A Semantically Rich Framework to Automate Cloud Service Level Agreements

Karuna P Joshi, *Member, IEEE,* Divya N Ganapathy, and Sudip Mittal

Abstract—Consumers evaluate the performance of their cloud-based services by monitoring the Service Level Agreements (SLA) that list the service terms and metrics agreed with the service providers. Current Cloud SLAs are long 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 ensure continuous SLA monitoring. 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 Ontology. 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 the challenges in automating cloud services management and describe our framework and the results of validating them against SLAs of selected Cloud vendors.

Index Terms—Service Level Agreements, Semantic Cloud Services, Cloud Service Level Agreements, Semantic Web, Modal Logic, Cloud Management and Operations.

1 INTRODUCTION

Organizations are increasingly migrating their data 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]. During the service negotiation phase [2] providers and consumers agree on the service performance metrics and data security and privacy policies, that are included in the SLA contracts. 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 contracts are currently managed as large text documents and have to be manually parsed by the consumers to determine which service best suits their needs which 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 the business process to track availability and user access statistics.

As SLA standards are still in 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. Monitoring service performance based on these documents is time consuming and tedious for the consumers and so often happens only when the provided service fails to live up to the expectations.

There should be a common platform with a common understanding through which information exchange and querying can be done. Therefore, it is important to monitor the SLA continuously. This can be done by automating the cloud service management by making the SLA machine understandable so that the monitoring tool could analyze and make decisions based on the knowledge extracted from the service contracts. We have used a combination of semantic web technologies, text mining techniques and natural language processing to build the framework. The ontologies created in previous work has been used and extended to add extra classes like service credits to depict the actual cost of services. We have also compared the changes that

- *K. Joshi is with the Information Systems Department, University of Maryland, Baltimore County, Baltimore, MD, 21250.*
- *D. N. Ganapathy is with the Computer Science and Electrical Engineering Department, University of Maryland, Baltimore County, Baltimore, MD, 212250.*
- *S. Mittal was with the Computer Science and Electrical Engineering Department, University of Maryland, Baltimore County, Baltimore, MD, 21250. and is now with the Computer Science Department at University of North Carolina Wilmington, Wilmington, NC, 28403.*

have occurred in the SLA of cloud service providers over the past few years [3]. We have monitored the change in the services provided as well as changes in structure and content of the SLA using modal word frequency and the ontology comparison.

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

2 Related Work

2.1 SLA Standards

Baset [1] describes an SLA as consisting of 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 specify the Service guarantee exclusions. Service violation measurement and reporting describes 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 Standardizations ISO/IEC DIS 19086 standard [4], European Commissions "Cloud Service Level Agreement Standardization Guidelines and NISTs Special Publication 500-307 on Cloud Computing Service Metrics Description [5], 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 [6]. 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) [5]. The SLA content areas are as follows:

1) Accessibility: Usability of a product, service, environment or facility by people within the widest range of capabilities.
2) Business continuity: Capability of the organization to continue delivery of products or services at acceptable predefined levels following a disruptive incident.
3) Cloud service agreement: The documented agreement between the cloud service provider and cloud service customer that governs the covered service.
4) Cloud SLA: Part of the cloud service agreement that includes cloud service level objectives and cloud service qualitative objectives for the covered cloud service.
5) Cloud service level objective (SLO): Commitment a cloud service provider makes for a specific, quantitative characteristic of a cloud service, where the value follows the interval scale or ratio scale.
6) Cloud service qualitative objective (SQO): Commitment a cloud service provider makes for a specific, qualitative characteristic of a cloud service, where the value follows the nominal scale or ordinal scale.
7) Disaster recovery: Ability of the ICT elements of an organization to support its critical business functions to an acceptable level within a predetermined period of time following a disaster
8) Failure notification policy: Policy specifying the processes by which the cloud service customer and cloud service partner can notify the cloud service provider of a service outage and by which the cloud service provider can notify the cloud service customer and cloud service partner that a service outage has occurred. Interval scale: continuous scale or discrete scale with equal-sized scale values and an arbitrary zero.
9) Metric: Standard of measurement that defines the conditions and the rules for performing the measurement and for understanding the results of a measurement.
10) Nominal scale: Scale with unordered labeled categories or ordered by convention and so on.

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 characteristic 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 monitoring and analysis of cloud service level agreements [2]. 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 [7] (Resource Description Language) and OWL [8] for describing the objects and relations, define ontologies and describing the meta-data using these ontologies.

Another approach uses Web Services Definition Language (WSDL) [9] 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. [10] 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.

Web Services Agreement Specification (WS-Agreement) is a web services protocol for establishing the agreement between two parties using an extensible XML language for specifying the nature of the agreement, and agreement templates to facilitate the discovery of compatible agreement parties. WS-Agreement limits the ability to match the agreements to syntactical matching and is also very limited in matching non-functional attributes that define policies on data and security, compliance issues, data quality levels, etc.

Oldham et al. [11] have proposed semantically rich extensions to extend the WS-Agreement. Aiello et al. [12] 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 [13] 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. Skogsrud et al. [14] propose a trust negotiation framework that supports policy lifecycle management for web services. However, this framework is limited to managing the user identity of customers accessing the service. Yao et al. [15] propose flexible strategies for web services negotiation. Their approach does not use policy management to automate service negotiation. There has also been work done on negotiating between web services, however, this work does not relate to the negotiation needed between service provider and consumer.

For our cloud services lifecycle framework, we have used semantic web technologies like OWL instead of WS-Agreement and WS-Negotiation protocol as we were able to more richly define the cloud SLA ontologies, thereby allowing us to incorporate different descriptions of the same SLA measure. Our proposed ontology is flexible enough to capture knowledge in terms of existing and future SLA standards.

2.3 Text Extraction

Researchers have applied Natural Language Processing (NLP) techniques to extract information from text documents. In Rusu et. al. [16] 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. [17] 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. [17] 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. [18] 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.

