calculated decision on choosing the service provider, an example of which is shown in figure 9.

5 Results and Validation

The evaluation is classified into two section Knowledge Graph Evaluation and Actor Evaluation

5.1 knowledge graph Evaluation

The populated knowledge graph can be checked for correctness using 2 methods as below.

5.1.1 Knowledge Graph Population Count

Every Population into the knowledge graph is recorded. Each populated class increases the total count by one. We have also captured the number of entries for each class population for a particular SLA. The knowledge graph population capacity has been captured for every instance as shown in figure 8.

We see that the class RemedyforNonComplaince population value for VMware is 0, now that does not mean that VMware does not have remedy for non-compliance, it only means that the remedies are not present in the SLA in the format that we are setting up in the framework. VMware's Remedy for non-compliance is scattered around in the entire text and not placed in a table format like other popular providers. Similarly count of definitions, obligation, permission statements, actors, etc. are captured. This gives us an estimate of the completeness of each provider's SLA with respect to NIST standards.

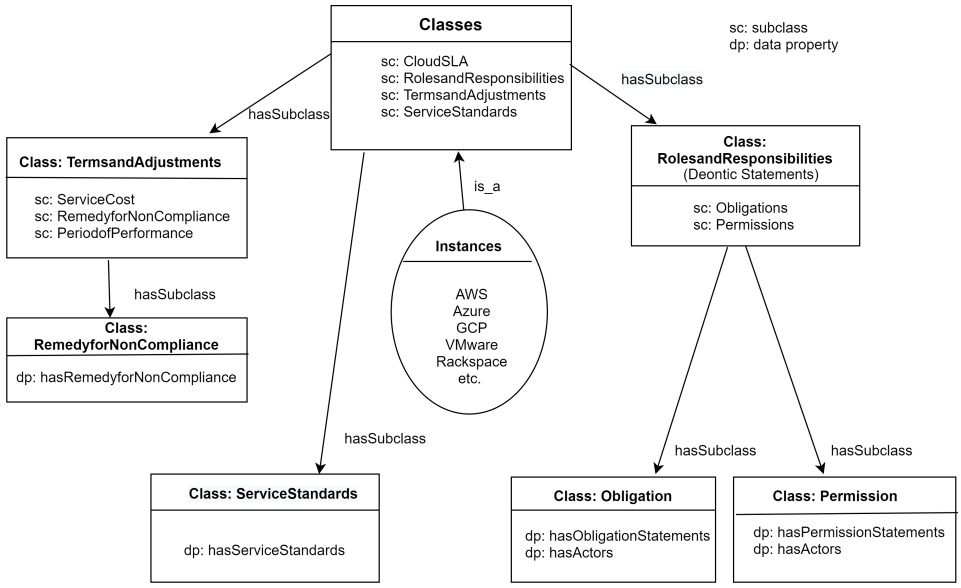

Fig. 7. SLA Knowledge Graph

GCP Population Count : 6 ObligationStatements : 11 ObligationActors : 22 PermissionStatements : 2 PermissionActors : 4 RemedyForNonCompliance : 1 Definitions: 8	**AWS** Population Count : 6 ObligationStatements : 12 ObligationActors : 24 PermissionStatements : 4 PermissionActors : 8 RemedyForNonCompliance : 3 Definitions: 4	**Azure Stats** Population Count : 6 ObligationStatements : 20 ObligationActors : 40 PermissionStatements : 9 PermissionActors : 18 RemedyForNonCompliance : 3 Definitions: 20	**VMware** Population Count : 5 ObligationStatements : 16 ObligationActors : 32 PermissionStatements : 14 PermissionActors : 28 RemedyForNonCompliance : 0 Definitions: 6
Rackspace Population Count : 5 ObligationStatements : 8 ObligationActors : 16 PermissionStatements : 3 PermissionActors : 6 RemedyForNonCompliance : 0 Definitions: 23	**SAP** Population Count : 5 ObligationStatements : 38 ObligationActors : 76 PermissionStatements : 36 PermissionActors : 72 RemedyForNonCompliance : 0 Definitions: 15	**Oracle** Population Count : 5 ObligationStatements : 48 ObligationActors : 96 PermissionStatements : 63 PermissionActors : 126 RemedyForNonCompliance : 0 Definitions: 16	**Alibaba Stats** Population Count : 6 ObligationStatements : 8 ObligationActors : 16 PermissionStatements : 3 PermissionActors : 6 RemedyForNonCompliance : 3 Definitions: 3

