Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

Towards development of Intent Driven Networking(IDN)

Kumar Shubham* | Dr.Shashwati Banerjea

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING, MOTILAL NEHRU NATIONAL INSTITUTE OF TECHNOLOGY(MNNIT) ALLAHABAD , PRAYAGRAJ,211004, UTTAR PRADESH, INDIA

Correspondence
*Kumar Shubham, Email: shubhamcsed@gmail.com

Abstract

With the rapid growth of technology in networking arena there are higher demands for performance. In-order to address these growing demands of consumers, Software Defined Networking (SDN)[1] is gaining importance day by day. SDN provides network flexibility, customization and optimized way of traffic management in the network by decoupling of control and packet forwarding planes. Though SDN model provides benefits but still configuration and management of network is hectic for network administration. Therefore, we require an advanced model of SDN model in-order to solve these issues. This model is known as Intent Driven Network (IDN) model. Ideally, IDN[2,3] should allow operators to express their intents in natural language without requiring knowledge of low-level configurations, eliminating the need for learning new languages. In this paper we propose an IDN model with RYU as SDN controller. Intents are specified in natural languages. In order to process and perform cleaning up of text we have made use of the capabilities of NLP. With reduced intent, a specific task is identified to be performed. The task is then send to RYU controller[4] in JSON format to be performed in real time. We also introduce the concepts of contradiction detection in order to identify any previous deployed intents on hosts and switches. For contradiction detection different machine learning algorithms and upon training and testing of different models we observe that decision tree classifier gives the best results. To our knowledge, this is the first work that provides such interface between network users (in the form of natural language) and RYU SDN controller along with contradiction detection.

KEYWORDS:
SDN, IDN, RYU controller, Natural Language Processing, Contradiction Detection

1 | INTRODUCTION

Every year, organizations spend billions of dollars for network management. This includes configuration of devices, their regular maintenance and troubleshooting.

Research efforts are being made to make the job of network management easy. Software Defined Networking[5] is one such effort in this direction. The separation of control and data planes allow virtual management of switches and flows. The controllers use south bound interfaces to establish flow rules in network devices. The north bound APIs are used for network management.

The APIs used for programming SDN are low-level, making it difficult to enforce network wide policies. Furthermore, in order to deploy policies in large heterogeneous networks, the operators have to learn different vendor specific languages to configure the network elements. Also, SDN mostly focuses on packet forwarding and does not consider issues that must be implemented

on middleboxes or custom hardware.
Intent based Networking (IBN) allows operators to specify network policies in high level language without considering the low level details required for programming the network[6]. The intent framework translates the high level service requests defined in descriptive language into a prescriptive language. For example, the network operator may specify a service request of the form "allow traffic between host A and Host B". The Intent framework reads the request, parses it and converts it into a prescriptive form such as src: 172.31.100.14;dst: 172.31.100.28;keyword: allow, host.
The research efforts in this direction focus on developing intent frameworks, compilers and languages. Majority of these efforts enable composition of high-level policies, deployment of intents in SDN controllers and abstraction for network operators.
There are two key challenges in intent deployment. Correct information extraction and conflict detection. Information extraction is difficult because a single intent can be specified in many different forms by using different expressions and terms. The second challenge is detection of conflicts. There are two types of conflicts: conflict and contradiction. Both of them occurs from incremental deployment of intents. Conflict arises when two different policies have opposite operations. As an example, consider that there are two requirements submitted by two different network operators. The first request allows packets on a firewall. On the other hand, the second request asks for dropping of the packets on the same firewall. Contradiction is a state where the description contradicts with previously deployed policies, but the result of the operations do not conflict. For example, one policy asks to drop certain packets on a firewall and another policy asks to re-route the same packets, so as to bypass the firewall.
The aim of the current work is to develop a system that understands the network intents specified in English language, check for conflicts with existing policies and translate the conflict-free intents into machine understandable commands. Our system consists of four stages. NLP phase, Task Identification phase, Conflict Detection phase and Response Generator phase. In the first phase text cleaning is performed with NLP tools, in the second phase task to be performed is identified with help of keywords. Examples of entities include keywords, endpoints, middleboxes etc. Next, the system checks to see if the intent conflicts with existing policies. We have used machine learning models to detect conflicts. In the third phase, the extracted intents are translated into a JSON format which is sent to controller for execution.The fourth stage is responsible for parsing and sending the request to controller and generating the desired responses.
As a proof-of-concept, we have implemented the proposed Intent based framework over Ryu controller.
Specifically, the contribution of the proposed work are as follows:

- We build a tool that parses user requirements given in English language and constructs network tasks[6].
- The network tasks are analyzed to check for conflicts. We have trained the framework using different machine learning models to detect conflicts among existing intents and new intents[7].

Rest of the paper is organized as follows. Section 2 presents an introduction to proposed work. Section 3 presents detailed study of all the phases of the model along with analysis of contradiction detection . Section 4 deals with the model interaction and showing the results of the intents. Section 5 deals with future work and finally section 6 presents the conclusion of the proposed work.

2 | PROPOSED WORK

The complete architecture of the proposed model is shown in below figure. As shown in the figure, user specifies the intent initially. This intent is then passed through NLP processing pipeline. The major steps involved in the pipeline are tokenization, case conversion, text filtration and lemmatization.

After the intent passes through the above mentioned steps it is now the time to parse the intent and finally convert it into a form that could be understood by the RYU controller. The major steps involved in parsing are task identification phase and JSON format composition phase which are explained briefly in implementation details section. The output from the NLP processing pipeline goes to the task identification phase. In this phase, network task is identified with help of certain keywords which are defined manually. With the help of these keywords, 45 different intents can be processed easily. After successful identification of task, we now move on to the next phase i.e. contradiction detection. Contradiction detection phase is a conditional phase and it is not involved in every intent processing. The condition here is the involvement of certain keywords such as [block, host], [enable, host], etc. If these keywords are involved, only then the contradiction detection phase will start else this phase is bypassed and we move directly onto the next phase i.e. JSON format composition. In the JSON format composition phase appropriate message

