User-driven Path Control through Intent-Based Networking

1st Anne-Ruth Meijer
Multiscale Networked Systems group
University of Amsterdam
Amsterdam, The Netherlands
anneruth02@gmail.com

2nd Leonardo Boldrini
Multiscale Networked Systems group
University of Amsterdam
Amsterdam, The Netherlands
l.boldrini@uva.nl

3rd Ralph Koning
SIDN Labs
SIDN
Arnhem, The Netherlands
ralph.koning@sidn.nl

4th Paola Grosso
Multiscale Networked Systems group
University of Amsterdam
Amsterdam, The Netherlands
p.grosso@uva.nl

***Abstract*—The UPIN (User-driven Path verification and control in Inter-domain Networks) project aims to implement a mechanism for a user to control the way data are traversing the network. Here we investigates the possibilities and limitations of Intent-Based Networking (IBN) for user-driven path control. Exploring several intent translation techniques allows us to define four main factors that influence the design of an Intent-Based Networking approach. The level of control, level of required knowledge, type of language, and network all influence the design. Based on the UPIN project demo, we design two approaches: technical-centric, which focuses on enabling Intent-Based Networking, and human-centric, which focuses on relieving restrictions from the expression methods for the user. Rasa is used to create a chatbot interface for the human-centric approach. We experiment with several configurations to discover the optimal pipeline for our training and testing data. Results indicate an 85 percent accuracy of intent recognition and 93 percent accuracy of entity extraction. Although the accuracy is not high enough to allow the Intent-Based Networking implementation to make decisions without supervision, the implementation proves to be a viable method of expressing intents for user-driven path control.**

I. Introduction

Governments and citizens depend on digital technologies that are deeply entangled in the core structures of society [1]. These technologies are built on the traditional Internet architecture and inherit some of its limitations, such as the lack user control of the network and a consequent erosion of trust [2]. The Responsible Internet paradigm wants overcome these problems by improving the Internet transparency, accountability and controllability [2]. The UPIN (User-driven Path verification and control in Inter-domain Networks) project, based on the notion of Responsible Internet, develops a framework for users to control the behaviour of the network [3] while integrating with the current Internet architecture. However, providing users with a degree of control requires to support efficient interactions between users and the network.

A method of translating desires into network level configurations is Intent-Based Networking (IBN) [4]. IBN allows users to control the network behaviour without requiring the knowledge of a network operator, even in increasingly complex networks. Managing, for example, 5G networks could be simplified and automated by IBN [5]. Likewise, managing Virtual Networks (VNs) can be complex. Using Network Virtualization, VNs can be created on top of the physical network infrastructure to allow multiple users to use the network independently [6]. IBN can be used to trace all the different intents in the network and to verify whether a new intent does not conflict with existing ones.

Our research investigates the possibilities and limitations of an IBN approach to provide the user-network communication required for user-driven path control. For example, this would benefit the setup of Wide Area Network paths for large scale scientific applications.

This paper first provides an overview of existing IBN translation techniques and presents two possible designs based on those techniques: a technical-centric and human-centric method. The technical-centric method enables the use of IBN in a UPIN project demo, while the human-centric method expands on this to provide the user with increased freedom of expressing intents. The human-centric method parses natural language using Natural Language Processing and here we investigate the performance of the adopted NLP techniques.

The rest of the paper is structured as follows. First, the concepts of Responsible Internet and the UPIN project are explained in section II to grasp the range of this research. Existing research on Intent-Based Networking is reviewed in section III. In section IV the design considerations for the implemented approaches are clarified. The implementation of the approaches is further detailed in section V. Last, the experiments and results are provided in section VI before discussing and making conclusions in sections VII and VIII.

II. Responsible Internet & UPIN

The Internet has evolved to a global infrastructure supporting a wide range of services and products on which all companies and governments depend upon [2]. This dependence is often based on systems manufactured and operated by entities in other countries, which generates in some circumstances political and economical concerns. As a consequence we are witnessing is the emergence of concerns regarding digital sovereignty and greater attention over control of data.

The Responsible Internet has been proposed to address these problems and provide a higher degree of trust and sovereignty for users of the Internet [2]. The main goal is to improve the transparency, accountability and controllability of the Internet.

Transparency provides the ability to describe the structure and properties of a network. Two types of transparency are identified: data and infrastructure transparency. Data transparency provides knowledge on how data is processed, including what devices it traversed and the type of devices. Infrastructure transparency provides insights into the properties of the infrastructure independent of the data flows.

Accountability can also be distinguished into a data and infrastructure type. Data accountability ensures that network operators explain the method of processing data along with routing decisions. Infrastructure accountability guarantees that the network operator explains the design of the infrastructure, for example an explanation on types of software.

Controllability provides the user with the ability to specify their expectations of the network. Based on descriptions of the network and data processing, users can express their desire on how the data is handled by the network. Controllability can be established by either policy making or creating alternative network functions, for example the ability to control paths based on certain properties.

A. The UPIN project

The UPIN project provides the concrete implementation for the Responsible Internet, namely transparency, controllability and accountability for the users of the network infrastructure [7][3].

UPIN introduces an Internet framework allowing the user to define network behaviour. The framework utilizes existing network paradigms, for example Software-Defined Networks, to blend with the current Internet infrastructure [7]. The UPIN domain is designed to work seamlessly alongside non-UPIN enabled domains to allow a UPIN domain to be integrated in the existing infrastructure.

The UPIN framework consists of a Domain Explorer, Path Controller, Path Tracer, Path Verifier, and Front-end [3]. The Domain Explorer obtains the metadata about properties of the network, including security and environmental details. It stores detailed knowledge on the nodes in the network. The Path Controller is in charge of setting the forwarding rules based on the desires of the user. The Controller is only able to influence the nodes in its own domain. The Path Tracer gathers measurements on the traffic in the UPIN domain. The goal is to store important details for the possible verification. The Path Verifier examines whether the desires of the user are satisfied. Using tracers and the original intent of the user the path can be verified. However, if the path traverses a non-UPIN enabled domain the Path Verifier cannot be certain whether the intent is satisfied over the full path. The Front-end provides a method of communication between the user and the domain. The work presented here relates closely to this last component in the framework.

B. The UPIN demo

The UPIN demo is an interactive environment to demonstrate the capabilities provided by a UPIN enabled network.

