INTERNATIONAL TELECOMMUNICATION UNION

TELECOMMUNICATION STANDARDIZATION SECTOR

STUDY PERIOD 2022-2024

FOCUS GROUP ON AUTONOMOUS NETWORKS (FG-AN)

AN-I-xx

Original: English

Question(s): NA February 2023 (TBC)

INPUT DOCUMENT

Source: InnovNet

Title: Report on activities for Build-a-thon 2022 from InnovNet team: Deriving new use cases based on link prediction algorithm

Contact: Tarek Mohamed E-mail: tarek.mohamed.abdullah@gmail.com

Contact: Vishnu Ram OV
Team Mentor E-mail: vishnu.n@ieee.org

Keywords: Autonomous Networks, Use case, Graph Database, Link predictions, Parsing

Abstract: Deriving new use cases for the autonomous networks can help us understand the potential of using autonomous networks, in this trial we got a document that contains old use cases between some network components, and from these old ones we apply the link prediction algorithm to investigate the future opportunities of making use of autonomous network, we have reached 86% of accuracy of predicting if the link exists or not, and as a start we predicted more than 6.5k possible relations that can be investigated further.

Summary

In this report we made some steps to make the final progress on making predictions for potential use cases, these steps can be as follows:

- Reading the document and getting all requirements of all use cases.
- Converting figures into parsable formats.
- Converting parsed requirements and figures into graph representation.
- Training Link Prediction algorithm of the given relations.
- Predicting new relations between actors.

1 References

1. [FGAN-use cases] ITU-T Focus Group Autonomous Networks Technical Specification "Use cases for Autonomous Networks" https://www.itu.int/en/ITU-T/focusgroups/an/Documents/Use-case-AN.pdf
2. [Build-a-thon 2022] https://github.com/vrra/FGAN-Build-a-thon-2022
3. [FG AN Arch framework] Architecture framework for Autonomous Networks, https://www.itu.int/en/ITU-T/focusgroups/an/Documents/Architecture-AN.pdf
4. [Link Prediction] Link Prediction pipeline, neo4j, https://neo4j.com/docs/graph-data-science/current/machine-learning/linkprediction-pipelines/link-prediction/
5. [Neo4j Graph database] Neo4j, Graph database, https://neo4j.com/
6. [PlantUML] PlantUML tool to represent figures, http://www.plantuml.com/plantuml/uml/SyfFKj2rKt3CoKnELR1Io4ZDoSa70000
7. [XML] XML, python library to process xml files, https://docs.python.org/3/library/xml.html
8. [Python Docx] Docx, A python library to process word documents, https://python-docx.readthedocs.io/en/latest/
9. [Diagrams.net] Diagrams.net, Online flowchart and diagrams maker, https://app.diagrams.net/
10. [Python] Python programming language.
11. [Presentation] InnovNet Final presentation, https://docs.google.com/presentation/d/1LubX__7ZXCu6qWEd6vR3fKIhABMCey7MZocPNpEs2IQ/edit?usp=sharing
12. [Github Repo for the code] Codes and reports Github repo, https://github.com/TarekMohamed1999/FGAN-build-a-thon

2 Abbreviations and acronyms

This document uses the following abbreviations and acronyms:

AN	Autonomous Networks
GDB	Graph Database
REF CODE	Reference Code on github
AI	Artificial Intelligence
ML	Machine learning
RF	Random Forest
UML	Unified Modelling Language
GUI	Graphical user interface
XML	eXtensible Markup Language
AUCPR	Area Under Curve Precision-Recall