Fig. 8. knowledge graph Population Count

5.1.2 SPARQL Query

The knowledge graph that is built can be queried to obtain various results based on the user's requirement. Below are a few user test cases (the information a user might be interested in while purchasing a service), with their respective SPARQL query and results.

1) Case 1: Provide the obligations statements with actors that talk about 'Service Credits'. Getting this kind of specific information by a simple query significantly benefits the user.
2) Case 2: Compare the credits refunded by Cloud Providers if the availability is around 95% We notice that the credits returned by Azure are a 100% whereas by GCP are just 50% for the same monthly drop in uptime. This comparison would be cumbersome if it had to be done manually.

5.2 Actor evaluation

Actors extracted can be of three categories, Service Provider, Customer, and Neither. The distribution of the different categories of actors across the obligation and permission statements for various SLA is as shown in figure 11. The framework extracts correct actors using the earlier mentioned methodology, however, the framework also incorrectly extracted actors for a few statements. This could be because of the complexity of the statement or that the statement does not follow the extraction structure which is the one followed by the majority of the popular providers.

Correct Extraction

Actor: Service Provider *"Actor: VMware Sentence : VMware will review the request and issue a Service Credit when VMware validates the SLA Event based on VMware's data and records "*

Actor: Customer *"Actor: you Sentence : Service Credits will not entitle you to any refund or other payment from AWS "*

```
PREFIX onto:<http://ebiquity.umbc.edu/ontologies/itso/1.0/cloudSLA.owl#>
SELECT * WHERE { {?subject onto:hasObligationActors ?object.filter regex(?object,'Service Credit','i')}}
```

onto:AWS	"Actor: you Sentence : Your failure to provide the request and other information as required above will disqualify you from receiving a Service Credit "
onto:AWS	"Actor: Credit Sentence : A Service Credit will be applicable and issued only if the credit amount for the applicable monthly billing cycle is greater than one dollar ($1 USD) "
onto:VMware	"Actor: you Sentence : Service Level Agreement Claims In order to request any Service Credit, you must file a support request at https://my.vmware.com within sixty (60) days of the suspected SLA Event "

Fig. 9. Case 1: Provide the obligation statements with actors that talk about 'Service Credits'.

```
PREFIX onto:
        <http://ebiquity.umbc.edu/ontologies/itso/1.0/cloudSLA.owl#>
SELECT * WHERE { ?subject onto:hasRemedyforNonCompliance ?object.filter regex(?object,'95','i')}
```

onto:AWS	"Monthly Uptime Percentage: Less than 95.0% Service Credit Percentage : 100% "
onto:GCP	"Monthly Uptime Percentage: 95.00% - < 99.00% Service Credit Percentage : 25% "
onto:GCP	"Monthly Uptime Percentage: < 95.00% Service Credit Percentage : 50% "
onto:Azure	"Monthly Uptime Percentage: < 95% Service Credit Percentage : 100% "

Fig. 10. Case 2: Compare the credits refunded by Cloud Providers if the availability is around 95%