The user interacts with the network environment through a browser-based user interface. The user is presented with a topology, either on a map or in a schematic graph. After logging in, the user can view which nodes data can traverse in the network. The user can change the properties of the path to be taken and the graph of traversed nodes changes accordingly. The user interface communicates with the underlying network via its back-end and through MQTT. MQTT is a publish and subscribe protocol [8] that is used as an interface between the demo components:

- domain agents to control each node and publish performance information;
- an aggregator component to collect the information from the nodes and compile a topology view for the user-interface;
- the IBN component in we describe in Sec. V.

The front-end of the demo is implemented in JavaScript using Snowpack [9], the back-end is implemented in Node.js and the rest of the components using Python.

III. Intent-Based Networking

Intent-Based Networking (IBN) provides users with the ability to express a desire and translate said desire into network level configurations, for example policies. IBN is based on the concepts of intent and policies and applied in Software-Defined Networks (SDN).

In 2017, the Internet Research Task Force (IRTF) has published a draft document to clarify the concepts behind IBN [4]. At the moment of writing, this document is a work in progress. The need for this document originates from the lack of a common understanding of the terms and concepts that surround IBN. Earlier definitions of intent describe the intent as a type of policy. To avoid intent and policy becoming synonyms the working document provides definitions for both terms [4]. The definition of policy is presented as:

> *"A set of rules that governs the choices in behavior of a system"* [4]

The definition of intent is presented as:

> *"A set of operational goals (that a network should meet) and outcomes (that a network is supposed to deliver), defined in a declarative manner without specifying how to achieve or implement them."* [4]

The difference between the two terms is that intent specifies a set of goals and outcomes, while a policy entails a set of rules.

Meaning, an intent is what a user expects and desires of the network, without stating how the network should achieve those expectations and desires. A policy is exactly a description of how the network should behave. Thus, an intent can be achieved by implementing policies.

In IBN the intent is provided about the network, in particular a Software-Defined Network. An SDN is a network approach that allows network operators to configure the network with programming through open interfaces [10]. In an SDN the control plane is separated from the data plane. This allows the control of the network to be centralized and be implemented in software [11][10]. SDNs are divided into three primary layers and two interfaces. The three primary layers are the control layer, application layer, and the infrastructure layer. The two interfaces are Northbound and Southbound APIs. In IBN the application layer sets the requirements of the network, but without knowledge of how the requirements are implemented. Through an alternative Northbound interface, the intent-based Northbound interface, the intent of the application is translated into a control layer configuration [11]. For the application layer it is no longer required to understand the components of the network. The intent is provided by the application layer and the intent-based Northbound interface translates this into policies understood by the network.

A. Intent-Based Networking

Intent-Based Networking is based on the notion of intent and allows users to express intents without hardware specific rules. To achieve this, IBN requires a method of translating the intent along with an activation and validation phase. The concept of IBN is visualized in Figure 1. First, a user expresses the intent. This is translated into rules comprehensible by the network, in other words policies. These policies are activated and configured in the infrastructure. The network infrastructure provides feedback to a validation phase. The activation and validation of the intents are continuously verifying whether the intents can be satisfied based on network-driven feedback. The status of the intents is fed back to the user that expressed the intent. Using this method of validation, the user intent has to be expressed once. After the expression the intent will be satisfied as long as the infrastructure supports it and the intent is not changed.

B. Limited Translation Techniques

The technique to translate an intent into network configurations is based on the language from which it needs to be translated. Most translation techniques provide a specific language set that can be translated. However, some techniques translate from plain English.

We classify all techniques that accept a limited syntax as a limited translation technique. These techniques include ONOS intent framework [6], [12], [13], the NEMO language [13], [14], the NIC project [13], [15], and Nile language[16]–[18].

1) ONOS: Open Network Operating System (ONOS) has build a framework for IBN[12]. It provides a system that translates a previously specified intent into operations on the network environment. The intent in the ONOS framework is an object that describes the request of an application to alter the behavior of the network. The intent can include descriptions of network resources, constraints, criteria, and instructions. The intent that users can express include intents already compiled in the framework or intents added to the compiler[19].

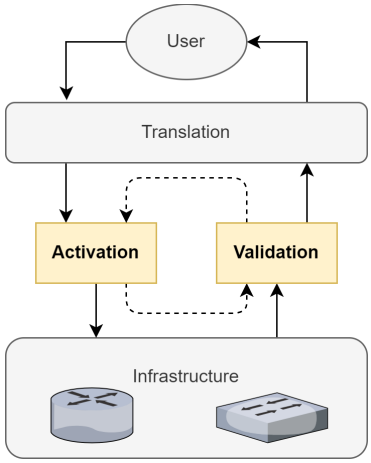

Fig. 1. Visualization of the concept of Intent-Based Networking

2) NEMO: The NEtwork MOdeling (NEMO) language is a part of the OpenDayLight framework [14]. This framework is focused on implementing an intent Northbound interface. NEMO provides each user with a *Virtual Network (VN) space*, a private space that is managed and operated by one user. The intent the users can express is limited by the syntax of NEMO [14]. Using this syntax users can describe their demands about entities of the network, including nodes, connections, and flows. A drawback of NEMO is the lack of adaptability of the network configurations in response to altering network conditions [20].

3) NIC: The Network Intent Composition (NIC) project is created by OpenDayLight [15]. The project uses an intent Northbound interface alongside OpenDayLight Network Service Functions and Southbound Plugins to control the physical devices. The intent that can be provided to the Northbound interface is limited to previously configured intents. Users can add, remove, list, show, and compile intents in the configuration of the Northbound interface.

4) Nile: The Network Intent LanguagE (Nile) acts as an abstraction layer between natural language and a programming language [17]. The gap between natural language and network configuration is large, so this language is proposed to bridge this gap. Nile is structured enough to be easily translated to different networks, yet stays close to natural language. An example of a Nile intent is visualized in Figure 2.