3 Conventions

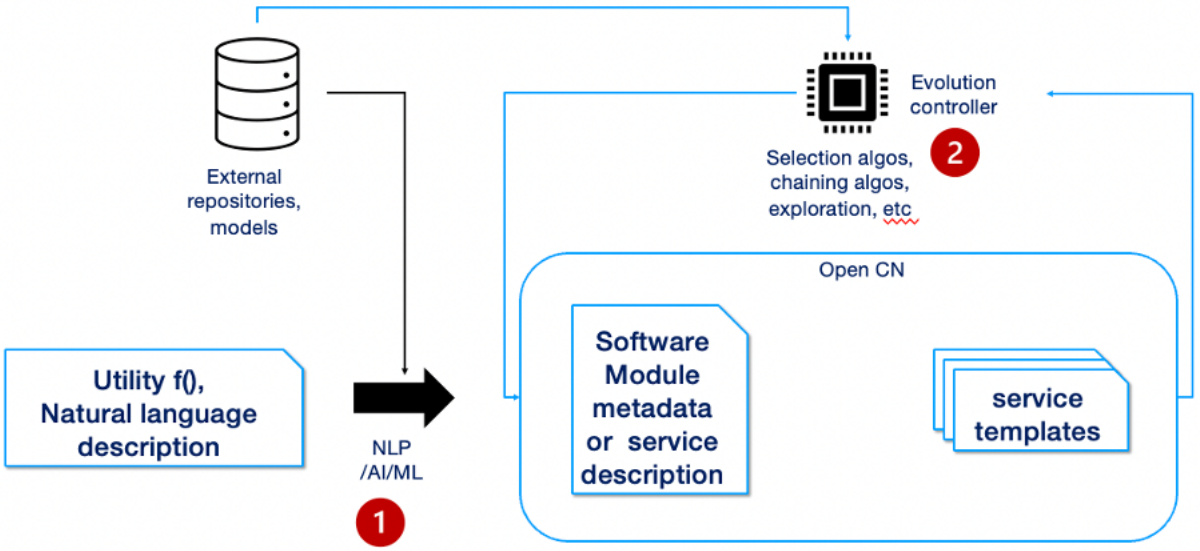
None

4 Introduction

It's important to investigate the hidden use cases for the autonomous networks tobe able to extract how to use the AN concepts in future networks effectively, and an effective way to do so is to make use of the recent use cases, to predict the possible ones.

Instead of analysing the existent use cases manually we can benefit from the huge advancements in the computational intelligence A.K.A. AI or ML, but for that purpose we need a dataset that contains information about the recent use cases.

In the [FGAN-I-289-R1] Fig 3, the following figure shows



In this figure, we analyse part-1 and For that we need to automate the process of getting information from [FGAN-use cases], and when we read it we found that we have 2 types of data, one is text, the second is graphs included in images.

Then, we need to represent this data into a format that is most suitable to capture the network structure and entities, we need an algorithm to run on this data to make predictions on the proposed links.

The document will be presented as follows: section 5 will be for Text and document parsing, section 6 will be for the figure parsing process, section 7 will be the graph database representation, section 8 will be for the link prediction algorithm and section 9 will be the conclusion.

5 Text And Document Parsing:

In [1], We can find the use cases for the build-a-thon challenge, and in order to make predictions for the new use cases we need first to get the existing details of the use cases to be able to represent them.

In that document, we can find tables, texts and images that are used to represent the use cases, in this section we will describe how we parsed the paragraphs and tables, and in the next section we will describe the parsing of images.

By using the docx library[8], which is a programming library based on Python programming language to process word documents, we can extract all the information we need from that document.

First we need to extract information that is found in the tables, because it represents the meta data and also the description of the use case itself, but the first problem we faced was we have a lot of tables other than use cases' one like the following tables, so we need to select those tables only by some sort of conditioning.

Use case id	FG-AN-usecase-001
Use case name	Use of knowledge in autonomous network
Base contribution	[FGAN-I-12-R1]
Creation date	21/January/2021
Use case context	Discussions during ITU webinar on autonomous networks (3 November 2020)
Use case description	To satisfy the key concepts of autonomous networks (evolution, experimentation and adaptation) while minimizing human intervention requires knowledge. This

Table 1: Use cases Table

Contributors:	Abhay Shanker Verma TEC, Ministry of Communications India	Email: as.verma@gov.in
	Abhishek Dandekar Fraunhofer HHI Germany	Email: abhishek.girish.dandekar@hhi-extern.fraunhofer.de
	Abhishek Thakur Institute for Development and Research in Banking Technology (IDRBT) India	Email: AbhishekT@idrbt.ac.in