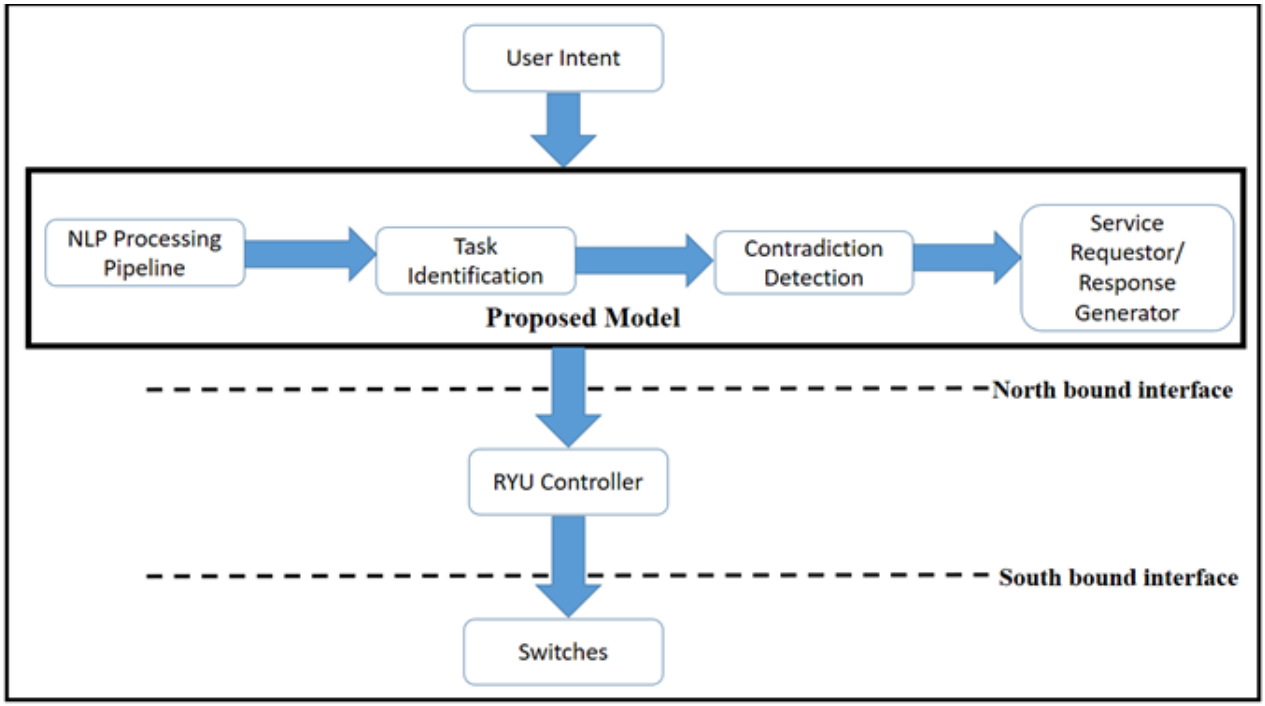

FIGURE 1 RYU based IDN model architecture

is constructed in JSON format w.r.t. the intent specified by network operators. The constructed JSON format is then send to the RYU controller as a request and the response of the request is sent to operator on the terminal window.

3 | IMPLEMENTATION DETAILS

3.1 | NLP PROCESSING PIPELINE AND USE OF NLTK PACKAGE:

We have made a python application that reads the intents of operators and then passes them over the NLP processing pipeline. For accomplishing the NLP part of our model NLTK package has been used in this application. NLP helps to bridge the gap between humans and machines by providing an efficient and convenient way of interaction in natural language.

Any intent specified by operators has to pass through the pipeline as shown in figure 2. The need for NLP processing pipeline is to remove any unwanted elements from the intent so that further processing of intent is easier. The steps involved in pipeline are following:

- Tokenization:
  Tokenization can be defined as the process of splitting a sentence, paragraph or an entire text data into smaller segments such as words. Each words, numbers or punctuation marks are called tokens. Tokens are created with the help of word boundaries. Word boundaries is the boundary between the ending point of word and starting point of next word. The generation of different tokens is an essential step for the process of Lemmatization which is performed at a later point.
  In the proposed model tokenization helps to separate out the words of the intent specified. The intent is the overall sentence that is to be processed. For an intent specified as “Get the number of Switches present in the Topology?”. After the intent passes through the tokenization phase of NLP processing pipeline the output as shown in below figure is [get, the, number, of, Switches, present, in, the, Topology,].
  For another intent specified as “Please Block a Host h1”. When this intent passes through tokenization then the output as shown in below figure is [Please, Block, a, Host, h1].
  There can be different methods that can be used to perform tokenization but in the proposed model we have used the

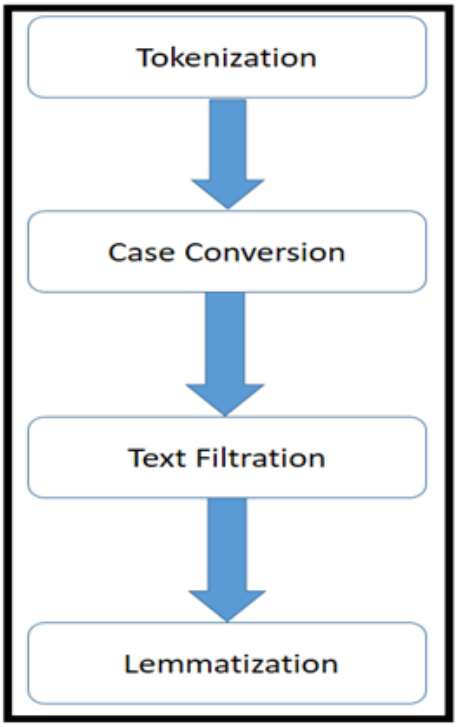

FIGURE 2 Natural Language Processing(NLP) pipeline

FIGURE 3 Intent after passing through tozenization phase

FIGURE 4 Intent after passing through tozenization phase

method of Natural Language Toolkit (NLTK) package to perform tokenization. The method named word_tokenize() in the package is used to perform word tokenization which separates the string at each space and produces the output as list of strings. In case if sentence tokenization is required which means separating the string at each full stop(.) then the method sent_tokenize() can be used.

- Case conversion:
  In this process each token that is generated previously is converted to its lower case so that the processing becomes completely case insensitive. Depending upon the requirement the tokens can also be converted to upper case. This process is effective in eliminating any errors occurring due to case mismatch. The input and the output is shown in below figure:

FIGURE 5 Figure depicting case conversion phase of NLP processing pipeline

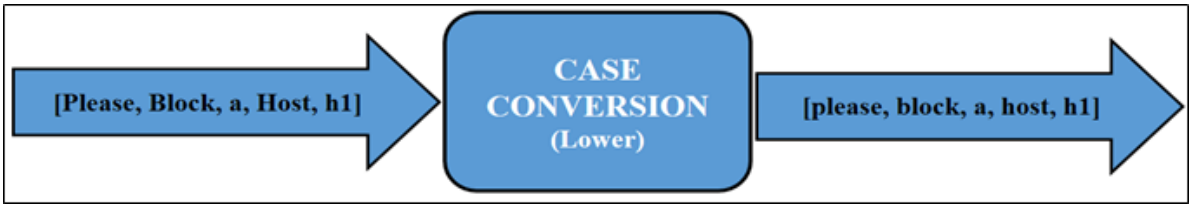

FIGURE 6 Figure depicting case conversion phase of NLP processing pipeline

- Text Filtration: This process is responsible for removing punctuation marks, special characters and stop words from tokens. There are fourteen punctuation marks in English such as full stop(.), question mark(?), comma (,), exclamation mark(!), colon (:), Semicolon (;), etc. Some of the special characters includes hash(#), dollar sign($), ampersand(), percent(%), etc. Stop words can be defined as most commonly occurring words such as a, an, the, in, etc. These words do not add any special meaning to the sentence.
  The elements such as punctuation marks, special characters and stop words are not needed to be processed as per the requirements and hence are removed. This is the most important step in the NLP processing pipeline and it is mainly responsible to reduce the overall size of the intent so that it could be processed much faster at later stages.
  Two examples of input and output from this phase is shown below:

FIGURE 7 Figure depicting input/output of text filtration phase

FIGURE 8 Figure depicting input/output of text filtration phase

- Lemmatization:
  The output of lemmatization is known as 'lemma' which is a root word. Unlike stemming where the root word can be valid or invalid in lemmatization the root word is always valid as it refers to the context of the sentence. Python NLTK package provides WordNet Lemmatizer which is very useful in generating the root word of a given word.
  In the proposed model use of lemmatization is responsible to convert all the tokens or words of the intent to its root form so that further there is no difficulty in the identification of network task. After completion of the above steps the tokens are again re-joined together and we observe an obvious reduction in the size of the intent as compared to its size originally. The reduced intent can be easily identified for specific network task details to which is discussed in the next section i.e. task identification phase.
  Input and output of lemmatization phase of NLP processing pipeline is shown below:

FIGURE 9 Figure depicting lemmatization phase of NLP pipeline

FIGURE 10 Figure depicting lemmatization phase of NLP pipeline

3.2 | TASK IDENTIFICATION PHASE:

After the output of the NLP pipeline it is now the time to identify the network task. As mentioned earlier, in order of identify a network tasks many different sets of keywords have been defined manually. As the intents are expressed naturally therefore there are many different ways of expressing the same intent. In-order to solve this every set of keywords is defined with synonyms. Table 1 shows list of different intents that could be processed by the proposed model.