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. [19] 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. [20] 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. [21] formed, proposed a semantic web-based policy framework to model conversation specifications and policies using obligations and permissions.

In one of our prior works, we described a new integrated methodology for the lifecycle of IT services delivered on the cloud and we demonstrated how it can be used to represent and reason about services and service requirements and to automate service acquisition and consumption from the cloud. We have divided the IT service lifecycle into five phases of requirements, discovery, negotiation, composition, and consumption. We detail each phase and describe the ontologies that we have developed to represent the concepts and relationships for each phase. We have described the five phases in detail along with the associated metrics.

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 Cloud Legal Documents

The Cloud legal documents are signed agreements between the cloud service provider and consumer. These are currently maintained as text documents and include proper metrics and measurements to standardize the procedure of agreements and also to monitor it once the agreement is done.

3.1 Types of Legal Documents

We conducted a comprehensive survey of publicly available cloud legal documents by reviewing over 100 cloud providers [2], [3], [22]. Details of results of that study have been published. The cloud legal documents fall into the following broad categories:

3.1.1 Service Contract documents

Cloud legal documents in this category lay down rules and clauses that specify service functionality, quality and performance metrics. They also specify user access policies and service availability. Documents in this category include the Service Level Agreement, Terms of Service, Customer Agreement, Acceptable Use Policy, etc. A key role of these contracts is to enable efficient management of the cloud service. Metrics listed in these contracts are used by the vendors to assure the consumer of high performance of their services. NISTs special publication 500-307 [5] defines 'metrics as provides knowledge about characteristics of a cloud property through both its definition (e.g. expression, unit, rules) and the values resulting from the observation of the property'. For instance, a customer response time metric can be used to estimate a specific response time property (i.e. response time from customer to customer) of a cloud email service search feature. It also provides the necessary information that is needed to reproduce and verify observations and measurement results. However, one glaring observation after reviewing these contracts was the sheer variety of formats of these documents and the metrics used to track the performance of the same type of service. Due to lack of a standard cloud contract format, we came across SLA documents that were 3 pages long to ones that were over 25 pages in length even though they were offering the same type of computing service.

3.1.1.1 ***Components of SLA***:
In our research, we focus on SLA documents for various cloud services. The main components of an SLA document are:

- Service Commitment: The SLA document specifies the service commitments made by the service provider to its customers.
- Definitions and metrics: The SLA documents also contain definitions and metrics such as 'Availability', 'Region' and 'Uptime/Downtime' which are used to compute the service commitments.
- Rules for computation of service credit corresponding to the commitments by the provider.
- Procedure to request for service credit by the user and payment method by the service provider. These describe the specifications about how a user can apply for a service credit, for example, the request for service credit must include the account information and exact dates of outages of the service.
- Conditions under which the service provider is not liable to its service commitment (exclusions) to the users
- Deontic Expressions: These are modal statements that comprise 'Permission', 'Obligation', 'Dispensation' and 'Prohibition'. These could pertain to the service provider or the customer based on the actor in that statement. It helps the customer to understand the key components like liabilities and rights concerning the subscribed service.

3.1.2 Privacy and Security Data Documents

Cloud legal documents in this category lay down the data security and privacy policies for the service. These documents are usually required by federal and state regulations. Cloud providers either create a single privacy policy for all their services or create multiple privacy policies depending on the service data/category. Privacy policy documents were observed to be similar in content and format across most of the cloud vendors.

3.1.3 Regulatory Compliance documents

Compliance are set of rules, policies or standards formulated by regulatory agencies or standards organizations [5]. Compliance models implement rules and regulations across various components of Information Technology (IT) to make them work harmoniously. Security and privacy compliance models have been proposed for cloud computing security to ensure data protection and user privacy. Some of the proposed models include ISO 27001 [4] , COBIT, etc.

3.2 Research Challenge

- 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. Also, regulatory and compliance bodies have also developed rules and policies that affect the way cloud services can be provided or consumed. Reviewing all these cloud