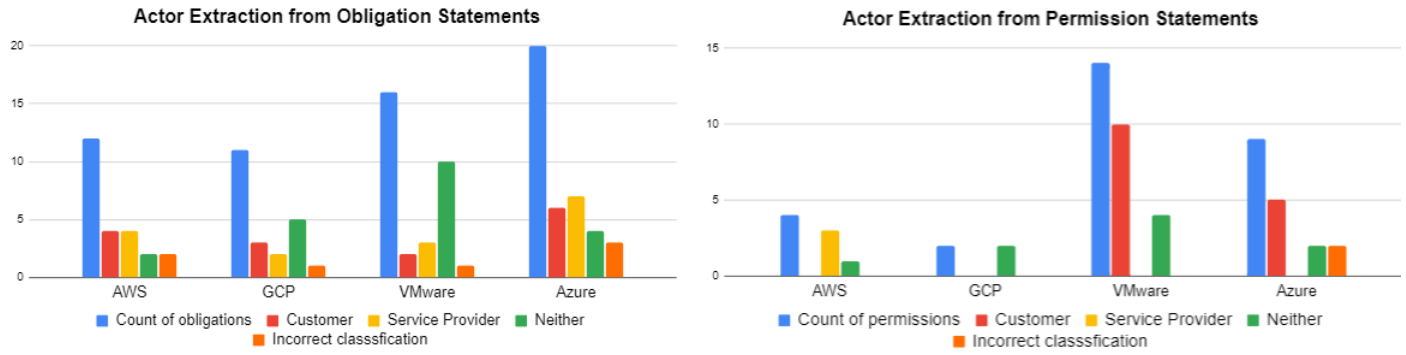

Fig. 11. Actors Distribution Plot

Neither: *"Actor: Sentence : Capitalized terms not defined in this SLA will have the meanings specified in the Terms of Service"*

Incorrect Extraction

"Actor: you Sentence : For these Services, any Service Credit that you may be eligible for will be credited in the form of service time (i.e., days) as opposed to service fees, and any references to Applicable Monthly Service Fees is deleted and replaced by Applicable Monthly Period. "

No actor should be extracted for the above statement(the output should be 'Neither'). This incorrect extraction is because of the structure of the statement that does not comply with both Type 1 or Type 2 statements. Adding this structure to the framework is possible, but has been deliberately not done to avoid over-fitting the framework for a small section of irregularity in the structure.

5.3 Survey to Check the Correctness of Actor Extraction

We evaluated the captured actors by randomly selecting 10 deontic statements. These statements were sent to 5 individuals to identify the actors in them if there are any. The statements sent for evaluation were :

Statements Number	Framework Extracted Actors	Majority Poll
1	Service Provider	Service Provider
2	Customer	Customer
3	Service Provider	Service Provider
4	Neither	Customer
5	Service Provider	Service Provider
6	Service Provider	Service Provider
7	Service Provider	Service Provider
8	Neither	Neither
9	Customer	Customer
10	Neither	Neither

Fig. 12. Evaluation of the framework but checking the correctness of actors capturing

1) *AWS will use commercially reasonable efforts to make the Included Services each available for each AWS region with a Monthly Uptime Percentage of at least 99.99%, in each case during any monthly billing cycle (the Service Commitment).*
2) *In the event any Single EC2 Instance does not meet the Hourly Commitment, you will not be charged for that instance hour of Single EC2 Instance usage.*
3) *For all Virtual Machines that have two or more instances deployed across two or more Availability Zones in the same Azure region, we guarantee you will have Virtual Machine Connectivity to at least one instance at least 99.99% of the time.*
4) *If the Agreement authorizes the resale or supply of Google Cloud Platform under a Google Cloud partner or reseller program, then all references to Customer in this SLA mean Partner or Reseller (as applicable), and any Financial Credit(s) will only apply for impacted Partner or Reseller order(s) under the Agreement.*
5) *If a dispute arises with respect to this SLA, Google will make a determination in good faith based on its system logs, monitoring reports, configuration records, and other available information, which Google will make available for auditing by Customer at Customers request.*
6) *At our discretion, we may issue the Service Credit to the credit card you used to pay for the billing cycle in which the Unavailability occurred.*
7) *A "Service Credit" is a dollar credit, calculated as set forth above, that we may credit back to an eligible account*
8) *The self-service console, available at https://vchs.vmware.com, cannot successfully authenticate a simulated user for more than five (5) consecutive minutes.*
9) *You may not unilaterally offset your Applicable Monthly Service Fees for any performance or availability issues.*
10) *Capitalized terms used but not defined in this SLA will have the meaning assigned to them in the Agreement.*