S.NO.	Network Intents	Keywords	REST API method
1.	Get the number of switches present in the topology?	get, number, switch	GET
2.	Get the description of switch s1?	get, description, switch	GET
3.	Get detailed flows installed in switch s1?	get, detailed, flow, switch	GET
4.	Get all the flows installed in switch s1 with rule id?	get, flow, switch, rule	GET
5.	Please Block a Host h1	block, host	POST
6.	What are the number of switches present in the topology?	what, number, switch7	GET
7.	Deny traffic between host h2 and host h3 block,	deny,host	POST
8.	Enable host h1	enable, host	POST
9.	Get the details of switch s1?	get, details, switch	GET
10.	Allow the traffic between host h2 and host h3	allow, host	POST
11.	Block switch s1	block, switch	PUT
12.	Activate switch s1	activate, switch	PUT
13.	How many switches are present in the topology?	how, many, switch.	GET
14.	Get the overall status of all the switches?	overall, status, switch	GET
15.	Delete flows from switch	Delete	DELETE
16.	Get port status of switch s1	get, port, status, switch	GET
17.	Get aggregate flows of switch s1	get, aggregate, flow, switch	GET
18.	Show the description of switch s3	Show, description, switch	GET
19.	Show the details of switch s4	Show, detail, switch	GET
20.	Display the details of switch s5	Display, detail, switch	GET
21.	Display the description of switch s6	Display, description, switch	GET
22.	What are the details of switch s7?	What, details, switch	GET
23.	Show the detailed flow entries of switch s2	Show, detailed, flow, switch	GET
24.	Display the detailed flows of switch s4	Display, detailed, flow, switch	GET
25.	Deny the traffic between host h2 and host h5	Deny, host	POST
26.	Disable host h3	Disable, host	POST
27.	Display the aggregate flows of switch s4	Display, aggregate, flow, switch	GET
28.	Show the flow entries of switch s3 with rule id	Show, flow, switch, rule	GET
29.	Deactivate the traffic between host h6 and host h7	Deactivate, host	POST
30.	Allow the communication between host h2 and host h5	Allow, host	POST
31.	Activate host h3 to communicate	Activate, host	POST
32.	Forward the traffic between host h6 and host h7	Forward, host	POST
33.	Get the overall status of switches present in the topology	Get, overall, status, switch	POST
34.	Disable switch s4	Disable, switch	PUT
35.	Delete a flow rule from switch	Delete	DELETE

TABLE 1 Different user's intent

The behaviour of task identification phase can be seen in below figure:

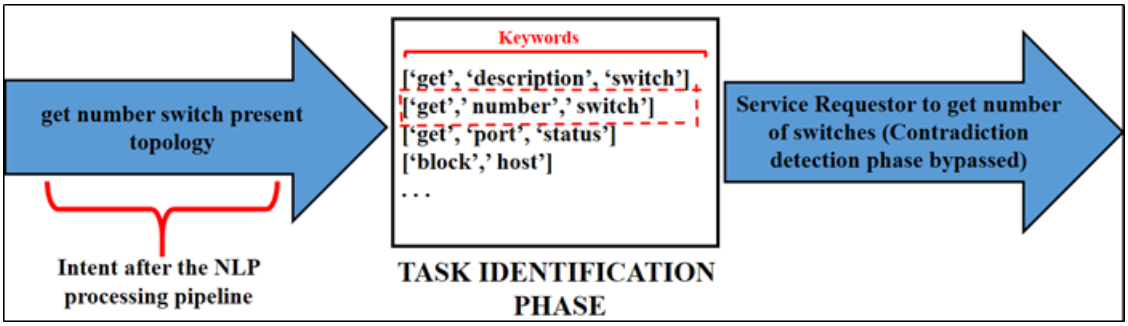

FIGURE 11 Figure depicting working of task identification phase and bypassing of contradiction detection phase

The above figure shows the working of task identification phase. The initial intent specified as "Get the number of switches present in the topology?" after passing through NLP processing pipeline becomes "get number switch present topology". The output from NLP processing pipeline acts as an input to the task identification phase. The task identification phase as mentioned above consists of manually defined sets of keyword which is eventually responsible for the recognition of task to be performed. In this case the keywords responsible for task recognition are ['get',' number', 'switch'] as shown by dashed red colour in the above figure. As there is no keywords which are concerned with blocking or enabling of hosts therefore the contradiction detection phase is bypassed and the task identification phase calls for service requestor to handle future operations concerned with getting number of switches present in the topology.
Consider another example of task identification phase shown in below figure when the next stage i.e. contradiction detection phase is not bypassed.

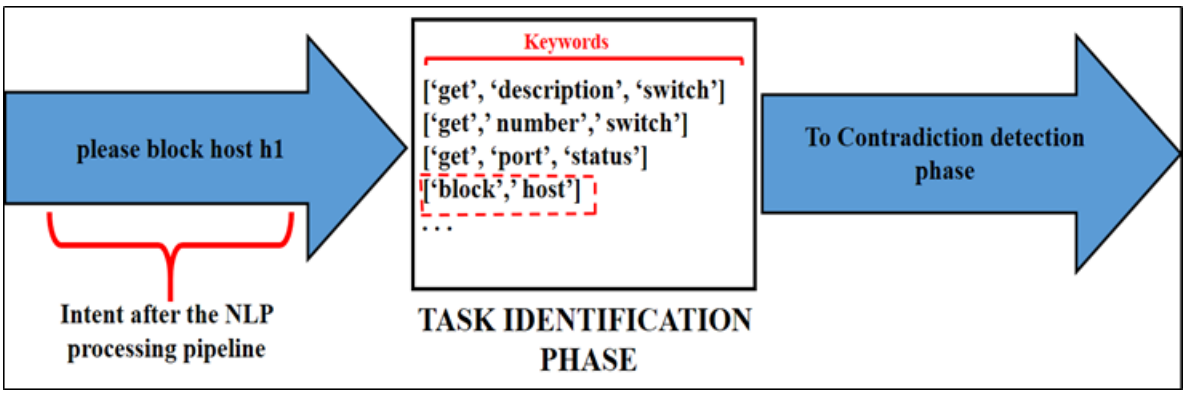

FIGURE 12 Figure depicting working of task identification phase and next phase as contradiction detection

With the reference to above figure the initial intent undergoing the NLP processing pipeline is "Please Block a Host h1". After passing through the pipeline the intent becomes "please block host h1" which is given as input to the task identification phase as shown in figure above. Task identification phase with help of keywords ['block', 'host'] identifies that blocking of certain host has to be performed and calls for contradiction detection to handle further process.

3.3 | CONTRADICTION DETECTION TECHNIQUE AND RESULTS:

As mentioned earlier contradiction detection technique will allow the network operators to detect contradictions with respect to previously deployed intents. In order to accomplish this task machine learning algorithm have been used and for detecting

contradictions mirroring is used.

To train and test our contradiction detection model we require a dataset. No such dataset is available and therefore we have composed a dataset with combination of 0's and 1's. In the dataset 0's indicates either no previous intent has been deployed for host/switch or host/switch is not enabled/blocked and 1's indicates that the host/switch is either enabled or blocked. In the dataset there are a total of 21 columns in which the first ten columns is the block status of host and next ten columns is the enable status of host finally the last column is the result whether contradiction has been detected or not. We have worked with four different machine learning models i.e. Logistic regression, Naïve Bayes, Random Forest Classifier and Decision Tree Classifier. In order to train and test the models we have used the standard 75%-25% train-test dataset split. In order to predict contradictions, we are composing a list with 21 indexes in real time. This list is used with predict methods of different models.

In-order to track the current status of hosts we have defined four arrays. Two arrays are one dimensional which are responsible to track block and enable status of one host and the other two arrays are two dimensional which are responsible to track block and enable status of two hosts. After a network task has been identified for blocking or enabling of host it first moves to the contradiction detection phase.

The important four steps involved in the contradiction detection phase are following:

- As a first step number of host involved is checked which could be either one or two.
- In the second step depending upon the request to block/enable host respected array is modified to hold the current status of host. If there is only one host involved then all the modifications will be done on one dimensional array whereas if two hosts are involved then two dimensional array is involved.
- In third step list is composed with 20 indexes with first 10 indexes are block status of host and next 10 indexes are enable status of host. The composed list is used to predict contradiction detection.
- In the fourth step the composed list is sent to the machine learning model for contradiction detection.