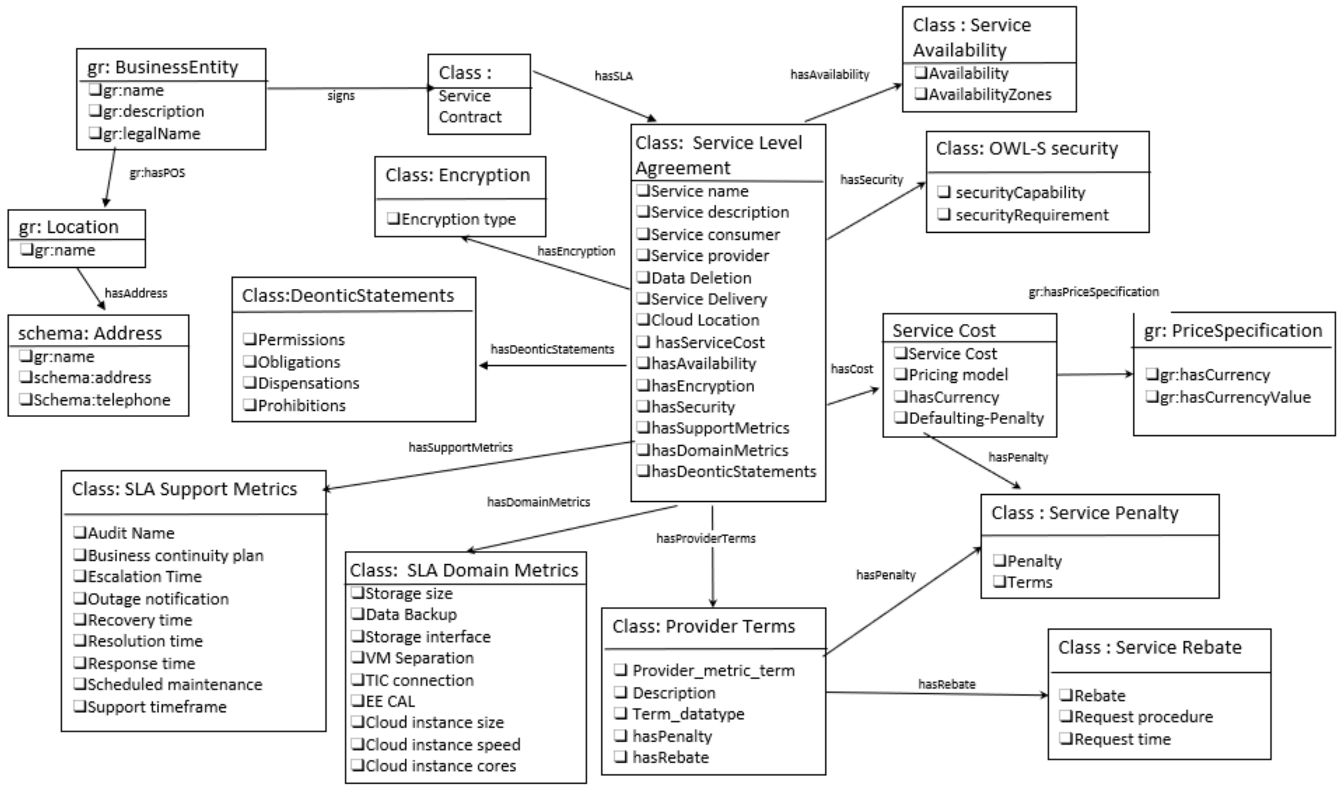


Fig. 1. Ontology for Cloud Service Level Agreement (SLA)

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. Hence, continuous cloud service monitoring has been identified as a key open issue by consumers. A critical step in automating cloud service management is to make the cloud SLAs machine-understandable so that monitoring tools can interpret the policy rules and metrics defined in the service contracts.

- 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.

4 Ontology for Cloud Service Level Agreement

4.1 Introduction

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 ontology. Ontology is based on knowledge representation language, OWL (Web Ontology Language). The ontology could be queried using SPARQL to obtain the stored relationship.

4.2 Classes

1) ***Class: ServiceLevelAgreement***: This is the main class containing subclasses like service name, service description, service delivery, etc. It has objectproperties relating it to other classes like hasDeonticStatements.
2) ***Class:DeonticStatements*** :The class contains subclasses "Permission", "Obligation", "Dispensation", "Prohibition". The statements in the SLA of a service provider 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: Service Penalty*** : The class Service Penalty has the penalty terms and conditions with respect to the service provider and customer.
4) ***Class: Service Rebate*** :This class defines the policy and procedure for the customer to get a refund from the service provider in situations of service unavailability or service failure.
5) ***Class: SLADomainMetrics*** : This class provides the service specifications like storage size, data backup etc.

4.3 Relationship

Relationships in the Ontology is the relationship between two classes which is called the ObjectProperty or the relationship between an instance and value of that instance called the DataProperty. These relationship has been extracted using various NLP techniques and represented in the Ontology.

1) ***hasRebate***: Service provider hasRebate Service Rebate. This relationship is the predicate part of the triple,

showing a relationship between the subject and the object. The Service Provider has a rebate which is defined in the class Service Rebate.

2) ***hasDeonticStatements***:This ObjectProperty represents the relationship between Service Level Agreements and the Deontic Statements class. The instances of Service Level Agreements will have the hasDeonticStatement relationship to the instance in Deontic Statements Class
3) ***hasSupportMetrics***: This relationship relates the instances of class Service Level Agreement to SLA support metrics which has various metrics like Audit Name, Escalation time, Recovery Time, etc.

We found that the most time-consuming process of cloud service consumption is the service negotiation process. Semantic web [23] usage for automation of this process itself is a performance improvement over tedious manual effort. The negotiation would involve discussion and agreement that the service provider and consumer have, regarding the service delivered and its acceptance criteria. A specification is laid out by the consumer as to what service they require in the RFS (Request for Service). The service provider would then layout the SLA which should be agreed upon by both the provider and the consumer. SLAs define the service data, delivery mode, agent details, quality metrics and cost of the service. While negotiating the service levels with potential service providers, consumers can explicitly specify service quality constraints (data quality, cost, security, response time, etc.) that they require. There could be instances where a service provider would provide a service that contains components provided by multiple providers. The SLA should clearly state this relationship and responsibilities. There could be a primary and secondary service provider, the primary provider would interface with the consumer and hence would be responsible for the composition of the services. The primary provider will negotiate the service metrics with the secondary provider. The service metrics have been defined in the ontology that was created using OWL. The Service Level Agreement class consists of properties that are common across all cloud applications. The class SLA has the property hasCost has a class Service Cost which has properties like Service Cost, Pricing Model, etc. SLA has property hasAvailability which has a class Service Availability which has properties like Availability and Terms of availability [2]. These are very helpfully in automating the service consumption and negotiation process as this would help in a clear comparison of various cloud service providers.

The cloud service consumers will have a service contract which contains SLA that can be stored as instances in the proposed ontology [2] and managed as RDF graphs, as shown in Figure 1, which can be automatically queried using SPARQL [24]. Any updates to the SLA would be stored as an RDF graph [7]. This was done so that the consumers can keep a track of SLAs of all the services that they have been using over the years and use that information to plan the future. This ontology is available in the public domain and can be accessed. This ontology forms part of the integrated ontology that we developed for our integrated cloud life cycle framework. The ontology has been extended to include hasExclusions property in Class Service Rebate which would determine the terms which cause the customers to not be able to get a service refund, this includes the procedure for a refund and the time taken to file the request. A property 'Availability Zones' is added as this one, from the research, seems to be is an important factor in calculating the availability of a service, especially when a refund is to be made when service is unavailable.