C. Natural Language Translation Techniques

Techniques that translate from natural language, for example English, are less limited. These techniques include iNDIRA [13], [21], Lumi [18] and BERT [13], [22].

```
define intent qosIntent:
  from   endpoint('gateway')
  to     endpoint('database')
  for    client('B')
  add    middlebox('firewall'), middlebox('ids')
  with   latency('less', '10s'),
         throughput('more or equal', '100mbps')
  allow  traffic('https')
  start  hour('09:00')
  end    hour('18:00')
```

Fig. 2. Example of a Nile intent [17]

1) iNDIRA: Intelligent Network Deployment Intent Renderer Application (iNDIRA) systems implement another layer between a user and the Northbound interface to translate natural language from the user into network commands [21]. The technique to translate natural language is *ontology engineering*. Ontology engineering converts the query provided by the user into a graph representation. The specific method used to construct the intent semantic web is a Resource Description Framework (RDF) graph.

2) Lumi: Lumi translates user expressed intents using Natural Language Processing (NLP) into Nile [18]. Lumi uses Named Entity Recognition (NER) to extract entities and label them. The entities are extracted from queries gathered in a chatbot interface. The NER method classifies the entities in the sentence based on experience gathered in the learning phase. There is not enough data to classify the entities with enough accuracy, so it uses user feedback to correct the classifications.

3) BERT: A survey from 2020 on the advances in Intent-Based Networking states that advances in Natural Language Understanding (NLU) can aid in processing queries expressed in a natural language [13]. The language representation model Bidirectional Encoder Representations from Transformers (BERT) proves to perform well in intent classification and entity extraction tasks. The benefit of BERT is that is does not require labeled data to be trained on. BERT uses unlabeled data to pre-train which improves the understanding of the complete language [22].

IV. Design considerations

We have identified four main factors to consider when developing our IBN based implementation: the level of control the user is allowed to have, the knowledge required, the language used by the user, and the network onto which to implement the intent.

A. Control

One of the important considerations for IBN is what the user is allowed to express. This depends on the level of control a user is allowed to have over the network. A limited translation technique could be sufficient if a user has a limited level of control and could only, for example, express intents about one feature. The level of control influences the number and type of intents. The level of control network operators are willing to provide to the users depends on the type of user and the type of network. For a system to be controlled by a user, the user also requires knowledge of the network. Thus, the level of control depends on the transparency of the network [7]. We define two levels of control: unlimited and limited. Unlimited control provides the user control over every service and aspect of the network. The limited control indicates that the network does not allow users to control every functionality.

B. Knowledge

The level of required knowledge for using an IBN approach is also an important factor for the design. It entails the knowledge that should either be provided or the users are expected to have. Allowing users to control their data flows, requires the network to provide the users with information on the current flows, i.e. a form of transparency by the network operators. The level of knowledge depends on the level of control and the network. The controlling ability provides an indication of the subject on which the user is required to have knowledge. Excluding vendors, for example, only requires knowledge on available vendors and the ones currently used. The network transparency influences the amount of knowledge the user is expected to possess. A user, possessing a detailed knowledge on the topology, can request specific paths, while others without that knowledge cannot.

C. Language

The language set used to express intents influences the complexity of the translation technique for IBN. Using a limited language set simplifies parsing user input, but it limits the freedom of the user and requires the user to gain knowledge of the usable syntax. We define two types of language sets that can be used to implement IBN in the UPIN context: restricted and unrestricted. The restricted language utilizes a specified language set, where each intent can only be expressed by a few queries. The unrestricted set supports natural language allowing several descriptions for the same intent. Research indicates that Natural Language Processing (NLP) provides opportunities for intent recognition from natural language [13].

NLP aims to gather knowledge from human expressed language to make computer systems understand provided queries and use the understanding to perform tasks [23]. This research primarily aims at the understanding of natural language, known as Natural Language Understanding (NLU). An important task for NLU is understanding what the context of the sentence is [24].

Intent detection is a classification problem [24]. Sentence features are used to classify the intent of the sentence. The features include words and grammar, but can be extended with word context, sentence context or meta-data.

Ambiguity in interpretation is an issue for intent detection [24]. It is difficult to classify different classes if the samples have similar features. An approach to mitigate this issue is to include multiple training samples of similar cases in the different classes to allow the training phase to learn the proper distinctions.

Training data is an important aspect of classification problems, since the classification is based on what is learned in the training phase. A solution for lack of training data is utilizing models that do not require labeled data or generating training data. However, generating data can cause a model to under perform on real world data.

Another possible issue for intent detection can be imbalanced training data [24]. Some classes might be represented less by the training data causing imbalanced classification performance. This issue is solved by balancing the samples.

D. Network

For expressing intents additional knowledge about network characteristics is required. If the network is not able to provide this information, the intents are limited. Hence the level of openness of a network influences the design and achievable scope of an IBN solution.

E. Approach

We design the implementation around a UPIN network demo. The demo environment provides knowledge about the network by visualizing it, while supporting limited services based on path creation. We must note that our IBN based solution will work on UPIN enabled domains.

As mentioned before, we have designed two IBN approaches based on the four relevant factors. Table I shows the approaches and their accompanying factor levels. To investigate the possibilities for IBN for user-driven path control we focus on a technical centric and human centric approach. This results in the first approach being limited by the language the user can use and the second providing more freedom. Both methods require knowledge of the network topology and its functionalities. Information on the functionalities provides users with knowledge of what to control. The topology provides insight on data flows.

TABLE I
DESIGN FACTORS FOR THE IBN APPROACHES

	Control	**Required knowledge**	**Language**	**Network**
Technical centric approach	unlimited	Network and Syntax	Restricted	UPIN demo
Human centric approach	unlimited	Network	Unrestricted	UPIN demo

1) Intent design: The UPIN users are allowed to influence the data paths, namely define the source and destination, as well as defining inclusion criteria, exclusion criteria, and capacity criteria. The inclusion and exclusion criteria relate to attributes of the network devices and the path itself. A visualization of the attributes that the user can influence is shown in Figure 3. The attributes of network devices involve three categories; the devices, the vendor, and the country. Included devices are devices that must be included in the path and excluded devices are forbidden to traverse. For the vendor and country we also distinguished an excluded and included category. If an included vendor is set, the path can only include devices from that vendor. The same goes for included country. We have distinguished path attributes as the capacity and the hop limit. The capacity entails bandwidth and latency constraints.