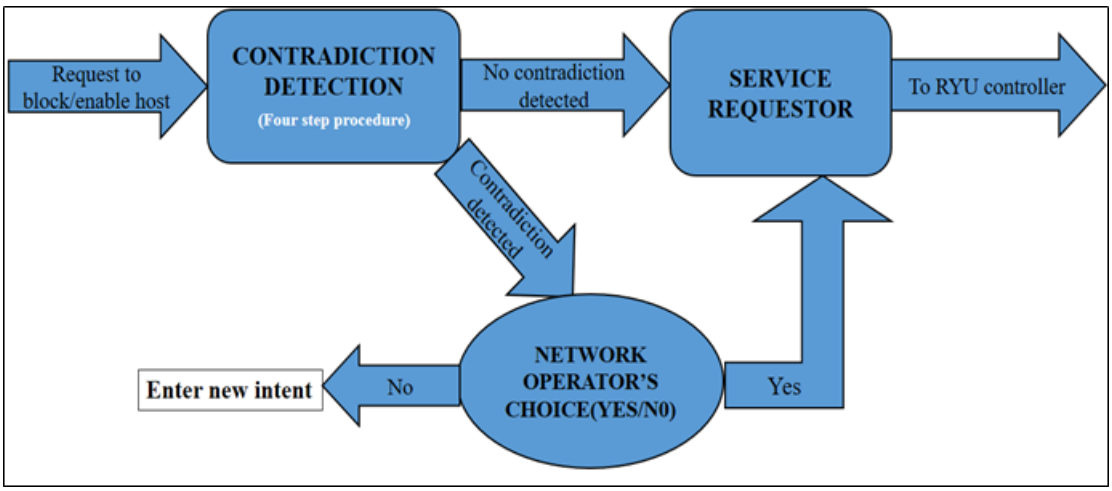

FIGURE 13 Figure depicting working of contradiction detection phase

After sending the list for contradiction detection there can be two possible outcomes. First outcome is that no contradiction could be detected and automatically service requestor is called which composes a data in JSON format and sends to RYU controller for flow installation. If contradiction is detected then the operator is prompted with a notification that contradiction has been detected and does he wants to proceed further and install the flow or not. If the operator chooses to proceed then service requestor is called which composes a data in JSON format and sends to RYU controller for flow installation process. If the operator chooses not to proceed further then service requestor phase is bypassed and the modified arrays are reverted back to hold the current status of hosts.

Consider three examples of list composition in real time as below:

- First example as list containing elements such as: 0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0 in this case while composing the list in real time contradiction will be detected as 1st index and 11th index both are 1.
- Consider second example as: 1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 in this no contradiction will be detected and our model assumes that there is a certain request to block host h1 as 1st index is 1.
- Consider third example as: 1,1,1,1,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0 in this case again no contradiction will be detected as this is the case of error. This is the case of error because our model is trained only to predict contradictions with one or at maximum two hosts therefore if contradiction is detected for more than two hosts then it becomes an error and our model predicts that no contradiction has been detected.As we are including cases of errors as well in the dataset hence its size very complex.

Consider four examples of intents as block host h1, enable host h1, block host h2 and host h3 and finally enable host h2 and host h3. These intents after passing through NLP processing pipeline and task identification phase will finally pass through the contradiction detection phase. The detailed description of every step is explained with four different intents as shown below:

- **INTENT: Deny host h1**
  - Only one host is involved here hence one dimensional arrays which keeps track of block and enable status of hosts are used.
  - This step involves modification of one dimensional arrays. Initially the status of block and enable arrays are:

h1	h2	h3	h4	h5	h6	h7	h8	h9	h10
0	0	0	0	0	0	0	0	0	0

TABLE 2 Initial block status of hosts

h1	h2	h3	h4	h5	h6	h7	h8	h9	h10
0	0	0	0	0	0	0	0	0	0

TABLE 3 Initial enable status of hosts

After the request to block host h1. Block array at index H1 is modified to 1 as shown below:

h1	h2	h3	h4	h5	h6	h7	h8	h9	h10
1	0	0	0	0	0	0	0	0	0

TABLE 4 Block status of hosts after modification

  - Now the list with 20 indexes is composed taking all of the block array and enable array. The contents of the list are: 1,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0.

– The composed list of 20 indexes is sent to the machine learning model for contradiction detection.

In this case no contradiction will be detected and service requestor is called automatically to compose the data in JSON format and send to RYU controller for flow installation process.

- **INTENT: Allow host h1**

  – Here again only one host is involved and hence one dimensional arrays will be used.
  – Initial block and enable status of host from arrays are:

h1	h2	h3	h4	h5	h6	h7	h8	h9	h10
1	0	0	0	0	0	0	0	0	0

TABLE 5 Current block status of hosts

h1	h2	h3	h4	h5	h6	h7	h8	h9	h10
0	0	0	0	0	0	0	0	0	0

TABLE 6 Current enable status of hosts

After the request to enable host h1. Enable array indexed at H1 is modified as:

h1	h2	h3	h4	h5	h6	h7	h8	h9	h10
1	0	0	0	0	0	0	0	0	0

TABLE 7 Enable status of hosts after modification

  – Now the list with 20 indexes are composed as: 1,0,0,0,0,0,0,0,0,0, 1,0,0,0,0,0,0,0,0,0
  – This list is then sent to the machine learning model for contradiction detection.

We observe that contradiction has been detected in this case and operator is given a choice that does he wants to proceed and install the flow or roll back from the current state.

- **INTENT: Deactivate the traffic between host h2 and host h3.**

  – Here two hosts are involved and hence two dimensional arrays are used.
  – Initially block and enable status of hosts are:
    After the request to block host h2 and host h3. Block array at indexes (2,3) and (3,2) will be modified to 1 and enable array will remain unchanged.
  – Now two list with 20 indexes are composed for strict verification of contradiction detection. Overall contradiction will be detected only if the machine learning model predicts contradiction detection from both lists. Here, since host h2 and host h3 are involved hence the first list involves the 2nd row of block and enable arrays while the 2nd list involves the 3rd row of block and enable arrays. The contents of first list are: 0,0,1,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0. Contents of second list are: 0,1,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0.

	h1	h2	h3	h4	h5	h6	h7	h8	h9	h10
h1	0	0	0	0	0	0	0	0	0	0
h2	0	0	0	0	0	0	0	0	0	0
h3	0	0	0	0	0	0	0	0	0	0
h4	0	0	0	0	0	0	0	0	0	0
h5	0	0	0	0	0	0	0	0	0	0
h6	0	0	0	0	0	0	0	0	0	0
h7	0	0	0	0	0	0	0	0	0	0
h8	0	0	0	0	0	0	0	0	0	0
h9	0	0	0	0	0	0	0	0	0	0
h10	0	0	0	0	0	0	0	0	0	0

TABLE 8 Initial block status of hosts

	h1	h2	h3	h4	h5	h6	h7	h8	h9	h10
h1	0	0	0	0	0	0	0	0	0	0
h2	0	0	0	0	0	0	0	0	0	0
h3	0	0	0	0	0	0	0	0	0	0
h4	0	0	0	0	0	0	0	0	0	0
h5	0	0	0	0	0	0	0	0	0	0
h6	0	0	0	0	0	0	0	0	0	0
h7	0	0	0	0	0	0	0	0	0	0
h8	0	0	0	0	0	0	0	0	0	0
h9	0	0	0	0	0	0	0	0	0	0
h10	0	0	0	0	0	0	0	0	0	0

TABLE 9 Initial enable status of hosts

	h1	h2	h3	h4	h5	h6	h7	h8	h9	h10
h1	0	0	0	0	0	0	0	0	0	0
h2	0	0	1	0	0	0	0	0	0	0
h3	0	1	0	0	0	0	0	0	0	0
h4	0	0	0	0	0	0	0	0	0	0
h5	0	0	0	0	0	0	0	0	0	0
h6	0	0	0	0	0	0	0	0	0	0
h7	0	0	0	0	0	0	0	0	0	0
h8	0	0	0	0	0	0	0	0	0	0
h9	0	0	0	0	0	0	0	0	0	0
h10	0	0	0	0	0	0	0	0	0	0

TABLE 10 Block status after modification

  - Both the composed lists are sent to the machine learning model for contradiction detection.