5 METHODOLOGY

Through this research, we aim to enable the consumer to manage multiple cloud services. Our framework has a few elements as below to achieve this overall goal. This is done using text mining and semantic web techniques on the available SLAs. We aim to have a system that would maintain knowledge about various terms and rules in the SLA obtained by SLAs compliance and regulatory policies, contracts, privacy documents, etc.

5.1 Data-set Used for the Experiment

We have used data from publicly available SLA 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].

5.2 SLA Ontology Creation

We begin by downloading publicly available SLA documents of various cloud service providers. We extract the key terms like availability, uptime, cost, etc. from these documents and then map it to the cloud SLA ontology. Once the SLA terms are identified we save them as an RDF graph which is machine-understandable. Once this is done it could be used to automate the monitoring of SLA compliance. To achieve this, we used semantic web technologies like OWL, RDF, and SPARQL. To store and query the cloud SLA ontology Jena Apache Fuseki store was used.

Using the prototype [22] the user selects the service provider from the drop-down list and then clicks on the extract SLA terms to display all the SLA terms included in that service provider. The service provider used in the prototype are Google for Google Apps, from Microsoft for MS Azure, from Amazon for AWS and from Hewlett Packard for their compute cloud. As part of ongoing work, more service providers will be added. When the extract term button is clicked the system parses the entire document to discover the key measures in that SLA.

The SLA terms identified in the terms of service document can be mapped to the ontology when a service provider is selected. The system identifies, for example the term 'availability', this term could now be mapped to the class 'Service Availability' in the cloud SLA. We have been able to add terms and values to the ontology in an automated fashion with no manual intervention. Different key terms could form classes in the ontology with a relationship between them. Adding these terms as classes and establishing the relationship between them in the graph could be done automatically with the help of a script that uses the library 'Owlready'.

Once the SLA terms are mapped we parse the text again to find the SLA measures corresponding to that term. For instance, the HP Compute service SLA has an availability of 99.95 %. Thus we set the SLA measure for service availability to 99.95 %.

One challenge that was faced was the usage of different terminology by different providers. The term 'availability' could also be termed as 'uptime'. This is difficult to capture in the above approach, we can address this issue by adding descriptions of the same measure in our ontology. We store the SLA instance in the form of an RDF graph using the SPARQL CONSTRUCT. This SLA graph is stored in a Fuseki graph store [30], which can be easily queried for continuous SLA monitoring since it is in a machine-readable format. For our prototype, we used the SLAs for the same service type and so all vendors SLAs fit our framework. In our ongoing work, we are going to explore other service domains.

'

5.3 Text Extraction

The text from the SLA documents could be extracted in two ways.

5.3.1 Term Definition Extraction

Legal documents define the terms 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 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 providers operation, the extractor 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 [31]. Here the sentences would be split into noun phrase X and verb phrase Y. 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.

5.3.2 SLA Term Relationship Extractor

Using this we could populate the SLA ontology. Every sentence extracted is passed through the Stanford POS Tagger [32] and CMU Link Parser [31] to generate Parse Trees.This triple could be generated. Using Noun Phrase Extraction, we look at the Noun Phrase part of the sentence and match it to the keywords in the cloud SLA ontology. For example, we extract the relationship between service commitment and financial credit present in the form of tables and rules present in the SLA document. To do this we also use obligations and permissions specified in the document.

5.4 Extraction of data from tables

Various legal documents contain high-quality details in the form of tables. 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. After parsing the table we end up with a dictionary of elements which is the contents of the table cells.

This extracted data is encoded as RDF statements. The RDF statements are then added to the knowledge base. Once we get the dictionary with data in it, the table headers are used to associate semantics to different cell data. Over the years the way the website displays their SLA has changed. Some have hyperlinks from the main SLA page to various other pages, some have the whole document on one single page. Some prefer to keep the data in PDF format, for this we follow a different approach of extraction as it would no longer follow the rules of HTML. We strove to keep the system reusable by not using any data or documents from a local copy but instead use real-time http links 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.

5.5 Extraction of data using rules and modal logic

We explored the use of Modal Logic to extract rules present in the cloud service SLA documents. 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. One big challenge doing this was that every vendor has different vendor-specific terminology. Hence, we used a generic grammar-based extraction procedure.

We first extract sentences containing modal logic and then further extract sentences with deontic expression. Deontic logic is a field of philosophical logic that deals with obligation, permission, and related concepts. This means the sentences containing the words "must", "should", "will" impose obligations on the actor in the sentence. If the customer is the actor in such sentences she is responsible for the thing being talked about in the sentences has a few obligations.

On the other hand if the actor in this kind of a sentence is the service provider (*"Amazon* ***will*** *provide 99.8 % availability"*) then the service provider has obligations to be followed. This kind of extraction is extremely worthwhile when purchasing or monitoring services. Researchers have used deontic principles in the past to analyze policy and legal documents. Work done by Travis D.Breaux [33] 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 have considered SLA documents from cloud service providers like Google Cloud Platform, Amazon Web Services, IBM Softlayer, HP Cloud Compute, Microsoft Azure Cloud and VMware vCloud Air Services.