Table 2: Contributors table

4 Abbreviations

AI	Artificial Intelligence
AN	Autonomous Networks
CI/CD	continuous integration and continuous delivery
CN	Controller
ER	Emergency Response
GNN	Graph Neural Networks
GUI	Graphical User Interface
IDSA	Inter-domain Service Automation
KB	Knowledge Base
KPI	Key Performance Indicator
LCM	Life Cycle Management

Table 3: Abbreviations Table

after Parsing those tables, we found that the fields on the tables is not consistent in naming, for example we have some problems like:

- [Category, Use case category, Notes on use case category] they all refer to the category of the use cases.
- [Base contribution, Base Contribution, Base contributions] All refers to the same information.
- [Use case description, Description, Use case description\n] All refers to the same information.
- [Open issues, Open issues (as seen by the proponent)] All refer to the same information.

So we need to make sure we have a consistent naming, we edited the confusion manually, and finally we got the following table that summarises all 40 cases:

	Use case id	Use case name	Base contribution	Creation date	Use case context	Use case description	Open issues	Use case category	Reference	Notes on priority of the use case
0	FG-AN-usecase-001	Use of knowledge in autonomous network	[FGAN-I-12-R1]	21/January/2021	Discussions during ITU webinar on autonomous n...	To satisfy the key concepts of autonomous netw...	- Representation mechanisms and transfer proto...	Cat 1: describes a scenario related to core au...	[b-Clark], [b-AN2020], [b-Jimenez-Ruiz], [b-My...	NaN
1	FG-AN-usecase-002	Configuring and driving simulators from autono...	[FGAN-I-12-R1]	21/January/2021	Discussions during ITU workshop on autonomous ...	To explore and experiment with various scenari...	Are simulators encapsulated in Sandbox? Or are...	Cat 1: describes a scenario related to core au...	[b-Y.ML-IMT2020-SANDBOX]	NaN
2	FG-AN-usecase-003	Peer-in-loop (including humans)	[FGAN-I-12-R1]	21/January/2021	Discussions during ITU workshop on autonomous ...	To guide the autonomous behaviour autonomous n...	Are some peers more equal than others (e.g. hu...	Cat 1: describes a scenario related to core au...	[b-Y.ML-IMT2020-MLFO]	NaN
3	FG-AN-usecase-004	Configuring and driving automation loops from ...	[FGAN-I-12-R1]	21/January/2021	Inspired by discussions on “demand mapping” du...	There are different automation loops in variou...	Where are the automation loops hosted? Are the...	Cat 1: describes a scenario related to core au...	[ITU-T Y.3173]	NaN
4	FG-AN-usecase-005	Domain analytics services for E2E service mana...	[FGAN-I-12-R1]	21/January/2021	Based on discussions with ETSI ZSM via [ML5G-I...	Section 6.5.3.2 of [ETSI ZSM ARCH] describes t...	Open issues (refer also those discussed in [ML...	Cat 2: describes a scenario related to applica...	[b-ETSI GS ZSM 002]	NaN
5	FG-AN-usecase-006	Automation and intelligent OAM(operation, main...	[FGAN-I-008]	NaN	NaN	Background: Dynamic radio environment, network...	NaN	Category 1 - Use case for autonomous behaviour	None	NaN

Table 4: Final parsed data from the tables.

Next step is to parse the requirements of each use case, the requirements in the document is represented as paragraphs after the table of each use case, and as an example, the first requirement in the 10th use case is :

- AN-UC10-REQ-001: It is critical that autonomous networks (AN) consider inputs from industry vertical solution provider regarding the required service characteristics, using an intent-based mechanism, while deciding the development and deployment options for industry vertical applications and network services.