In this case there is no contradiction detected as there is no previous intent specifying enabling of host h2 and host h3.

- **INTENT: Allow host h2 and host h3**
  - In the specified intent two hosts are involved and therefore both the 2D arrays will be involved.
  - The status of both 2D arrays as of now are shown below:

	h1	h2	h3	h4	h5	h6	h7	h8	h9	h10
h1	0	0	0	0	0	0	0	0	0	0
h2	0	0	1	0	0	0	0	0	0	0
h3	0	1	0	0	0	0	0	0	0	0
h4	0	0	0	0	0	0	0	0	0	0
h5	0	0	0	0	0	0	0	0	0	0
h6	0	0	0	0	0	0	0	0	0	0
h7	0	0	0	0	0	0	0	0	0	0
h8	0	0	0	0	0	0	0	0	0	0
h9	0	0	0	0	0	0	0	0	0	0
h10	0	0	0	0	0	0	0	0	0	0

TABLE 11 Current block status of hosts

	h1	h2	h3	h4	h5	h6	h7	h8	h9	h10
h1	0	0	0	0	0	0	0	0	0	0
h2	0	0	0	0	0	0	0	0	0	0
h3	0	0	0	0	0	0	0	0	0	0
h4	0	0	0	0	0	0	0	0	0	0
h5	0	0	0	0	0	0	0	0	0	0
h6	0	0	0	0	0	0	0	0	0	0
h7	0	0	0	0	0	0	0	0	0	0
h8	0	0	0	0	0	0	0	0	0	0
h9	0	0	0	0	0	0	0	0	0	0
h10	0	0	0	0	0	0	0	0	0	0

TABLE 12 Current enable status of hosts

After the request to enable host h2 and host h3 the enable array will be modified at index positions (2,3) and (3,2) to 1.

	h1	h2	h3	h4	h5	h6	h7	h8	h9	h10
h1	0	0	0	0	0	0	0	0	0	0
h2	0	0	1	0	0	0	0	0	0	0
h3	0	1	0	0	0	0	0	0	0	0
h4	0	0	0	0	0	0	0	0	0	0
h5	0	0	0	0	0	0	0	0	0	0
h6	0	0	0	0	0	0	0	0	0	0
h7	0	0	0	0	0	0	0	0	0	0
h8	0	0	0	0	0	0	0	0	0	0
h9	0	0	0	0	0	0	0	0	0	0
h10	0	0	0	0	0	0	0	0	0	0

TABLE 13 Enable status of hosts after modification

– The two lists after composition are:
List1: 0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0.
List2: 0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0.
The composed list is now ready to be sent to machine learning model.

– When both these lists are sent to the machine learning model for contradiction detection one by one then contradiction is detected in both the lists and hence there is an overall contradiction detection. Contradiction is detected as there was a previous intent specifying blocking of host h2 and host h3 and then request to enable these hosts are specified.

After the contradiction is detected the operator as usual is given two choices whether he wants to proceed with installation of new flow or rollback and take no actions.
As an initial step we are able to detect contradictions among 10 hosts and 10 switches.

3.3.1 | Analysis and results

S.No.	Model Name	Accuracy Score	Precision Score	F1 Score	Recall Score
1.	Logistic Regression	0.809	0.650	0.556	0.553
2.	Naïve Bayes	0.809	0.650	0.556	0.553
3.	Random Forest Classifier	0.977	0.983	0.960	0.941
4.	Decision Tree Classifier	0.999	0.999	0.998	0.998

TABLE 14 Evaluated results for contradiction detection

The above table shows the evaluation results of different machine learning models. The machine learning models are judged on the basis of accuracy score, precision score, f1 score and recall score. On analysing these results we come the fact that decision tree classifier is the best model for contradiction detection.

After analysing the machine learning models with four parameters mentioned above these models are tested with ROC curve. The ROC curve for different models are shown below:

- Logistic Regression:

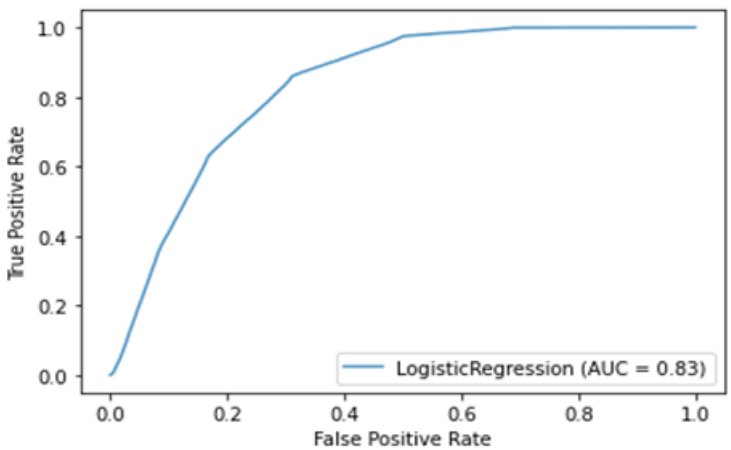

FIGURE 14 ROC curve for Logistic Regression model

- Random Forest Classifier:

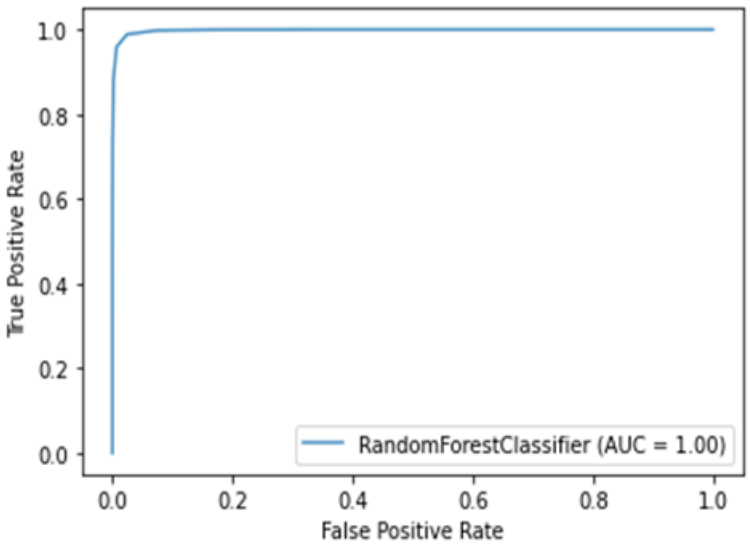

FIGURE 15 ROC curve for Random Forest Classifier model

- Naive Bayes:

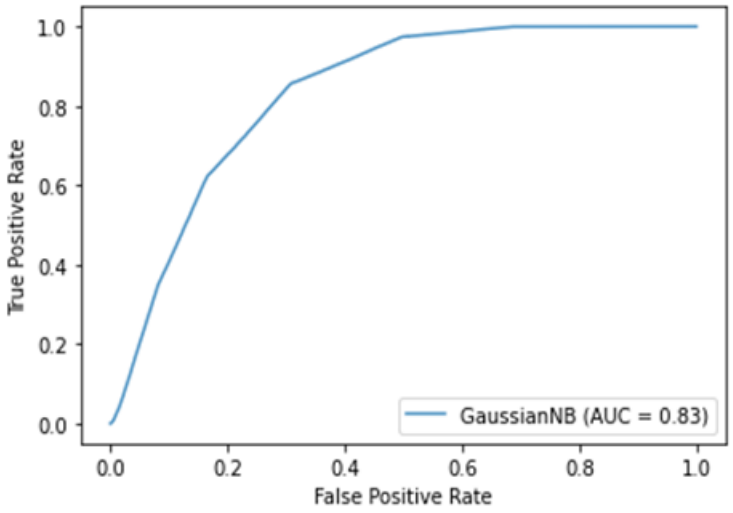

FIGURE 16 ROC curve for Naive Bayes model

- Decision Tree Classifier:

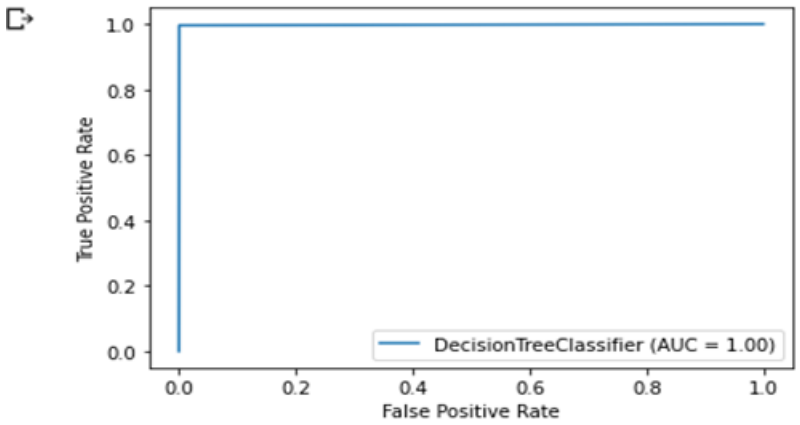

FIGURE 17 ROC curve for Decision Tree Classifier model

On observing the above ROC curve for different models we found that decision tree classifier has the best area under curve. The area under the curve for logistic regression and Naïve Bayes machine learning models are less and it should not be preferred for contradiction detection.

3.4 | SERVICE REQUESTOR AND RESPONSE GENERATOR:

After the completion of above phases, we are finally assured what task has to be performed by the RYU controller. This is the phase where parsing takes place and the intent specified in natural language is finally converted to a form that is understood by the RYU controller. For this phase corresponding to each network task a service handler function is defined that will be called and it will be responsible for creating a request in desired format and then sending it to the controller as a request for the service. After sending the request for a service, controller generates responses as per the requirements specified in the intent. The response of the controller is unordered and difficult to read. Therefore, our function modifies the response of the controller and then produces the final output on the screen which is ordered and readable.
All the GET and PUT requests that is sent to the RYU controller is directly sent to a specified URL without requiring the need to compose the data in JSON format. The requests that are send as POST and DELETE are sent to RYU on a specified URL along-with the data in JSON format. There are two types of POST request that could be sent to RYU controller and these are request to block host and request to enable host. There is one DELETE request that could be sent to controller for deletion of flow rules installed in switches. Consider an example of blocking two hosts h2 and h3, for this the data in JSON format that will be sent is:

{
“priority”: 101,
”nw_proto”: ICMP,
“nw_src”: 10.0.0.2,
“nw_dst”: 10.0.0.3,
“actions”: DENY
}

The URL on which this data will be sent is: http://localhost:8080/firewall/rules/all
Consider another example of enabling two hosts h1 and h4, for this the data in JSON format that will be sent is:

{
“priority”: 102,
“nw_proto”: ICMP,
“nw_src”: 10.0.0.1,
“ nw_dst”: 10.0.0.4,
“actions”: ALLOW
}

Along with this another data in JSON format that will be sent is:

{
“priority”: 102,
“nw_proto”: ICMP,
“nw_src”: 10.0.0.4,
“ nw_dst”: 10.0.0.1,
“actions”: ALLOW
}

The URL on which these both data in JSON format will be sent is same as above i.e. http://localhost:8080/firewall/rules/all In-order to get the I.P. address of hosts we have made a mapping of host to I.P. address and all the I.P. addresses are fetched from this mapping. Considering the delete operation as third example where the data will be sent as JSON format. Suppose there is request to delete flow rule with id 3 from switch s1. For this the data in JSON format will be:
{
“rule_id”: 3
}
The delete request will be sent on URL:
http://localhost:8080/firewall/rules/ 0000000000000001.
In this URL the switch id must always be in 16-digit hexadecimal format.
The different GET requests and their respected URL are given below:

- Getting the number of switches present in the topology:
  http://localhost:8080/stats/switches
- Getting the hardware and software description of switch:
  http://localhost:8080/stats/desc/swid
  swid stands for switch id.
- Getting detailed flows of switch:
  http://localhost:8080/stats/flow/swid
- Getting flows installed in switches with rule id’s:
  http://localhost:8080/firewall/rules/SWID
  Here, the SWID also stands switch id but it must be in 16-digit hexadecimal format.
- Getting the status of all switches:
  http://localhost:8080/firewall/module/status
- Getting the port status of switch:
  http://localhost:8080/stats/port/swid
- Getting aggregate flows of switch:
  http://localhost:8080/stats/aggregateflow/swid

After sending these requests to RYU controller we get response from the controller which is processed and arranged in ordered manner and then provided to the operator.

4 | MODEL INTERACTION:

This section deals with interaction with proposed model. The response of different intents can be easily seen in this section. As an example linear topology with 7 switches and 7 hosts as shown in the figure below is taken for viewing the response after processing of user's intents.

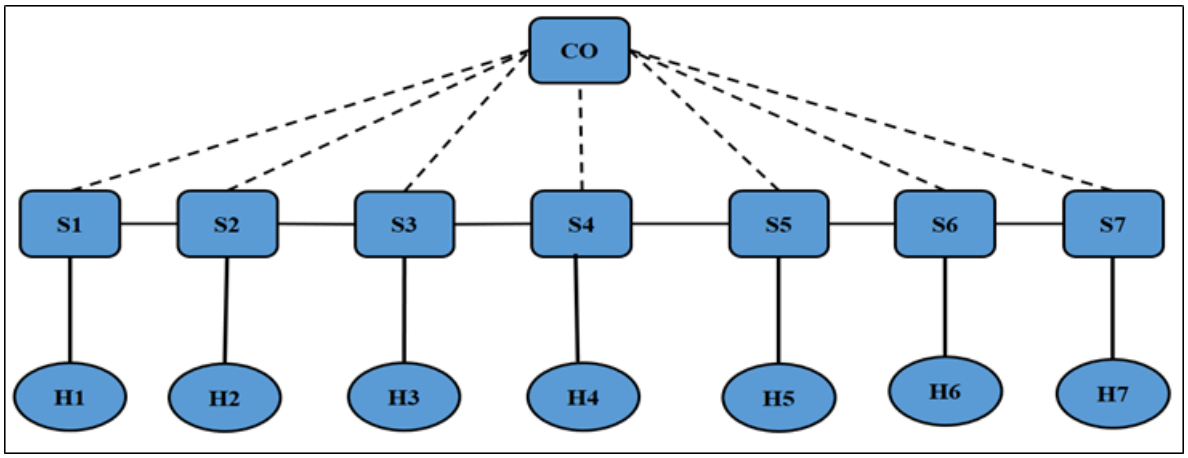

FIGURE 18 Linear Topology

- Intent to get the number of switches present in the topology.

  – Please enter new intent or enter exit
  How many switches are present in the topology?
  RESPONSE CODE 200
  Switch IDs present are: [6, 1, 2, 5, 4, 7, 3]
  Number of switches are: 7

- Intent to get the hardware and software details of switch s7.

  – Please enter new intent or enter exit
  What are the details of switch s7?
  RESPONSE CODE 200
  Manufacturer description ::Nicira, Inc.
  Hardware description ::Open vSwitch
  Software description ::2.13.1
  Serial number ::None
  Datapath description ::s7

- Intent to block the traffic of host h1.