The Steps followed to extract modal words are as below:

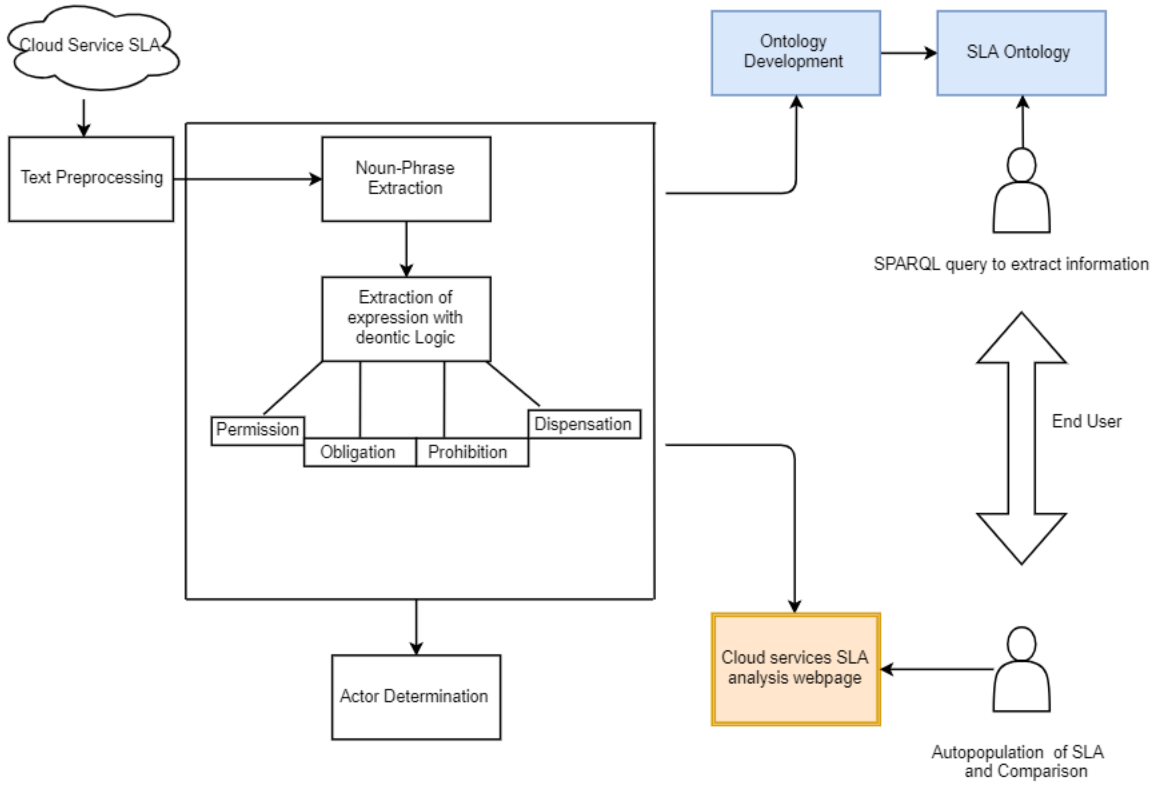


Fig. 2. Architecture to analyse Cloud Service SLA

1) Using Parts of Speech Tagging for extracting modalities: The SLA documents are first sentence tokenized, then every sentence is passed through a Stanford POS tagger. This Parts of Speech tagger tags every word present in every sentence into a part of speech. Every word gets tagged an NN if it is a noun, MO if it is a modal verb, a VB if it is verb, CD for cardinal digit and so on. We will make use of this tag to form a general grammar rule. Our grammar rule extracts sentences that match the rule. The format of grammar rule used is as below.

Positive Modalities:

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

Negative Modalities:

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

The output of this was all the sentences in SLA documents that have modal verbs in them. Modal verbs are verbs that indicates modalities like 'will', 'must', 'may' etc. This gives us the knowledge of what category among 'permission', 'obligation', 'dispensation' and 'prohibition' does the sentence belong to. The frequency of appearances of modal verbs in an SLA document is captured to give more inference. This was done in the year 2014 and then once in 2019, to observe the trend of how every cloud service under this experiment has started incorporating more modal words usage into their SLA.

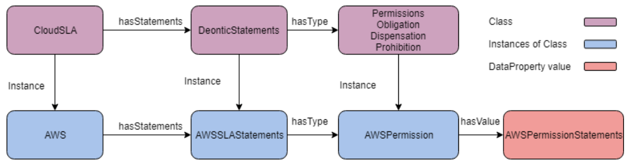


Fig. 3. Class and Relationships populated onto the knowledge graph

We also observed from the extracted sentences with modal verb, that we have two kinds of actors in the SLA documents. The noun and pronoun part of the modalities are used to define the actors associated with the modality. One would be the 'Service Provider and other the 'Customer. The sentences extracted represent obligations and permissions and mapping each Obligation/Permission with the actor will give us a clearer picture.

2) Comparison of SLAs across multiple service providers over the past few years: We have compared the number of statements in an SLA vs the number of statements containing modalities as shown in the figure 5. We have also compared the results from 2016 with the ones from 2019 to see if the companies have changed the structure of their SLA to meet the new regulations. Statements containing modalities are useful for customers as they can easily compare their rights and obligations across multiple providers, which then helps them in picking the right provider for their needs. One thing to be noted

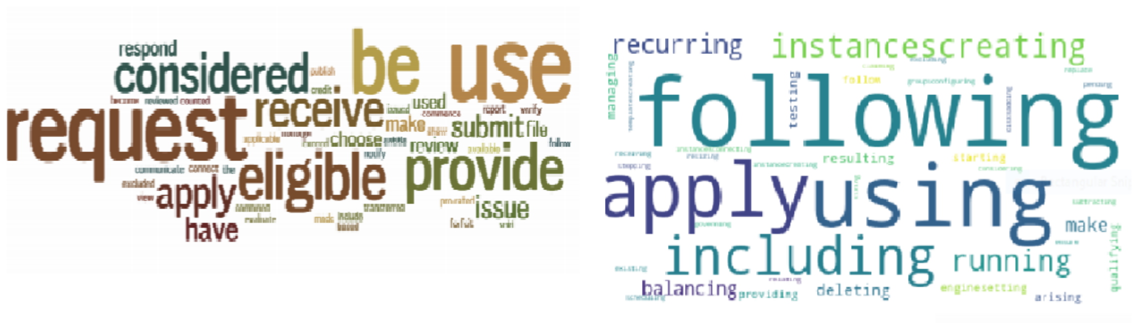


Fig. 4. Comparison of Word cloud from 2016 vs 2019 representing the verbs in the statements extracted using the Modal Logic grammatical rules.