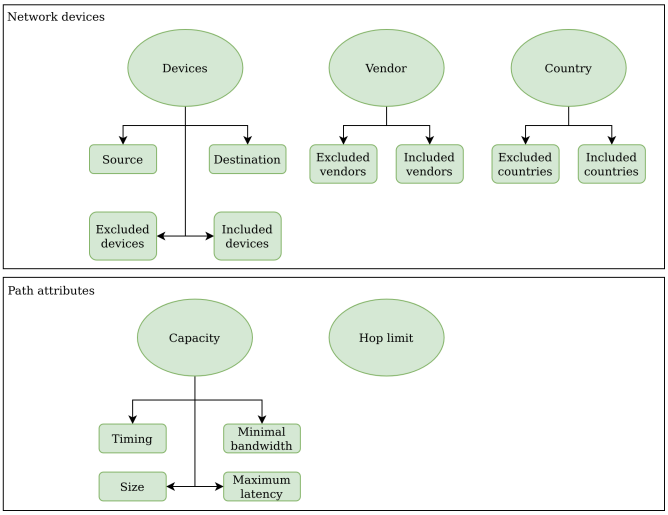

Fig. 3. Design of the path creation intent

2) Technical -centric approach: The goal for this approach is to allow users to express intents about the service provided by the UPIN demo, ie path creation. The translation technique for this approach is simple as we parse specific and well defined input, since here we do not provide freedom in user expressions.

3) Human centric approach: The human-centric approach uses Intent Detection to understand the desires of the users from their expressed query. The goal of this approach is to allow users to express their intents in natural language. The main challenge for developing an approach using Natural Language Understanding revolves around the training data. We integrate user feedback in our design and use a chatbot interface for communication, inspired by [18]. The chatbot advances the demo environment by translating plain English into configurations for the demo. Apart from specific demo configurations our design includes a more general translation enabling the approach to be implemented in other network environments.

V. IMPLEMENTATION

A. Technical centric implementation

In this implementation we provide users with limited syntax. This is listed in table II. Currently, the demo only supports the listed services. Still, the last two rows of the table show how we provide mechanisms for implementing more services in the demo syntax in the future. We have developed an IBN module and implemented this in the demo back-end. The module includes logic to parse syntax and find available paths. The demo front-end includes an editor style interface where the user can enter the specific commands to express

TABLE II
SYNTAX FOR THE IBN MODEL IN THE UPIN DEMO

Class	Service	Syntax Demo
Path creation	Create path between device1 device2	SELECTPATH 'Device1' 'Device2'
Exclusion on bandwidth	Create path with the minimum bandwidth set to X	WHERE 'bandwidth' = X
Exclusion on latency	Create path with maximum latency set to X	WHERE 'latency' = X
Inclusion of device(s)	Create path including device3	VIA 'device3'
Exclusion of device(s)	Create path not including device3	EXCLUDING 'device3'
Exclusion of vendor(s)	Create path not including vendor X devices	WHERE 'vendor' !IN [X]
Inclusion of vendor(s)	Create path only including vendor X devices	WHERE 'vendor' IN [X]
Limiting path length	Create path with a maximum of X devices	LIMIT X
Exclusion of country	Create path not including devices situated in country X	WHERE 'country' !IN [X]
Inclusion of country	Create path only including devices situated in country X	WHERE 'country' IN [X]
Exclusion of attribute X	Create path not including devices where attribute X is equal to Y	WHERE 'X' !IN [Y]
Inclusion of attribute X	Create path only including devices where attribute X is equal to Y	WHERE 'X' IN [Y]

the intent. These commands are parsed in the demo back-end using JavaScript. The uppercase terms in the syntax are used to detect the correct intent. Based on the intent, the relevant entities are extracted. Since we only support the defined syntax we do not include a validation process, and only the precise syntax can be parsed. A list is formed with the nodes that adhere to the intent expressed, based on the intent commands. Using these nodes all available paths are found using breadth first search.

B. Human centric implementation

The human-centric approach expands the technical-approach by offering an alternative method of communication to the demo environment. To implement the chatbot we use Rasa Open Source [25]. Rasa is a framework for automated conversations supporting message understanding, saving conversations and providing API endpoints. It provides a dialogue system based on machine learning to understand natural language. Since it is open source, extensible and well documented, Rasa proves to be more versatile than enterprise chatbot development software [26] [27].

For the implementation in the UPIN demo network we expand the front-end with a chat interface and connect the back-end to Rasa. The chatbot provides the user with the syntax which can be sent to the editor interface. In the editor interface the user is able to plan the path, similar to the technical centric approach. We make this code available via BitBucket[1] to support reproducability of our work as well as engage with other researchers interested in these efforts.

C. Configuration

To train the intent classification, entity extraction and decision making components of the chatbot, Rasa uses a configuration file. The configuration defines the pipeline and policies used to make predictions based on the user input. The pipeline can be divided into 4 main components: the tokenizer, featurizers, classifiers, and selector.

The tokenizer splits the query into tokens [25]. To split the sentence on word level we use the *WhitespaceTokenizer*. This includes the substitution of special characters into whitespaces under the condition of it being attached to a word and preceded or followed by a whitespace, begin-, or end of the sentence. This allows *"test! "* and *"test"* to be tokenized into the same word.

When the tokens are created the features of the words are extracted [25]. Our implementation utilizes three types of sparse featurizers. A sparse featurizer returns a featureset as vectors. The first featurizer occuring in the pipeline is the *RegexFeaturizer* which creates features for both intent detection and entity extraction. The second featurizer in the pipeline, *CountVectorsFeaturizer*, creates a bag-of-words representation of the messages [25]. The bag of words representation disregards word order and focuses on the amount of similar words in a sentence. The last featurizer, *LexicalSyntacticFeaturizer* is implemented to support entity extraction [25]. We apply this featurizer to create additional features for entity extraction since our main intent has the ability to contain several different entities.

The intent classifier assigns an intent to an input query. The Entity extractor extracts the entities from that query. The execution of these tasks is combined in the *DIETclassifier*. The Dual Intent Entity Transformer (DIET) extracts the entities, intent, and an intent ranking using the features created previously in the pipeline [25]. A benefit of using the *DIETclassifier* is the dual intent recognition and entity extraction ability and that it performs well for custom entities. Both these benefits are of use for our implementation.

The *ResponseSelector* predicts the appropriate response for the chatbot based on the user messages. The selector uses the features to rank the possible responses by confidence.