For this example we can find that we can extract everything about the requirement like the use case number and the number of the requirement itself, also the priority level of the requirement can be extracted, and we have 3 levels of priority will be as follows:

- Critical: Important and required for the use cases implementation.[1]
- Expected: Possible requirement which would be important but not absolutely necessary to be fulfilled.[1]
- Added Value: possible requirement which would be optional to be fulfilled (e.g., by an implementation), without implying any sense of importance regarding its fulfilment.[1]

Finally We managed to get the final table that represents all the requirements as follows:

	id	001	002	003	004	005	006	007	008	009	010
0	UC001	It is critical that AN enable exchange of kno...	It is critical that AN enable optimization of...	It is critical that AN enable creation of rep...	It is critical that AN enable exchange of kno...	It is critical that AN use knowledge base for...	It is expected that AN enable exchange of kno...	it is of added value that AN use Auto-control...	NaN	NaN	NaN
1	UC002	It is critical that AN components arrive at p...	It is critical that autonomous networks (AN) ...	It is critical that autonomous networks (AN) ...	It is critical that autonomous networks (AN) ...	NaN	NaN	NaN	NaN	NaN	NaN
2	UC003	It is critical that autonomous networks (AN) ...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	UC004	It is critical that autonomous networks (AN) ...	It is critical that autonomous networks (AN) ...	It is critical that closed loops monitor the ...	It is critical that AN components consider th...	NaN	NaN	NaN	NaN	NaN	NaN
4	UC005	It is critical that AN support discovery and ...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	UC006	It is critical that autonomous networks enabl...	It is critical that AN enables data quality a...	It is critical that AN enables capturing and ...	It is expected that AN uses AI and big data t...	It is of added value that a varying set of KP...	It is of added value that AN solutions may be...	NaN	NaN	NaN	NaN
6	UC007	It is critical that autonomous networks (AN) ...	It is expected that autonomous networks (AN) ...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Table 5: Final requirements table.

Now we have now two tables, one for requirements and the other one for the information about the use cases, now we need to combine those tables together to have all details in one place, and the final details will be as follows:

Open issues	Use case category	Reference	Notes on priority of the use case	[redacted]	006	006_importance	007	007_importance	008	008_importance
- Representation mechanisms and transfer proto...	Cat 1: describes a scenario related to core au...	[b-Clark], [b-AN2020], [b-Jimenez-Ruiz], [b-My...	NaN		It is expected that AN enable exchange of kno...	expected	it is of added value that AN use Auto-control...	added value	NaN	NaN
Are simulators encapsulated in Sandbox? Or are...	Cat 1: describes a scenario related to core au...	[b-Y.ML-IMT2020-SANDBOX]	NaN		NaN	NaN	NaN	NaN	NaN	NaN
Are some peers more equal than others (e.g. hu...	Cat 1: describes a scenario related to core au...	[b-Y.ML-IMT2020-MLFO]	NaN		NaN	NaN	NaN	NaN	NaN	NaN
Where are the automation loops hosted? Are the...	Cat 1: describes a scenario related to core au...	[ITU-T Y.3173]	NaN		NaN	NaN	NaN	NaN	NaN	NaN
Open issues (refer also those discussed in [ML...	Cat 2: describes a scenario related to applica...	[b-ETSI GS ZSM 002]	NaN		NaN	NaN	NaN	NaN	NaN	NaN
	Category 1 -				It is of					

Table 6: Final combined tables of the requirements.

Now we have all the details about each use case in a row, next step we need to parse the figures that represent the relations between actors.

6 Figures representation:

In the document, we saw two types of figures, one for possible components and the other in a form of sequence diagram, both representing the interactions between the actors.

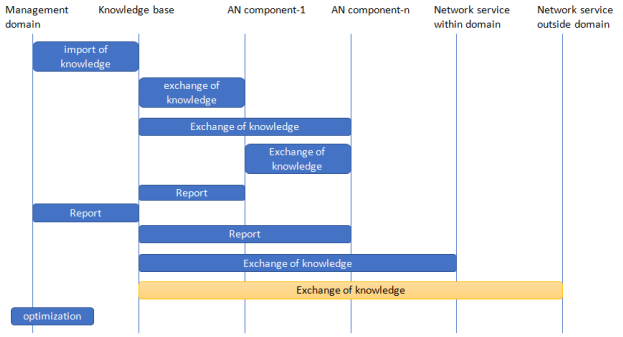


Figure 1: Example Actor Interaction diagram

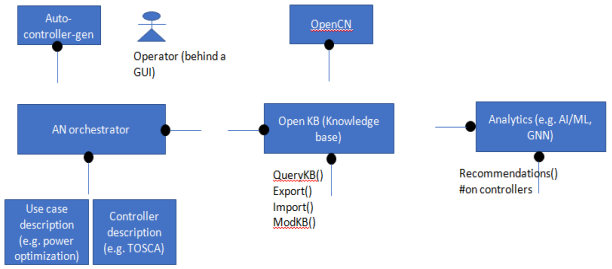


Figure 2: Possible components diagram

The problem with images is that they are not parsable, we cannot extract the desired information from these images in an automated fashion, so to be able to extract the information, we will convert it to a machine readable format and then extract the desired information.