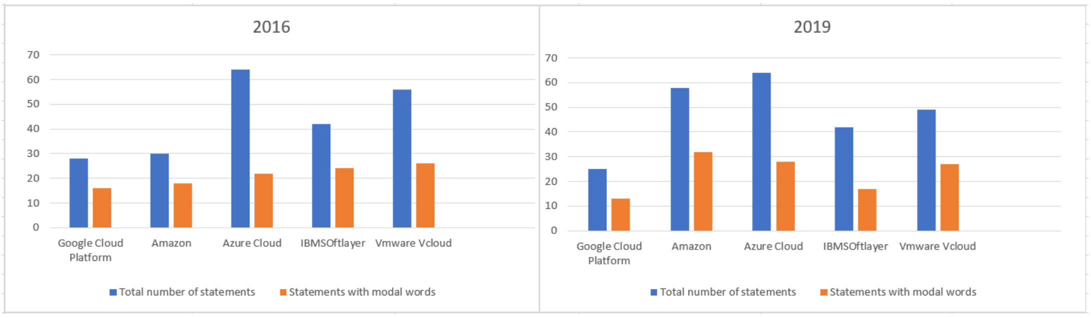
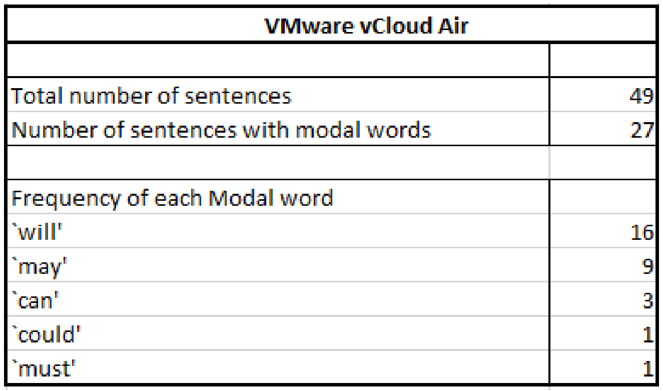


Fig. 5. Comparison of total number of sentences to number of modal words in cloud service SLA over the past 4 years

is that the extreme difference in the AWS graph from 2014 to 2019 is because of their change in pricing and SLA structure. AWS was providing exclusive SLA for EC2 which they have now chosen to merge with other AWS products like EBS and ECS for some reason. And also, HPcloud has made is slightly more difficult to access the SLAs as compared to 2016 with their new structuring of the website and SLA. We also analyzed the key term usage in SLA changes that have happened over the years, we see that the importance given to the words like 'instance creation' has increased compared to terms like 'request' as shown in figure 4 which shows the change in the trend of application of cloud services.

3) Extracting Obligations and Permissions from SLAs: We extracted and separated rules that are permissions and obligations from statements containing modalities.
   a) Permissions/Rights: Permissions are expressions that describe rights or authorizations for an entity/actor.
   b) Dispensations: Dispensations describe optional or non-mandatory statements.
   c) Obligations: Obligations define the responsibilities that an entity/actor must perform.
   d) Prohibitions: Prohibitions specify the conditions or actions which an entity is not permitted to perform

The categorization of sentences from SLA into the above categories resulted in the below findings
Example Sentence:
if this class of service is unavailable for more than 438 minutes total in a given calendar month, then you ***may*** *request a second SLA Credit.*
Type: Dispensation
Actor: User
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 ontology (the statement in this example would get populated as a DataProperty value of the subclass Dispensation under the class DeonticStatements.



VMware vCloud Air	
Total number of sentences	49
Number of sentences with modal words	27
Frequency of each Modal word	
`will'	16
`may'	9
`can'	3
`could'	1
`must'	1

Fig. 6. Analysis of VMware vCloud Service Provider

5.6 Knowledge Graph Auto-population

Our end goal is to have a knowledge base with SLA term definitions and measures of all cloud service providers. This could be queried later to get the required information. To populate the ontology we developed an automated system that would fetch the key terms and its relationship from the SLA and populate it onto the knowledge graph. This is done in the following steps:

- Fetch and 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. This is achieved using a python library 'Owlready'

For example, we extract the permission statements obtained from AWS SLA with the help of the modal verbs like 'may', 'can', 'could' etc. These permission sentences are written onto a JSON file with AWSPermission as the key and the permission statements as the value. The JSON file is then read and used to populate the knowledge graph. This is done by making the term 'AWSPermission' as an instance of the class 'Permission' under the SuperClass 'DeonticStatements'. The value in this case (the permission statements) would then be populated as the DataProperty value of the instance 'AWSPermission'. The relationship between the key and the value (AWSPermission and Permission Statements in AWS SLA) would be 'hasPermissionStatements'. The relationship being populated onto the knowledge graph is as shown in figure 3

5.7 Question and Answering system for management of cloud SLA

Once the ontology was created we developed a Legal Question and Answering system that can be used by cloud consumers to automatically analyze various cloud SLA documents using simple natural language questions. There are two types of question and answering system, open domain and closed domain system. A closed domain system is one that deals with a specific domain and can exploit domain-specific knowledge. This is much easier compared to the open domain system where the system can handle any kind of questions and relies on general ontology. In earlier work, semantically rich ontology is created which is now queried to get exact answers. Using the question and answering system the user would be able to enter a query regarding a cloud service, we would then parse the query, link the entities, and extract information from the corresponding knowledge graph. The extracted information is displayed in the form of text, graph or tables. The first step to doing this would be to build a back-end to this system would be to create ontology like the one that we have done, which uses OWL to represent the cloud SLA in the public domain. The user can either select a cloud service name to analyze the properties or type in a natural language query. The purpose of this framework is to present to the user, main components of an SLA such as definitions, metrics, permissions, and obligations which are stored in the knowledge graph and provides a user the ability to query and reason over them. The question answering system has two parts to it