The configuration pipeline extracts the intent and entities from the user input [25]. Based on this input actions need to be taken at every step using the policies.

The human centric approach is developed to extend the UPIN demo, providing users with an alternative method of expressing intents. To achieve this the implementation translates user input into demo understandable syntax. However,

[1]https://bitbucket.org/thesis-chatbot/chatbot

TABLE III
TRAINING DATA AMOUNT FOR THE INTENTS

Intent	Number of examples
greet	13
goodbye	10
correct	18
deny	12
create_path	92
bot_challenge	4
reset	9
help	7
more	5
list	6

this is not a general adopted syntax to configure networks. To improve the generalizability of this extension we provide a second output method to Nile syntax. Nile syntax is used in earlier research and provides an abstraction layer that resembles plain English [16]–[18]. The syntax allows to be easily parsed into network configurations for different types of networks. We have implemented Nile syntax based on the previous research adapted to our intent.

VI. EXPERIMENTS & RESULTS

A. Experimental setup

1) Training data: Our implementation of the chatbot model saves previous conversations that can be appended to the training data. It is important for the NLU model to use real world data. Synthetic messages will not fully represent human conversations, which will cause the model to underperform [25]. If the testing phase uses the same synthetic data the model could appear as a well performing model, while this might not perform well on real world data. For our intents there are no existing datasets. Therefore, we have chosen to create the data ourselves.

Table III shows the amount of examples per intent. The *create_path* intent includes entities which need to be represented in the examples. The amount of examples per entity is presented in table IV. The entities *entity, value*, and *option* are not mentioned since these are used for forms and are not present in intents. We have set a minimum of five examples per intent and ten examples per entity. The entities have more examples because these contain more similarities and we aim to prevent ambiguity in interpretation.

By providing more examples we allow the chatbot to learn the proper distinctions. We try to prevent imbalanced classification performance by balancing the data between the entity roles. As a consequence, each role in the entities contain roughly the same amount of examples. The exceptions to this rule are the source and destination because these are required for every path.

2) Configuration pipelines: To test what configuration works best for our implementation we test several NLU configuration pipelines for Rasa. We divide the tests into two categories. First, we test the differences of adding more characteristics to base features on. Second, we test the benefits of additional methods to gather features. These experiments give us the best performing configuration for our data.

TABLE IV
TRAINING DATA AMOUNT FOR THE ENTITIES

Entity	Role	Number of examples
devices	*source*	92
devices	*destination*	92
devices	*excluded_device*	20
devices	*included_device*	20
vendor	*included_vendor*	10
vendor	*excluded_vendor*	10
country	*included_country*	10
country	*excluded_country*	10
limit		10
capacity	*timing*	10
capacity	*size*	10
capacity	*minimal_bandwidth*	10
capacity	*maximum_latency*	10

Table V presents the different features applied in the *LexicalSyntacticFeaturizer*. Each table entry shows the features for the previous token, the current token, and the next token. The options for placement are BOS (beginning of sentence) and EOS (end of sentence). We have three options for case sensitivity: low (lower case), upper (uppercase), and title (first character is uppercase). Partial tokens are extracted as features as well. The beginning characters are extracted using prefix2 or prefix5 to look at the first two or five characters. The ending characters are presented as the features: suffix1, suffix2, suffix3, suffix5. The case and placement features are included in every configuration, since we know that these are indicators for differences in our data. For example, vendors and countries are often written with a title. We test what the effects are of applying different character features. Each character feature is applied to the preceding token, the token, and the succeeding token. Comparing the benefits of each feature on the different tokens allows us to find the ideal features.

The best combination of features is tested by comparing the different configurations listed in table V.

3) Evaluation methods: The pipelines are compared through the performance of their accompanying NLU models. This performance is measured in accuracy, precision, recall, and F1 score. To calculate the performance metrics we use the following notions:

$$TP = \text{true positives}$$
$$FP = \text{false positives}$$
$$TN = \text{true negatives}$$
$$FN = \text{false negatives}$$

a) Accuracy: The accuracy measures how well the model has classified the intents and the entities [28]:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

b) Precision: The precision is an indicator of how many classified entries are correctly predicted [28]. It looks at the

TABLE V
CONFIGURATION PIPELINES WITH DIFFERENT FEATURES FOR THE *LexicalSyntacticFeaturizer*

Name	Preceding token	Token	Succeeding token
PF1	BOS, low, title, upper	BOS, EOS, low, title, upper	EOS
PF2	BOS, low, title, upper, **suffix1**	BOS, EOS, low, title, upper	EOS
PF3	BOS, low, title, upper, **suffix2**	BOS, EOS, low, title, upper	EOS
PF4	BOS, low, title, upper, **suffix3**	BOS, EOS, low, title, upper	EOS
PF5	BOS, low, title, upper, **suffix5**	BOS, EOS, low, title, upper	EOS
PF6	BOS, low, title, upper	BOS, EOS, low, title, upper, **suffix1**	EOS
PF7	BOS, low, title, upper	BOS, EOS, low, title, upper, **suffix2**	EOS
PF8	BOS, low, title, upper	BOS, EOS, low, title, upper, **suffix3**	EOS
PF9	BOS, low, title, upper	BOS, EOS, low, title, upper, **suffix5**	EOS
PF10	BOS, low, title, upper	BOS, EOS, low, title, upper	EOS, **suffix1**
PF11	BOS, low, title, upper	BOS, EOS, low, title, upper	EOS, **suffix2**
PF12	BOS, low, title, upper	BOS, EOS, low, title, upper	EOS, **suffix3**
PF13	BOS, low, title, upper	BOS, EOS, low, title, upper	EOS, **suffix5**
PF14	BOS, low, title, upper, **prefix2**	BOS, EOS, low, title, upper	EOS
PF15	BOS, low, title, upper, **prefix5**	BOS, EOS, low, title, upper	EOS
PF16	BOS, low, title, upper	BOS, EOS, low, title, upper, **prefix2**	EOS
PF17	BOS, low, title, upper	BOS, EOS, low, title, upper, **prefix5**	EOS
PF18	BOS, low, title, upper	BOS, EOS, low, title, upper	EOS, **prefix2**
PF19	BOS, low, title, upper	BOS, EOS, low, title, upper	EOS, **prefix5**

proportion of correctly predicted entries out of all predicted entries:

$$precision = \frac{TP}{TP + FP} \tag{2}$$

c) Recall: The recall is an indicator of the proportion of correctly classified entries[28]. The recall does not look at the incorrect labeled entries in a class, but focuses on the proportion of entries that are correctly predicted to the class:

$$recall = \frac{TP}{TP + FN} \tag{3}$$

d) F1 score: The F1 score is an alternative test for the accuracy of classification derived from the precision and recall [28].The F1 score combines the recall and precision into a measure that provides the harmonic mean for the two metrics. With unbalanced data the F1 score provides an alternative method of calculating the accuracy to make sure the performance is well measured.