To do that job we tried several tools to make a convenient representation, first tool is PlanUML[6], and it's a tool to make several types of diagrams using the UML format, by using this tool we are able to create sequence diagrams, and the result our first try was as in the next figure:

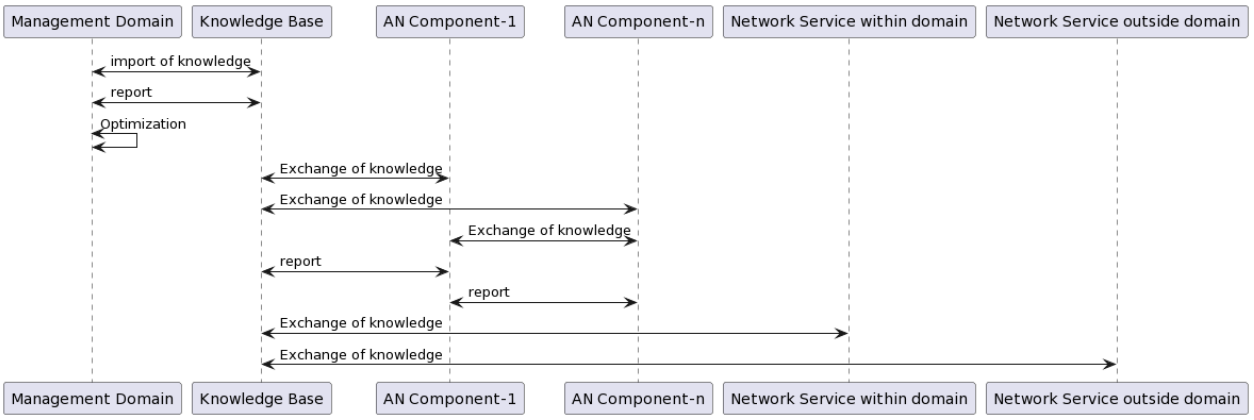


Figure 3: First try of using PlantUML

By comparing this figure and and figure 1 we can see that this representation is most like the desired one, but has one but crucial disadvantage, which is the optimization function of the management domain here is represented as a self message, but in the real time it's a function inside the management domain component.

Then we tried another representation, in this time we tried another type available in PlantUML, and the result was as follows:

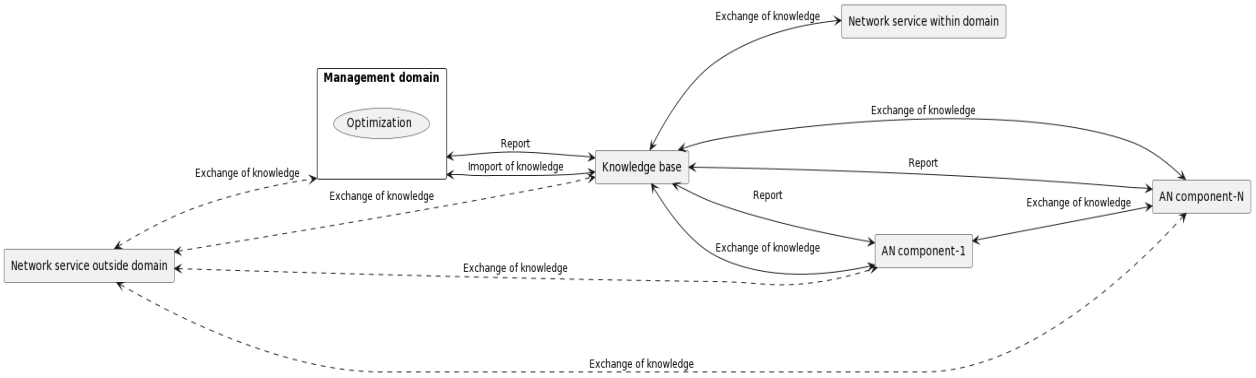


Figure 4: Second try result of figure representation using PlantUML

This representation is sufficient if we don't want to see a visual representation, here we overcame the problem of representing the function, but we have a bad visual that is not like the one we want to be consistent.