The results obtained are as shown in Figure 12

We notice that there is a mismatch in the majority poll and the extracted actor for statement 4, and this is because of the complexity of the statement. Though there is no well defined actor in statement 4, (according to dependency parsing rule), human beings could read, comprehend the statement and decipher that the actor here is 'Customer' which the system is failing to do. This is expected behaviour in the framework, more complex rules of extraction can overcome this issue.

6 Conclusion and Future Work

Currently, the cloud service SLA are text documents that require manual effort to read through and choose the right service, which is prone to error and bias. We have extracted the text from these documents and built an automatic extraction and comparison of cloud service level agreements system. This is done with the help of semantic web technologies and text mining techniques involving the usage of OWL and RDF graph to store the extracted components. In this paper, we have described in detail the cloud SLA knowledge graph that we have developed. We have also described the prototype that we have developed to illustrate how the SLA measures can be automatically extracted from legal terms of service or customer agreement documents, that are available in the public domain.

Using this framework we have extracted the modal statements and their associated actors. With this information, we then classify the modal statements as permissions and obligations. Each permission/obligation is associated with specific actors. The actors here are 'Customers', 'Service Providers' or 'Neither'. Doing this, we have eased the process of finding the most suitable service provider for every customer. The customers now have the clarity of what their rights are and what are they obliged to do.

In our future work, we intend to expand this work to include other aspects of cloud SLA like compliance, and other documents in cloud services like the scope of services, liability insurance, etc. This can be done by adding a new class to the base knowledge graph, which would involve different NLP techniques based on the application. We also intend on making the knowledge graph more involved so that it can capture the complexity and ambiguity of the sentences which were incorrectly classified and tagged currently, which could be achieved by finding alternatives and improvements to dependency parsing. In addition to expanding the framework to other providers, which would bring in newer formats of phrase and term usage. We also have plans on automating text extraction across domains rather than just cloud service providers making it a one for all text processing framework. We plan on exploring other small text modelling techniques for better capture of important topics from the SLA as the traditional topic modeling does-not give satisfactory results due to the small size of the SLA.We will also expand the data-set for our system to include all legal documents associated with cloud services like terms of service, privacy policy, and service level agreements. This would also include a streamlined auto service selection process.

Acknowledgement

This research was partially supported by a DoD supplement to the NSF award 1747724, Phase I IUCRC UMBC: Center for Accelerated Real time Analytics (CARTA).

References

[1] S. A. Baset, "Cloud slas: present and future," *SIGOPS Operating Systems*, 2012.
[2] "International organization for standardization's iso/iec dis 19086," 2015. [Online]. Available: http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=67545

[3] "Cloud computing service metrics description," 2014. [Online]. Available: http://www.nist.gov/itl/cloud/upload/RATAX-CloudServiceMetricsDescription-DRAFT-20141111.pdf

[4] "Federal risk and authorization management program (fedramp)," 2015. [Online]. Available: http://csrc.nist.gov/groups/SMA/ispab/documents/minutes/2012-02/feb3_fedramp_ispab.pdf

[5] J. M. García, P. Fernández, C. Pedrinaci, M. Resinas, J. Cardoso, and A. Ruiz-Cortés, "Modeling service level agreements with linked usdl agreement," in *IEEE Transactions on Services Computing, vol. 10, no. 1, pp. 52-65, 1 Jan.-Feb. 2017*.