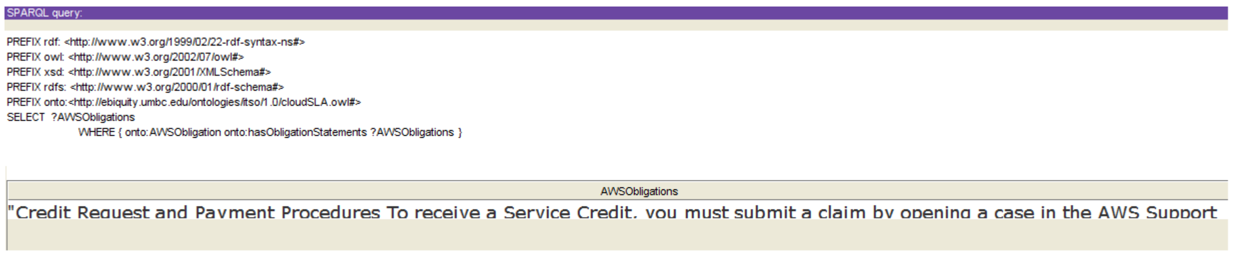
1) The question processing module- In this module we let the user enter a question in natural language. QA systems have gained widespread application as they provide a break from keyword-based query systems to a more Natural Language question-based system. Each question could be of four different types what, where, how, and when. We parse the string using Stanford pos tagger. This is then used to create tree bank for questions phrasal structure. This structure is used to create template-based SPARQL [24] queries.

2) Answering module- A mapping between the question and the SPARQL query is made to extract relevant data from the knowledge graph. The sentences once parsed and tagged, now have tags like NNP, VB, etc. The tags of interest could be NNP (proper singular noun) and NN. The words with tags NNP would be the names of the cloud service provider in our case. This tells which provider the question is concerned about 'amazon, 'Google cloud platform etc. example- 'What is amazon's uptime'. This sentence would have NNP or NNPS (proper plural noun) tag on the word amazon. The next tag of interest would be 'NN' which would give us the properties of the cloud provider like 'availability, 'uptime, 'service credit etc. These two important nouns once identified are then used to populate template SPARQL [24] queries. The next task would be to determine which of the 4 types of questions does the current question belongs to, 'What in the question would mean that it deals with definition. 'Where in the question means that it deals with location, 'When deals with temporal aspects and 'How deals with description. A query like *"What is Amazon's uptime?"* extracts the service provider as 'Amazon, the property of interest as 'uptime and the query is of type 'what. SPARQL [24] is a semantic query language with which we can query the data stored in the knowledge graph in RDF format. We used the Apache Jena Framework [30] for running the SPARQL queries. One with basic SPARQL query knowledge would be able to extract the information from the ontology directly as shown in the figure 9.

Here the user can compare the permissions of various service providers side by side, similarly, the obligations, dispensation, and prohibitions can also be compared to make a calculated decision.

5.8 Evaluation and Validation

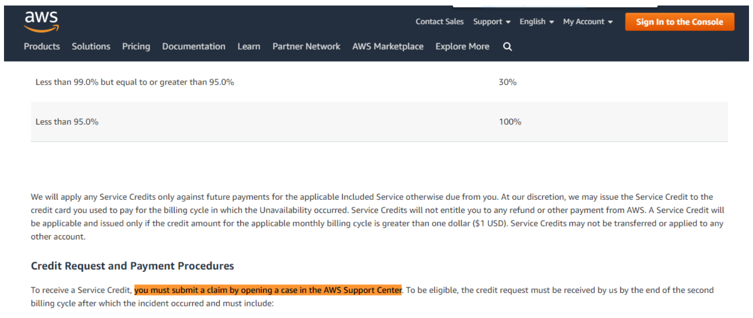
We evaluated this system by asking different individuals to type in the question they would want to know about cloud services. The questions were like 'What is amazon's availability', 'What would be the uptime of Microsoft Azure?' etc. These questions are included and handled in the question and answering system that has been developed. This gave us a 95 % success in answering the questions. Individuals wanted to know for example what they are obliged to when they purchase AWS services and our system has been able to answer just that. For example, to find what the obligations are in AWS SLA, we query the ontology using the query shown in the figure 7. We were able to cross verify and confirm the result with a manual reading of the SLA from the AWS website as shown in figure 8.


SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX onto:<http://ebiquity.umbc.edu/ontologies/itso/1.0/cloudSLA.owl#>
SELECT ?AWSObligations
        WHERE { onto:AWSObligation onto:hasObligationStatements ?AWSObligations }
```

AWSObligations
"Credit Request and Payment Procedures To receive a Service Credit, you must submit a claim by opening a case in the AWS Support

Fig. 7. Query the ontology for obligations in AWS SLA



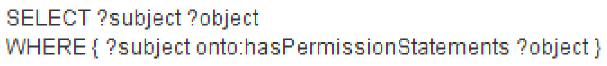
Less than 99.0% but equal to or greater than 95.0%	30%
Less than 95.0%	100%

We will apply any Service Credits only against future payments for the applicable Included Service otherwise due from you. 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. Service Credits will not entitle you to any refund or other payment from AWS. 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). Service Credits may not be transferred or applied to any other account.

Credit Request and Payment Procedures

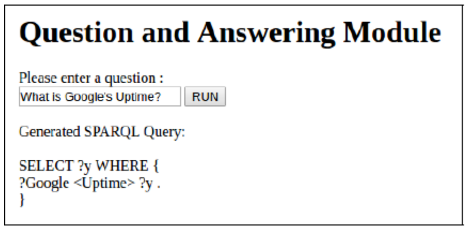
To receive a Service Credit, you must submit a claim by opening a case in the AWS Support Center. To be eligible, the credit request must be received by us by the end of the second billing cycle after which the incident occurred and must include:

Fig. 8. Manual verification with the SLA available on the service provider's webpage