The final try we used another tool to make the representation, this time we used Draw.io[9], which is a GUI based tool, from which we can extract the XML or any other machine readable format, and the result was as follows:

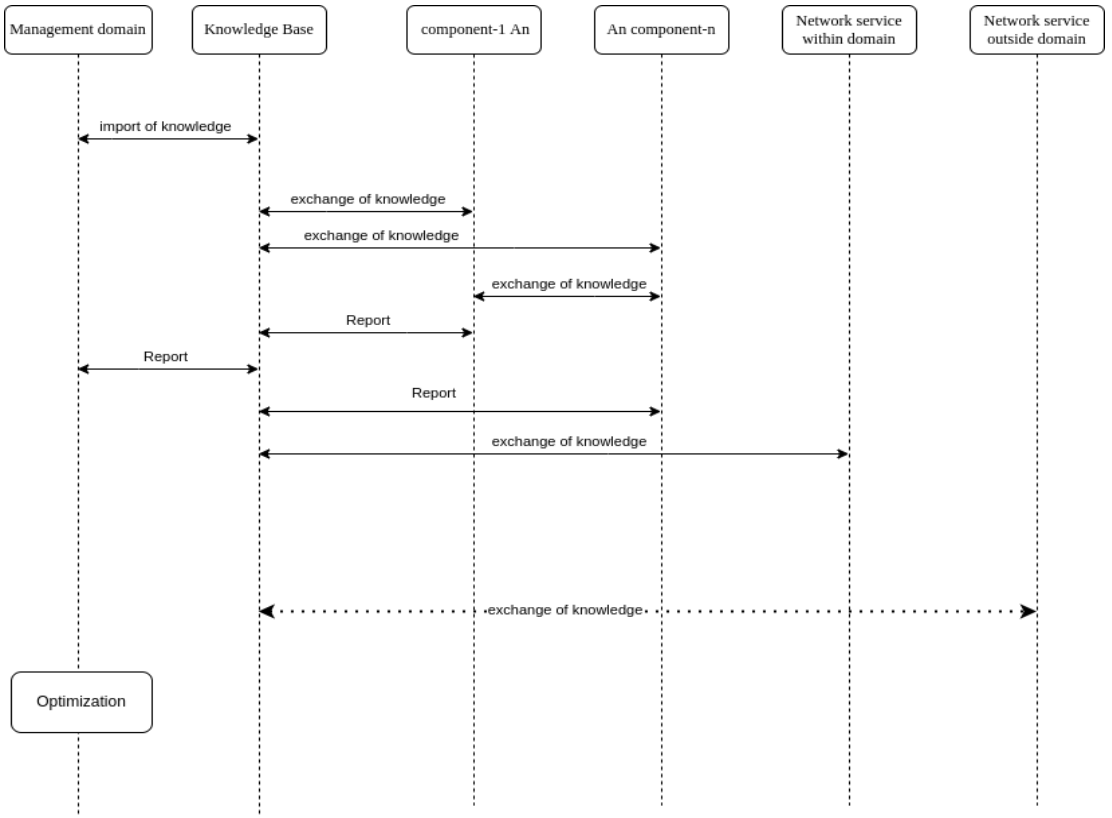
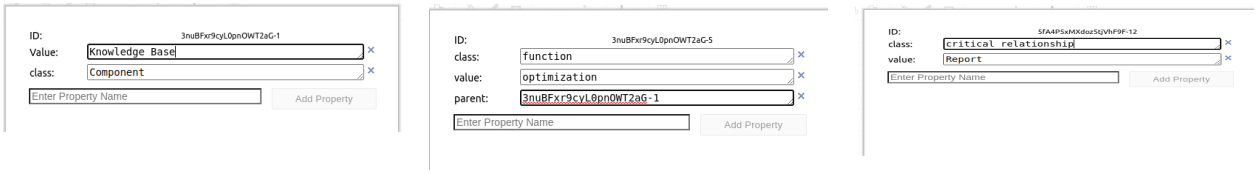


Figure 5: Final try in figure representation.