$$F1 = \frac{2 * (precision * recall)}{precision + recall} \tag{4}$$

e) Average: The performance metrics are calculated as an average over the intent and entity classes. We have chosen to calculate this as micro averages, since we have unbalanced amount of examples per intent and entity classes. Taking the micro average means that we do not calculate the average as a harmonic mean between the classes, but we take the average over all data entries.

B. Pipeline comparison results

The performance of the different features on entity classification are listed in table VI. We focus on the entity classification performance, since the *LexicalSyntacticFeaturizer* creates features for entity extraction. The highlighted entries represent the features that perform best in one of the measurements. The table shows that the FP2, FP4, and FP15 perform best for the preceding token. FP2 and FP4 add the last character and the last three characters to the featureset. Adding the last character improves the precision and adding the last three characters improves the recall. FP15 includes the first five characters to the featureset. This performs best in all measurements. For the token itself FP7 and FP16 outperform the other features in all measurements. FP7 appends the last two characters and FP16 the first two characters as features. For the succeeding token the best performing features are FP12, FP13, and FP19. For this token the first three and five characters improve the performance as well as the last five characters.

The best performing features are implemented in the *LexicalSyntacticFeaturizer*. To compare the effect of different featurizers the performance of them is tested on the test data. Table VII presents the results of the four compared NLU pipelines. The highlighted table entries indicate the best performance on the different measurements. Overall P4 performs best with recording the highest performance in every category but one. For entity precision P3 performs better. However, P3 has worse results in the intent category compared to P4. Apart from comparing the measurements for the best performance we have also tested the intent recognition ability with different amounts of training data. Figure 4 shows the F1 scores of the different pipelines. The figure indicates that an increased amount of training data improves the F1 score on P2, P3, and P4. P1 plateaus around 60 percent, while the other pipelines increase from an F1 score between 64 and 69 percent to an F1 score between 80 and 85 percent.

Figure 5 depicts the performance of the P4 pipeline. The confusion matrix indicates which intents and entities where misclassified during the tests along with the class they got confused with. The following intents got confused for each other once: *deny* and *correct*, *greet* and *correct*, and *deny* and *more*. The entities that got misclassified consist mostly of included entities that got confused for excluded entities and vice versa. *Bandwidth* and *latency*, and *source* and *destination* have been confused as well. The confidence distribution histogram for intent classification depicted in Figure 6 provides an insight on the confidence the classifier had while predicting the intents. The confidence while making a wrong intent prediction ranges between 39 and 98 percent. For the entity predictions six

TABLE VI
AVERAGE ACCURACY, PRECISION, RECALL AND F1 SCORE OF THE COMPARISON OF THE FEATURES FOR ENTITY CLASSIFICATION OF THE *LexicalSyntacticFeaturizer* ROUNDED TO TWO DECIMAL PLACES

Name	Accuracy	Precision	Recall	F1 score
FP1	0.80	0.39	0.36	0.38
FP2	0.90	0.74	0.68	0.71
FP3	0.89	0.73	0.66	0.69
FP4	0.90	0.72	0.71	0.72
FP5	0.90	0.73	0.68	0.70
FP6	0.78	0.35	0.35	0.35
FP7	0.83	0.48	0.46	0.47
FP8	0.82	0.44	0.43	0.43
FP9	0.81	0.42	0.43	0.42
FP10	0.84	0.55	0.51	0.53
FP11	0.85	0.59	0.54	0.56
FP12	0.85	0.87	0.58	0.57
FP13	0.89	0.69	0.64	0.66
FP14	0.88	0.69	0.65	0.67
FP15	0.90	0.75	0.69	0.72
FP16	0.82	0.45	0.45	0.45
FP17	0.81	0.43	0.43	0.43
FP18	0.88	0.43	0.49	0.44
FP19	0.88	0.49	0.63	0.64

TABLE VII
AVERAGE ACCURACY, PRECISION, RECALL AND F1 SCORE FOR ENTITY AND INTENT CLASSIFICATION TO COMPARE THE PERFORMANCE OF ADDITIONAL FEATURIZERS ROUNDED TO TWO DECIMAL POINTS

	P1	P2	P3	P4
Accuracy intent	0.63	0.84	0.83	0.85
Precision intent	0.63	0.86	0.83	0.86
Recall intent	0.63	0.84	0.83	0.85
F1 score intent	0.63	0.83	0.83	0.85
Accuracy entity	0.74	0.82	0.93	0.93
Precision entity	0.33	0.46	0.85	0.83
Recall entity	0.29	0.46	0.80	0.80
F1 score entity	0.31	0.46	0.82	0.82

wrong prediction are made with a confidence over 70 percent, and six with a confidence less than 70 percent. For both intent and entity classification the confidence for the correct predictions succeeds 90 percent for most cases.

VII. DISCUSSION

The first experiment results indicate that adding features based on characters in the token and its surrounding tokens improve the performance of the chatbot. The structure of our data causes the classifier to benefit from character and placement information for decision making. Often the word “from” appears in front of the source for example. Experimenting with the different features provides an insight into what token characteristics are important for classification. For our data we see that adding features based on small character sets of even one character can improve the accuracy from 80 percent to 90 percent and the F1 score from 38 percent to 71 percent. Another method to structure the data is to divide the services into different intents.