- Please enter new intent or enter exit
  Deny host h1
  RESPONSE CODE 200

  Mininet result after installation of above flow rule in switches
  mininet> pingall
  *** Ping: testing ping reachability
  h1 -> X X X X X X
  h2 -> X h3 h4 h5 h6 h7
  h3 -> X h2 h4 h5 h6 h7
  h4 -> X h2 h3 h5 h6 h7
  h5 -> X h2 h3 h4 h6 h7
  h6 -> X h2 h3 h4 h5 h7
  h7 -> X h2 h3 h4 h5 h6
  *** Results: 28% dropped (30/42 received)

- Intent to get the detailed flow entries in switch s1.

  - Please enter new intent or enter exit
    Get the detailed flows installed in switch s1
    Getting you the flow entry of switch with ID:1
    RESPONSE CODE 200
    Priority :65534
    Cookie :0
    Idle Timeout :0
    Hard Timeout :0
    Actions :['OUTPUT:NORMAL']
    Match :'dl_type': 2054
    Byte Count :2142
    Duration Sec. :3794
    Duration nSec. :521000000
    Packet Count :51
    Table ID :0

    Priority :101
    Cookie :3
    Idle Timeout :0
    Hard Timeout :0
    Actions :['OUTPUT:CONTROLLER']
    Match :'dl_type': 2048, 'nw_src': '10.0.0.1', 'nw_proto': 1
    Byte Count :1176
    Duration Sec. :436
    Duration nSec. :695000000
    Packet Count :12
    Table ID :0

    Priority :100
    Cookie :2
    Idle Timeout :0
    Hard Timeout :0

Actions :['OUTPUT:NORMAL']
Match :'dl_type': 2048, 'nw_proto': 1
Byte Count :588
Duration Sec. :3629
Duration nSec. :630000000
Packet Count :6
Table ID :0

Priority :0
Cookie :0
Idle Timeout :0
Hard Timeout :0
Actions :['OUTPUT:CONTROLLER']
Match :
Byte Count :1482
Duration Sec. :3794
Duration nSec. :521000000
Packet Count :18
Table ID :0

- Intent to get the flows of switch s1 with rule id's.

Please enter new intent or enter exit
Show the flows of switch s1 with rule id
RESPONSE CODE 200
'rule_id': 3, 'priority': 101, 'dl_type': 'IPv4', 'nw_src': '10.0.0.1', 'nw_proto': 'ICMP', 'actions': 'DENY'
'rule_id': 2, 'priority': 100, 'dl_type': 'IPv4', 'nw_proto': 'ICMP', 'actions': 'ALLOW'

- Intent to enable host h1 to communicate.

Please enter new intent or enter exit
Allow host h1 to communicate
Contradiction detected
Do you want to proceed? Yes/No:
Yes
RESPONSE CODE 200

Mininet result after installation of above flow rule in switches
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7
h2 -> h1 h3 h4 h5 h6 h7
h3 -> h1 h2 h4 h5 h6 h7
h4 -> h1 h2 h3 h5 h6 h7
h5 -> h1 h2 h3 h4 h6 h7
h6 -> h1 h2 h3 h4 h5 h7
h7 -> h1 h2 h3 h4 h5 h6
*** Results: 0% dropped (42/42 received)

- Intent to block host h2 and host h3.

  Please enter new intent or enter exit
  Disable host h2 and host h3 to communicate
  RESPONSE CODE 200

  Mininet result after installation of above flow rule in switches
  mininet> pingall
  *** Ping: testing ping reachability
  h1 -> h2 h3 h4 h5 h6 h7
  h2 -> h1 X h4 h5 h6 h7
  h3 -> h1 X h4 h5 h6 h7
  h4 -> h1 h2 h3 h5 h6 h7
  h5 -> h1 h2 h3 h4 h6 h7
  h6 -> h1 h2 h3 h4 h5 h7
  h7 -> h1 h2 h3 h4 h5 h6
  *** Results: 4% dropped (40/42 received)

- Intent allow host h2 and host h3 to communicate.

  Please enter new intent or enter exit
  Activate host h2 and host h3 to communicate
  Contradiction detected
  Do you want to proceed? Yes/No:
  Yes
  RESPONSE CODE 200

  Mininet result after installation of above flow rule in switches
  mininet> pingall
  *** Ping: testing ping reachability
  h1 -> h2 h3 h4 h5 h6 h7
  h2 -> h1 h3 h4 h5 h6 h7
  h3 -> h1 h2 h4 h5 h6 h7
  h4 -> h1 h2 h3 h5 h6 h7
  h5 -> h1 h2 h3 h4 h6 h7
  h6 -> h1 h2 h3 h4 h5 h7
  h7 -> h1 h2 h3 h4 h5 h6
  *** Results: 0% dropped (42/42 received)

- Intent to disable switch s5 to forward packets.

  Please enter new intent or enter exit
  Block switch s5
  RESPONSE CODE 200

  Mininet result after passing above intent
  mininet> pingall
  *** Ping: testing ping reachability
  h1 -> h2 h3 h4 X X X

h2 -> h1 h3 h4 X X X
h3 -> h1 h2 h4 X X X
h4 -> h1 h2 h3 X X X
h5 -> X X X X X X
h6 -> X X X X X h7
h7 -> X X X X X h6
*** Results: 66% dropped (14/42 received)

- Intent to activate switch s5.
  Please enter new intent or enter exit
  Enable switch s5
  Contradiction detected
  Do you want to proceed? Yes/No:
  Yes
  RESPONSE CODE 200

  Mininet result after passing above intent
  mininet> pingall
  *** Ping: testing ping reachability
  h1 -> h2 h3 h4 h5 h6 h7
  h2 -> h1 h3 h4 h5 h6 h7
  h3 -> h1 h2 h4 h5 h6 h7
  h4 -> h1 h2 h3 h5 h6 h7
  h5 -> h1 h2 h3 h4 h6 h7
  h6 -> h1 h2 h3 h4 h5 h7
  h7 -> h1 h2 h3 h4 h5 h6
  *** Results: 0% dropped (42/42 received)

- Intent to get the flows of switch s1 with rule id's.

  Please enter new intent or enter exit
  Get flows of switch s1 with rule id
  RESPONSE CODE 200
  'rule_id': 5, 'priority': 204, 'dl_type': 'IPv4', 'nw_src': '10.0.0.2', 'nw_dst': '10.0.0.3', 'nw_proto': 'ICMP', 'actions': 'DENY'
  'rule_id': 6, 'priority': 205, 'dl_type': 'IPv4', 'nw_src': '10.0.0.2', 'nw_dst': '10.0.0.3', 'nw_proto': 'ICMP', 'actions': 'ALLOW'
  'rule_id': 7, 'priority': 205, 'dl_type': 'IPv4', 'nw_src': '10.0.0.3', 'nw_dst': '10.0.0.2', 'nw_proto': 'ICMP', 'actions': 'ALLOW'
  'rule_id': 2, 'priority': 202, 'dl_type': 'IPv4', 'nw_src': '10.0.0.1', 'nw_proto': 'ICMP', 'actions': 'DENY'
  'rule_id': 3, 'priority': 203, 'dl_type': 'IPv4', 'nw_src': '10.0.0.1', 'nw_proto': 'ICMP', 'actions': 'ALLOW'
  'rule_id': 4, 'priority': 203, 'dl_type': 'IPv4', 'nw_dst': '10.0.0.1', 'nw_proto': 'ICMP', 'actions': 'ALLOW'
  'rule_id': 1, 'priority': 201, 'dl_type': 'IPv4', 'nw_proto': 'ICMP', 'actions': 'ALLOW'

- Intent to delete a flow from a switch with rule id.

  Please enter new intent or enter exit
  Delete a flow from switch
  NOTE: Please do not delete any flow entries which are currently in use by RYU controller

RULE ID: 2
SWITCH ID: 1
RESPONSE CODE 200

- Intent to view the flows of switch s1 with rule id after deletion.