As we can see here an identical copy of the diagram in figure 1, also there are other types of shapes that can represent any other type of diagrams, also it gives us the flexibility to enter any other metadata that can help us when we parse the figures, and we used these metadata as follows:



(a) Component metadata (b) Functions Metadata (c) Relations Metadata

Figure 7: Added Metadata

We added the following meta data:

- Class: To represent the class of the item, whether it was component, function or the type of the relation, this field is added to all items in the figure.
- Value: The name of the item or the relation to all items.
- Parent: To contain the parent component ID that contains this function, only included in the function items.

After successfully representing the figure in the XML, which is a machine readable format, we can use Python[10] and its library XML[8] to parse the figure, and we ended up with the following table:

	source	target	message	class	id
0	Knowledge base	AN component-N	exchange of knowledge	critical relationship	UC001
1	AN Component-1	AN component-N	exchange of knowledge	critical relationship	UC001
2	Knowledge base	AN Component-1	Report	critical relationship	UC001
3	Knowledge base	AN Component-1	exchange of knowledge	critical relationship	UC001
4	Management domain	Knowledge base	import of knowledge	critical relationship	UC001
...	...	...	...	...	...
125	AN_1	AN_3	negative acknowledgement for the unicast reply	critical relationship	UC040
126	AN_1	AN_2	Configuration and context setup request requir...	critical relationship	UC040
127	AN_2	AN_1	context setup response	critical relationship	UC040
128	AN_1	AN_2	AN_1 configures the network underlays for rout...	critical relationship	UC040
129	AN_1	AN_2	AN_1 pays AN_2 for the service	critical relationship	UC040

Table 7: Relations Table

And also we have a dictionary that contains the functions and its corresponding parent actors.

Now we have completed parsing the document and the following figure represents the what we did till now:

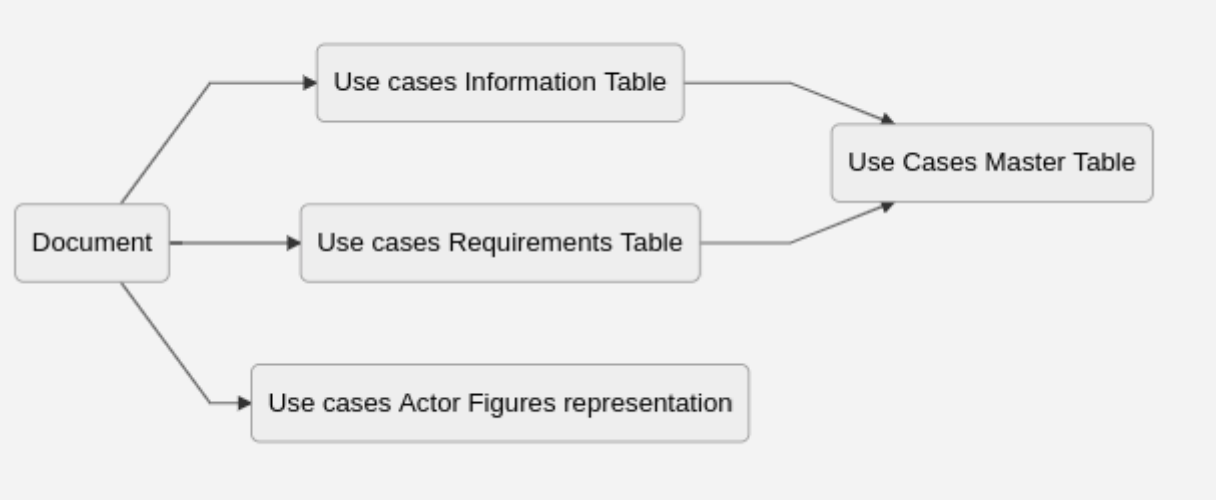


Figure 8: Resources we have

After that, we need a way to convert this into a type of dataset to be able to run machine learning algorithms on it.