The results of the second experiment indicate that the best performance is achieved by configurations with the character

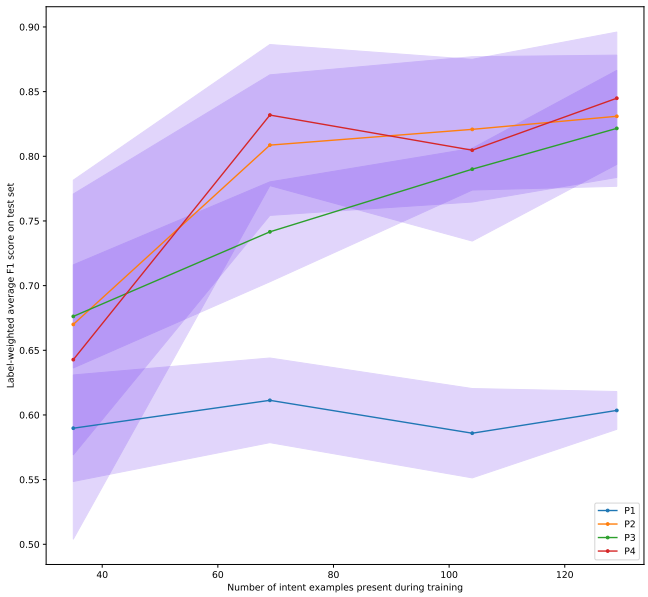

Fig. 4. F1 score of different pipelines with different amounts of intent training examples

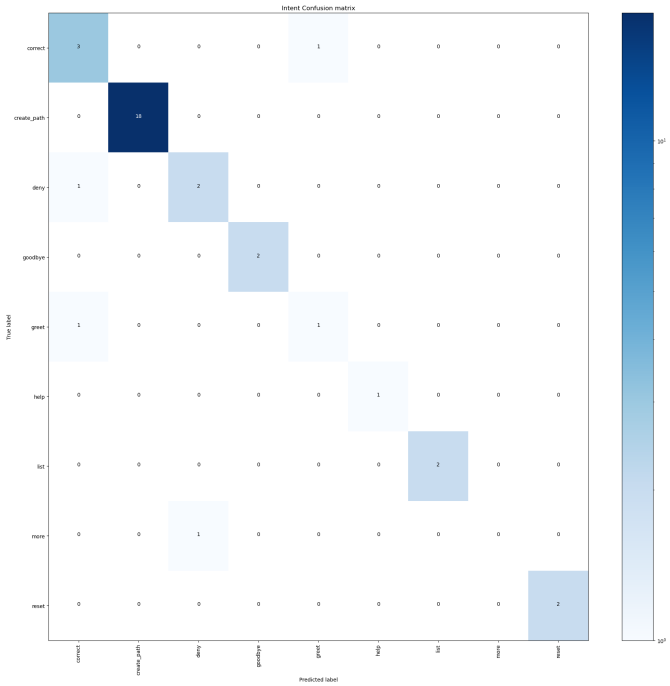

Fig. 5. Confusion matrix of the intent prediction for the test data

and placement features included. The configuration without the bag-of-word features does perform worse on intent recognition and the configuration without the placement and character features performs worse on entity extraction. A combination of the two overall performs best. The performance on entity classification shows that our extensive use of entities in one intent

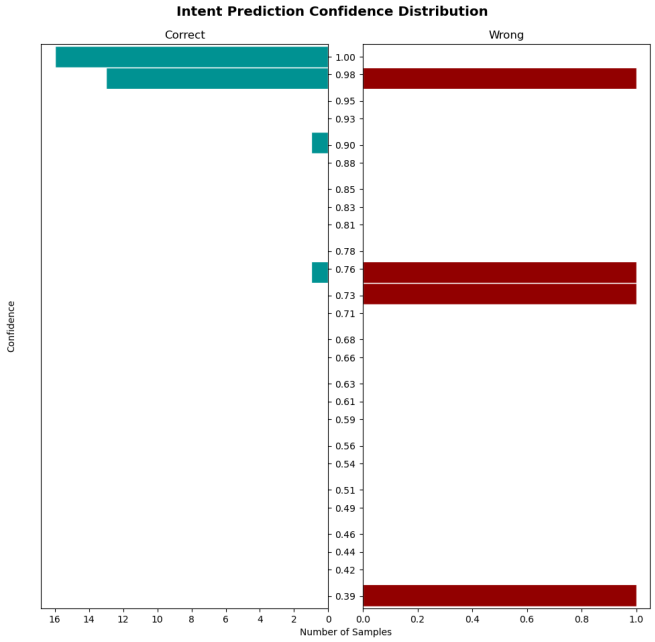

Fig. 6. Confidence distribution of the intent prediction for the test data

requires the *LexicalSyntacticFeaturizer*. The best performing pipelines improve when the amount of training data increases, showing no signs of plateauing. For this reason we cannot determine an optimal number of training examples. This can only be determined by adding data to the training and test sets, preferably real world representative data. Otherwise, the model doesn't perform optimally in a real world deployment.

The confusion matrix on the best performing model indicates some confusion mostly on entities that could be filled with the same values. An example that provides confusion is the sentence *"From A to B including Cisco devices"*. Changing two characters in this sentence changes the word *"including"* to *"excluding"*, which changes the entity. Learning these precise differences is difficult with limited data.

An unexpected result is the confidence when a wrong classification is made. This could be due to ambiguity in the training data, since some the examples are very similar. Restructuring the data or providing more examples could be the solutions to this problem.

Overall the performance of the model is high, but not high enough to allow the chatbot to classify intents and entities without supervision and a method of rectifying for the user. Improving the model with more training data could eliminate this need. The model should also be trained with real world data. Since it is trained and tested on data created by us, the results could potentially be biased on this type of data. Providing the model with real world training and testing data provides a transparent insight into its performance.

VIII. Conclusion

The aim of this research is to investigate the limitations and possibilities of Intent-Based Networking for user-driven path control. We have implemented Intent-Based Networking in the UPIN demo. This work, however, can be extendable beyond UPIN and the following considerations do not apply only within the UPIN demo. The limitations we identified are mostly based on the network possibilities and the language used for expression. The level of knowledge of the network limits the ability of the user. The language used limits the method of expression. Not limiting the user in method of expression is only possible when Natural Language Understanding is used. This technique provides possibilities for freedom in expression. However, the performance will depend on quality of the data available and the performance of classifiers. The results show that for user-driven path control an implementation using NLU is possible. However, it does not perform perfectly so user feedback is required.