Please enter new intent or enter exit
Show the flows of switch s1 woth rule id
RESPONSE CODE 200
’rule_id’: 5, ’priority’: 204, ’dl_type’: ’IPv4’, ’nw_src’: ’10.0.0.2’, ’nw_dst’: ’10.0.0.3’, ’nw_proto’: ’ICMP’, ’actions’: ’DENY’
’rule_id’: 6, ’priority’: 205, ’dl_type’: ’IPv4’, ’nw_src’: ’10.0.0.2’, ’nw_dst’: ’10.0.0.3’, ’nw_proto’: ’ICMP’, ’actions’: ’ALLOW’
’rule_id’: 7, ’priority’: 205, ’dl_type’: ’IPv4’, ’nw_src’: ’10.0.0.3’, ’nw_dst’: ’10.0.0.2’, ’nw_proto’: ’ICMP’, ’actions’: ’ALLOW’
’rule_id’: 3, ’priority’: 203, ’dl_type’: ’IPv4’, ’nw_src’: ’10.0.0.1’, ’nw_proto’: ’ICMP’, ’actions’: ’ALLOW’
’rule_id’: 4, ’priority’: 203, ’dl_type’: ’IPv4’, ’nw_dst’: ’10.0.0.1’, ’nw_proto’: ’ICMP’, ’actions’: ’ALLOW’
’rule_id’: 1, ’priority’: 201, ’dl_type’: ’IPv4’, ’nw_proto’: ’ICMP’, ’actions’: ’ALLOW’

- Intent to get the overall status of switches.

Please enter new intent or enter exit
Get the overall status of all the switches

Switch ID :7
Status :enable

Switch ID :4
Status :enable

Switch ID :1
Status :enable

Switch ID :5
Status :enable

Switch ID :6
Status :enable

Switch ID :3
Status :enable

Switch ID :2
Status :enable

- Intent to get the port status of switch s1.

Please enter new intent or enter exit
Get the port status of switch s1

Port number: 1
Received Packets: 155
Transmitted Packets: 188
Received Bytes: 11026
Transmitted Bytes: 14042
Packets dropped(receiver side): 0
Packets dropped(transmitter side): 0
Received packets errors: 0
Transmitted packets errors: 0
Collisions: 0
Duration(sec): 2329

Port number: 2
Received Packets: 187
Transmitted Packets: 173
Received Bytes: 13952
Transmitted Bytes: 13532
Packets dropped(receiver side): 0
Packets dropped(transmitter side): 0
Received packets errors: 0
Transmitted packets errors: 0
Collisions: 0
Duration(sec): 2329

- Intent to get the aggregate flows of switch s1.

Please enter new intent or enter exit
Get the aggregate flows of switch s1
Packet Count : 308
Byte Count : 21222
Flow Count : 8

5 | FUTURE WORK

In the future the researches based on this field should be accomplishing more network related task and providing the network operators with more functionalities. In the proposed model all those intents are being processed which contains some specific set of keywords regardless of any grammatical mistakes but with the help of NLP and other techniques we can process only those intents which are grammatically correct. In the contradiction detection part focus must be laid to decrease the dataset size with increase in accuracy and number of hosts which is currently limited to ten. All the task mentioned above are a big challenge to be accomplished in the near future.

6 | CONCLUSION

The work that we have proposed is an interface between user and network which could be completely operated by intents in natural languages. Currently with the help of manually defined keywords we are able to process more than forty (40) intents and

generate response from RYU controller. For accomplishing the task for contradiction detection we have composed a dataset and then trained and tested four different machine learning models i.e. Logistic Regression, Naïve Bayes, Random Forest Classifier and Decision Tree Classifier. Upon observation of results with certain parameters such as accuracy, precision, F1 score, recall score and ROC curve we found out that Decision Tree Classifier gives best results as compared with other machine learning models. To the best of our knowledge this is the first work that provides such an interface by using NLP features and RYU controller additionally it also provides contradiction detection on hosts and switches to detect previously deployed intents.

ACKNOWLEDGMENTS

I would like to take this opportunity to express my deep sense of gratitude to Dr. Shashwati Banerjea who provided me help from time to time which helped me in doing lots of research and I came to know about many new things in the field of computer networks. Secondly I would also like to thank my parents, friends and rest all other who directly or indirectly helped me in completion of my work.

Financial disclosure

None.

Conflict of interest

The authors declare no potential conflict of interests.

References

1. Feamster N, Rexford J, Zegura E. The Road to SDN: An Intellectual History of Programmable Networks. *ACM SIGCOMM Computer Communication* 2014; 44(2).
2. Pang L, Yang C, Chen D, Song Y, Guizani M. A Survey on Intent-Driven Networks.. 2020.
3. Kiran M, Pouyoul E, Mercian A, Tierney B, Guok C, Monga I. Enabling intent to configure scientific networks for high performance demands.. 2017.
4. RYU controller documentation available: https://ryu.readthedocs.io/en/latest/index.html.
5. Kreutz D, M. V. Ramos F, Esteves P, Esteve Rothenberg C, Azodolmolky S, Uhlig S. Software-Defined Networking: A Comprehensive Survey. *In Proceedings of the IEEE* 2015; 103(1).
6. Alsudais A, Keller E. Hey Network, Can You Understand Me?. *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS): SWFAN 17* 2017.
7. S. Jacobs A, J. Pitscher R, H. Ribeiro R, A. Ferreira R, Z. Granville L, G. Rao S. Deploying Natural Language Intents with Lumi. *ACM* 2019.
8. Kaur S, Singh J, Singh Ghumman N. Network Programmability Using POX Controller. 2014.
9. Dr. Ing. Georg Carle P. Implementation and Performance Analysis of Firewall on OpenvSwitch.. 2015.
10. Esposito F, Wang J, Contoliy C, Davoliy G, Cerroniy W, Callegatiy F. A Behavior-Driven Approach to Intent Specification for Software Defined Infrastructure Management. *In IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)* 2018.
11. POX documentation available https://noxrepo.github.io/pox-doc/html/.
12. Mininet - An Instant Virtual Network on your Laptop or other PC. [Online] Available: http://mininet.org/.

13. Soule R, Basu S, Jalili Marandi P, et al. Merlin: A Language for Managing Network Resources. *IEEE/ACM TRANSACTIONS ON NETWORKING* 2018; 26(5).

14. Intent: Don't Tell Me What to Do! Accessed available at https://www.sdxcentral.com /articles/contributed. 2015.

15. Schulz D. Intent-based automation networks: Toward a common reference model for the self-orchestration of industrial intranets. 2016: 4657– 4664.

16. Saha BT, Tandur D, Haab L, Podleski L. Intent-based networks: An industrial perspective. 2018: 35–40.

17. Heorhiadi V, Chandrasekaran S, Reiter MK, Sekar V. Intent Driven composition of resource-management SDN applications. 2018: 86–97.

18. Zhang H, Wang Y, Qi X, Xu W, Peng T, Liu S. Demo abstract: An intent solver for enabling intent-based SDN. 2017: 968–969.

19. Gartner . Innovation Insight: Intent-Based Networking Systems available: https://www.gartner.com/doc/3599617/ innovation-insight-intent based-networkingsystems.. 2017.

20. Cisco Intent-Based Networking available: https://www.cisco.com/c/en/us/solutions/intent-based-networking.html. 2018.

21. Huawei : IDN Maximize Your Business Value available on https://developer.huawei.com/ict/en/site-idn. 2018.

22. Bureau C. Network operators to spend over 1.8 billion dollars in 2016. 2016.

AUTHOR BIOGRAPHY

Kumar Shubham

He is currently pursing his M.Tech in Information Security from MNNIT Allahabad under the department of Computer Science and Engineering (CSED).He has completed his B.Tech in Electronics and Communication (ECE).His areas of interest are Computer Networking, Software Defined Networking, Intent Driven Network, Natural Language Processing, Data Structures and Algorithms.

Dr. Shashwati Banerjea

She is Assistant Professor in the department of Computer Science and Engineering at Motilal Nehru National Institute of Technology Allahabad (MNNIT Allahabad). Her area of interests are peer to peer computing, distributed systems, database management systems.