```
SELECT ?subject ?object
WHERE { ?subject onto:hasPermissionStatements ?object }
```

Fig. 9. SPARQL query to extract statements having permissions


Question and Answering Module

Please enter a question :

Generated SPARQL Query:

```
SELECT ?y WHERE {
?Google <Uptime> ?y .
}
```

Fig. 10. Question and Answer User Interface which takes NLP as input and the corresponding SPARQL query is generated

6 Conclusion and Future Work

Currently, the cloud service SLA is text documents that require manual effort to read through and choose the right service. We have extracted the text from these documents and built a publicly available automatic extraction and monitoring of cloud service level agreements system, which is done with the help of semantic web technologies and text mining techniques involving the usage of OWL and RDF graph. In this paper, we have described in detail the cloud SLA ontology 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 document, that are available in the public domain.

For our initial study, we selected four providers that are the main providers of cloud-based services. In the following work we have identified and extracted modalities from SLA documents. Using which we have extracted the modal verbs and their associated actors. With this information, we then classify the modal verbs as permissions and obligations. Each permission/obligation is associated with specific actors. The actors here are 'Customers' and 'Service Providers'. 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 on expanding the number of service providers we are checking. We also intend to do the automation across domains rather than just cloud service providers. We also intend on exploring newer 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. Our question and answering system is closed domain for now and we intend on making it open domain. As part of future work, 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] 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.
[3] 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.
[4] "International organization for standardization's iso/iec dis 19086." [Online]. Available: http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=67545
[5] "Cloud computing service metrics description." [Online]. Available: http://www.nist.gov/itl/cloud/upload/RATAX-CloudServiceMetricsDescription-DRAFT-20141111.pdf
[6] "Federal risk and authorization management program (fedramp)." [Online]. Available: http://csrc.nist.gov/groups/SMA/ispab/documents/minutes/2012-02/feb3_fedramp_ispab.pdf
[7] "Resource description framework (rdf)." [Online]. Available: http://www.w3.org/RDF/
[8] "Owl web ontology language." [Online]. Available: http://www.w3.org/TR/owl-features/
[9] E. Christensen and G. Meredith, "Web services description language (wsdl) 1.1," 2001.
[10] M. Turner, D. Budgen, and P. Brereton, "Turning software into a service." *Computer.*, vol. 36, no. 10, pp. 38–44, 2003.
[11] 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.
[12] 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.
[13] 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.
[14] H. Skogsrud, B. Benatallah, and F. Casati, "Trust-serv: model-driven lifecycle management of trust negotiation policies for web services," in *Proceedings of the 13th international conference on World Wide Web*. ACM, 2004, pp. 53–62.
[15] Y. Yao, F. Yang, and S. Su, "Flexible decision making in web services negotiation," in *Artificial Intelligence: Methodology, Systems, and Applications*. Springer, 2006, pp. 108–117.
[16] 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.
[17] 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.
[18] K. Barker and N. Cornacchia, "Using noun phrase heads to extract document keyphrases," in *Advances in Artificial Intelligence*. Springer, 2000, pp. 40–52.
[19] 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.
[20] T. N. C. S. Bhagavatula and D. Downey, "tabel: Entity linking in web tables," in *The Semantic Web-ISWC 2015. Springer*, 2015, p. 425441.
[21] L. Kagal and T. Finin, "Modeling conversation policies using permissions and obligations," *Autonomous Agents and Multi-Agent Systems*, 2006.
[22] 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.
[23] S. Kumar and R. B. Mishra, "An approach to multi-attribute negotiation between semantic web services," *International Journal of Web Engineering and Technology*, vol. 5, no. 2, pp. 162–186, 2009.
[24] "Sparql 1.1 overview." [Online]. Available: http://www.w3.org/TR/sparql11-overview/
[25] "Vmware service level agreement." [Online]. Available: https://www.vmware.com/support/vcloud-air/sla/
[26] "Amazon service level agreement." [Online]. Available: https://aws.amazon.com/compute/sla/
[27] "Microsoft azure sla." [Online]. Available: https://azure.microsoft.com/en-us/support/legal/sla/virtual-machines/v1_8/
[28] "Google compute engine sla." [Online]. Available: https://cloud.google.com/compute/sla
[29] "Service level agreement for ibmsoftlayer." [Online]. Available: https://www.softlayer.com/sites/default/files/sla.pdf
[30] "Jena apache fuseki: serving rdf data over http,." [Online]. Available: http://jena.apache.org/documentation/serving_data/index.html
[31] "Link grammar." [Online]. Available: http://www.link.cs.cmu.edu/link/
[32] "Stanford lexicalized parser." [Online]. Available: http://nlp.stanford.edu/software/lex-parser.shtml
[33] 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.

Dr. Karuna P. Joshi is an Assistant Professor of Information Systems at the University of Maryland, Baltimore County. She received her Ph.D. in Computer Science from UMBC. Her research is focused on Cloud Automation and Security,Data Science and Health IT. She has been awarded the prestigious IBM PhD fellowship.She also has over 15 years of industrial experience, primarily as an IT project manager.She worked at the international Monetary Fund for nearly a decade.

Divya Natolana Ganapathy is a Master's student at the University of Maryland, Baltimore County. Her primary research focus is on application of Data Science on various text data sources. She is working on developing techniques to automate Cloud Service Level Agreements (SLAs) and Privacy Policies. Prior to masters she worked with HP and Nokia for 4 years.

Sudip Mittal is an Assistant Professor at Computer Science, University of North Carolina Wilmington.Hr received a Ph.D. in computer science form UMBC.His primary research interests are cybersecurity and artificial intelligence. His goal is to develop the next generation of cyber defense systems that help protect various organizations and people.He has also previously worked with Accelerating Cognitive Cyber Security Research Lab (ACCL), Ebiquity Research Lab, Center for Hybrid Multicore Productivity Research (CHMPR) and Cybersecurity Education and Research Centre (CERC@IIITD).