[6] L. Sun, J. Ma, H. Wang, Y. Zhang, and J. Yong, "Cloud service description model: An extension of usdl for cloud services," in *IEEE Transactions on Services Computing, vol. 11, no. 2, pp. 354-368, 1 March-April 2018*.

[7] S. Tata, M. Mohamed, T. Sakairi, N. Mandagere, O. Anya, and H. Ludwig, "rsla: A service level agreement language for cloud services," in *IEEE 9th International Conference on Cloud Computing (CLOUD), San Francisco, CA, 2016, pp. 415-422*.

[8] "Guide to information technology security services," 2015. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-35.pdf

[9] K. P. Joshi, Y. Yesha, and T. Finin, "Automating cloud services life cycle through semantic technologies," *Services Computing, IEEE Transactions on*, vol. 7, no. 1, pp. 109–122, 2014.

[10] A. Gupta, S. Mittal, K. P. Joshi, C. Pearce, and A. Joshi, "Streamlining Management of Multiple Cloud Services," in *IEEE International Conference on Cloud Computing*. IEEE Computer Society, June 2016, p. 8.

[11] S. Mittal, K. P. Joshi, C. Pearce, and A. Joshi, "Automatic extraction of metrics from slas for cloud service management," in *IEEE International Conference on Cloud Engineering*, 2016.

[12] "Resource description framework (rdf)," 2015. [Online]. Available: http://www.w3.org/RDF/

[13] "Owl web ontology language," 2015. [Online]. Available: http://www.w3.org/TR/owl-features/

[14] E. Christensen, G. Meredith, S. Weerawarana, and F. Curbera, "Web services description language (wsdl) 1.1." [Online]. Available: https://www.w3.org/TR/2001/NOTE-wsdl-20010315

[15] M. Turner, D. Budgen, and P. Brereton, "Turning software into a service." *Computer.*, vol. 36, no. 10, pp. 38–44, 2003.

[16] N. Oldham, K. Verma, A. Sheth, and F. Hakimpour, "Semantic ws-agreement partner selection," in *Proceedings of the 15th international conference on World Wide Web*. ACM, 2006, pp. 697–706.

[17] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu, "Web services agreement specification (ws-agreement)," in *Open Grid Forum*, vol. 128, 2007, p. 216.

[18] T. Bui and A. Gachet, "Web services for negotiation and bargaining in electronic markets: Design requirements and implementation framework," in *System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on*. IEEE, 2005, pp. 38–38.

[19] D. Rusu, L. Dali, B. Fortuna, M. Grobelnik, and D. Mladenic, "Triplet extraction from sentences," in *Proceedings of the 10th International Multiconference" Information Society-IS*, 2007, pp. 8–12.

[20] O. Etzioni, M. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates, "Unsupervised named-entity extraction from the web: An experimental study," *Artificial intelligence*, vol. 165, no. 1, pp. 91–134, 2005.

[21] K. Barker and N. Cornacchia, "Using noun phrase heads to extract document keyphrases," in *Advances in Artificial Intelligence*. Springer, 2000, pp. 40–52.

[22] Z. S. V. Mulwad, T. Finin and A. Joshi, "Using linked data to interpret tables," in *Proceedings of the First International Conference on Consuming Linked Data-Volume 665. CEUR-WS. org*, 2010, pp. 109–120.

[23] T. N. C. S. Bhagavatula and D. Downey, ""tabel: Entity linking in web tables," in *The Semantic Web-ISWC 2015. Springer*, 2015, p. 425–441.

[24] L. Kagal and T. Finin, "Modeling conversation policies using permissions and obligations," *Autonomous Agents and Multi-Agent Systems*, 2006.