Improving the performance of the chatbot to enable the ability to individually classify entities and intent could be interesting future research. Our advice is to use saved conversations from the current chatbot implementation, including data with, for example, common spelling mistakes. Future research could also include security risks of implementing a human centric IBN method. Knowing how a user could maliciously influence the system could provide alternative insights in the design considerations. Providing users with network transparency could also impact the security.

Acknowledgements

This research received funding from the Dutch Research Council (NWO) under the project UPIN.

References

[1] T. Dufva and M. Dufva, "Grasping the future of the digital society," *Futures*, vol. 107, pp. 17–28, 2019.

[2] C. Hesselman, P. Grosso, R. Holz, *et al.*, "A responsible internet to increase trust in the digital world," *Journal of Network and Systems Management*, vol. 28, no. 4, pp. 882–922, 2020.

[3] R. Bazo, L. Boldrini, C. Hesselman, and P. Grosso, "Increasing the transparency, accountability and controllability of multi-domain networks with the upin framework," in *Proceedings of the ACM SIGCOMM 2021 Workshop on Technologies, Applications, and Uses of a Responsible Internet*, 2021, pp. 8–13.

[4] A. Clemm, L. Ciavaglia, L. Z. Granville, and J. Tantsura, "Intent-Based Networking - Concepts and Definitions," Internet Engineering Task Force, Internet-Draft draft-irtf-nmrg-ibn-concepts-definitions-09, Mar. 2022, Work in Progress, 29 pp. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-irtf-nmrg-ibn-concepts-definitions-09.

[5] T. A. Khan, A. Mehmood, J. J. D. Ravera, A. Muhammad, K. Abbas, and W.-C. Song, "Intent-based orchestration of network slices and resource assurance using machine learning," in *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*, IEEE, 2020, pp. 1–2.

[6] Y. Han, J. Li, D. Hoang, J.-H. Yoo, and J. W.-K. Hong, "An intent-based network virtualization platform for sdn," in *2016 12th International Conference on Network and Service Management (CNSM)*, IEEE, 2016, pp. 353–358.

[7] L. Boldrini, R. Bazo, C. Hesselman, and P. Grosso, "Upin–a shift in network control from operator to end user," in *ICT Open 2021*, 2021.

[8] D. Soni and A. Makwana, "A survey on mqtt: A protocol of internet of things (iot)," in *International Conference On Telecommunication, Power Analysis And Computing Techniques (ICTPACT-2017)*, vol. 20, 2017, pp. 173–177.

[9] Snowpack, *How snowpack works*, https://www.snowpack.dev/concepts/how-snowpack-works, Accessed: 12=06-2022, 2020.

[10] S. H. Haji, S. Zeebaree, R. H. Saeed, *et al.*, "Comparison of software defined networking with traditional networking," *Asian Journal of Research in Computer Science*, pp. 1–18, 2021.

[11] M. Pham and D. B. Hoang, "Sdn applications-the intent-based northbound interface realisation for extended applications," in *2016 IEEE NetSoft conference and workshops (NetSoft)*, IEEE, 2016, pp. 372–377.

[12] A. Koshibe, *Intent framework*, https://wiki.onosproject.org/display/ONOS/Intent+Framework, Accessed: 26-05-2022, 2016.

[13] E. Zeydan and Y. Turk, "Recent advances in intent-based networking: A survey," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, IEEE, 2020, pp. 1–5.

[14] OpenDayLight, *Nemo:user manual*, https://wiki-archive.opendaylight.org/view/NEMO:User_Manual, Accessed: 25-01-2022.

[15] ——, *Network intent composition*, https://wiki.opendaylight.org/display/ODL/Network+Intent+Composition, Accessed: 25-01-2022, 2022.

[16] M. Riftadi and F. Kuipers, "P4i/o: Intent-based networking with p4," in *2019 IEEE Conference on Network Softwarization (NetSoft)*, IEEE, 2019, pp. 438–443.

[17] A. S. Jacobs, R. J. Pfitscher, R. A. Ferreira, and L. Z. Granville, "Refining network intents for self-driving networks," in *Proceedings of the Afternoon Workshop on Self-Driving Networks*, 2018, pp. 15–21.

[18] A. S. Jacobs, R. J. Pfitscher, R. H. Ribeiro, *et al.*, "Hey, lumi! using natural language for {intent-based} network management," in *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, 2021, pp. 625–639.

[19] R. A. Addad, D. L. C. Dutra, M. Bagaa, T. Taleb, H. Flinck, and M. Namane, "Benchmarking the onos intent interfaces to ease 5g service management," in *2018 IEEE Global Communications Conference (GLOBECOM)*, IEEE, 2018, pp. 1–6.

[20] Y. Tsuzaki and Y. Okabe, "Reactive configuration updating for intent-based networking," in *2017 International Conference on Information Networking (ICOIN)*, IEEE, 2017, pp. 97–102.

[21] M. Kiran, E. Pouyoul, A. Mercian, B. Tierney, C. Guok, and I. Monga, "Enabling intent to configure scientific networks for high performance demands," *Future Generation Computer Systems*, vol. 79, pp. 205–214, 2018.

[22] Q. Chen, Z. Zhuo, and W. Wang, "Bert for joint intent classification and slot filling," *arXiv preprint arXiv:1902.10909*, 2019.

[23] K. Chowdhary, "Natural language processing," *Fundamentals of artificial intelligence*, pp. 603–649, 2020.

[24] H. Weld, X. Huang, S. Long, J. Poon, and S. C. Han, "A survey of joint intent detection and slot-filling models in natural language understanding," *arXiv preprint arXiv:2101.08091*, 2021.

[25] R. Technologies, *Introduction to rasa open source*, https://rasa.com/docs/rasa/, Accessed: 12-06-2022, 2021.

[26] R. K. Sharma and M. Joshi, "An analytical study and review of open source chatbot framework, rasa," *International Journal of Engineering Research and*, vol. 9, no. 06, 2020.

[27] S. Pérez-Soler, S. Juarez-Puerta, E. Guerra, and J. de Lara, "Choosing a chatbot development tool," *IEEE Software*, vol. 38, no. 4, pp. 94–103, 2021.

[28] R. Jindal, R. Malhotra, and A. Jain, "Techniques for text classification: Literature review and current trends.," *webology*, vol. 12, no. 2, 2015.