[25] "Vmware service level agreement," 2019. [Online]. Available: https://www.vmware.com/support/vcloud-air/sla/

[26] "Amazon service level agreement," 2019. [Online]. Available: https://aws.amazon.com/compute/sla/

[27] "Microsoft azure sla," 2019. [Online]. Available: https://azure.microsoft.com/en-us/support/legal/sla/virtual-machines/v1_8/

[28] "Google compute engine sla," 2019. [Online]. Available: https://cloud.google.com/compute/sla

[29] "Service level agreement for ibmsoftlayer," 2019. [Online]. Available: https://www.softlayer.com/sites/default/files/sla.pdf

[30] "Rackspace service level agreement," 2019. [Online]. Available: https://www.rackspace.com/information/legal/cloud/sla

[31] "Sap cloud service level agreement," 2019. [Online]. Available: https://www.sap.com/about/trust-center/agreements/cloud/cloud-services.html?search=Service+Level+Agreement&sort=title_asc&tag=language\%3Aenglish&pdf-asset=ca85dd8e-027d-0010-87a3-c30de2ffd8ff&page=1

[32] "Alibaba cloud service level agreement," 2019. [Online]. Available: https://www.alibabacloud.com/help/doc-detail/42436.htm

[33] "Oracle service level agreement," 2019. [Online]. Available: https://www.oracle.com/cloud/iaas/sla.html#availability

[34] "Link grammar," 2015. [Online]. Available: http://www.link.cs.cmu.edu/link/

[35] S. Mittal, K. P. Joshi, C. Pearce, and A. Joshi, "Parallelizing natural language techniques for knowledge extraction from cloud service level agreements," in *Big Data (Big Data), 2015 IEEE International Conference on*. IEEE, 2015, pp. 2831–2833.

[36] T. D. Breaux and A. I. Anton, "Analyzing goal semantics for rights, permissions, and obligations," in *RE'05: Proceedings of the 13th IEEE International Requirements Engineering Conference (RE'05)*. IEEE Computer Society, August 2005, pp. 177–186.

[37] "Stanford pos tagger," 2015. [Online]. Available: https://nlp.stanford.edu/software/tagger.shtml

[38] D. J. . J. H. Martin., "Speech and Language Processing," October 2019. [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/15.pdf

[39] "Dependency parsing," 2018. [Online]. Available: http://nlpprogress.com/english/dependency_parsing.html

[40] "Direct object dependency," 2018. [Online]. Available: https://universaldependencies.org/docs/u/dep/dobj.html

[41] "Nominal subject dependency," 2018. [Online]. Available: https://universaldependencies.org/u/dep/nsubj.html

[42] "Auxiliary dependency," 2018. [Online]. Available: https://universaldependencies.org/u/dep/aux_.html

[43] "Sparql 1.1 overview," 2013. [Online]. Available: http://www.w3.org/TR/sparql11-overview/

[44] "Jena apache fuseki: serving rdf data over http,," 2015. [Online]. Available: http://jena.apache.org/documentation/serving_data/index.html

Divya Natolana Ganapathy is a Master's student at the University of Maryland, Baltimore County. Her primary research focus is knowledge extraction from various text data sources. She is working on developing techniques to automate Cloud Service Level Agreements (SLAs) and Privacy Policies. Prior to her masters, she has worked with companies like HP and Nokia as a research and development engineer.

Dr. Karuna P. Joshi is an Associate Professor of Information Systems at UMBC and UMBC Site Director of Center for Accelerated Real Time Analytics (CARTA). She also directs the Knowledge Analytics Cognitive and Cloud (KnACC) Lab. Her research focus is in the areas of Data Science, Cloud Computing, Data Security and Privacy and Healthcare IT systems. She has published over 50 papers and her research is supported by ONR, NSF, DoD, GE Research and Cisco. She teaches courses in Big Data, Database Systems Design and Software Engineering. She received her MS and PhD in Computer Science from UMBC, where she was twice awarded the IBM PhD Fellowship, and her Bachelors in Computer Engineering from the University of Mumbai, India. Dr. Joshi also has extensive experience of working in the industry primarily as an IT Program/Project Manager at the International Monetary Fund.