7 Graph database representation:

There are several graph databases, we chose Neo4j[5] which is one of the most popular graph databases. Also it has some advantages like native representation of graphs, a lot of other packages especially in data science, has its Querying language Cypher which help us in query graphs and Finally Has an API with Python so it can help us automating our stuff.

It's more convenient to represent the network with a graph database, in the graph database we represent the data in the form of nodes and relations, which is the most relevant format for network structure.

We made some iterations to represent, the first try based only in the relations table, which contains only the components in the relations table with 130 relations only, but this representation has some drawbacks which is special to those use cases that have figures to represent it on the document, which are not all use cases, so using only these relations will ignore important parts of the use cases.

Next try is to read each use case carefully and extract the true relations and functions inside the graphs and also the hidden relations in the requirements and descriptions of each use case, and the final product is the reference code, which is sufficient to represent the use cases, now we have +500 relations and +300 Component.

A part of our representation is in the following figure:

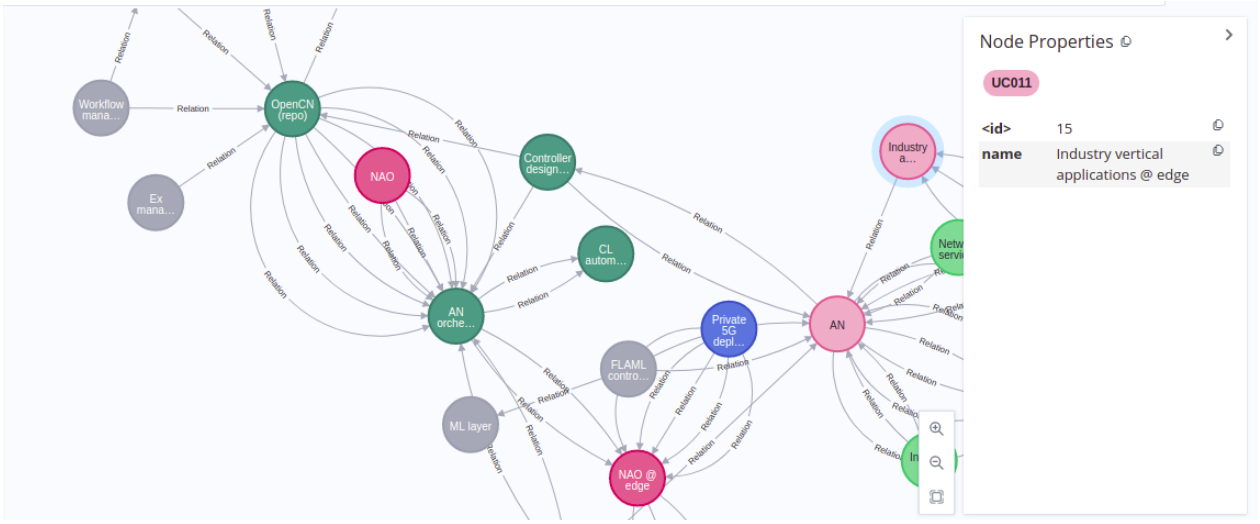


Figure 9: Part of the graph representation.

Now we have represented the data, next step is to run the link prediction algorithm on it to be able to investigate new use cases.

8 Link Prediction Algorithm:

We can consider the link prediction pipeline as a classification problem to classify if a certain relation exists between two nodes or not, and this classification can be more accurate if we calculate something that measures the how likely these two nodes to have a relation in between, and actually the current representation needs some numeric properties.

And the pipeline of the link predictions contains several steps as follows:

8.1 Make the projection:

First step is to project the graph database in the memory, this is done by using the native projections in neo4j.

8.2 Add node properties:

We need to add some properties to the nodes, these nodes are:

8.2.1 Node Embeddings:

For each node we need specific embedding like what is happening when we decode a categorical feature in the ordinary data preprocessing.

Node embedding algorithms compute low-dimensional vector representations of nodes in a graph. These vectors, also called embeddings, can be used for machine learning.

We have used the FastRP algorithm because it's the only one in production quality, and works in directed and undirected relationships.

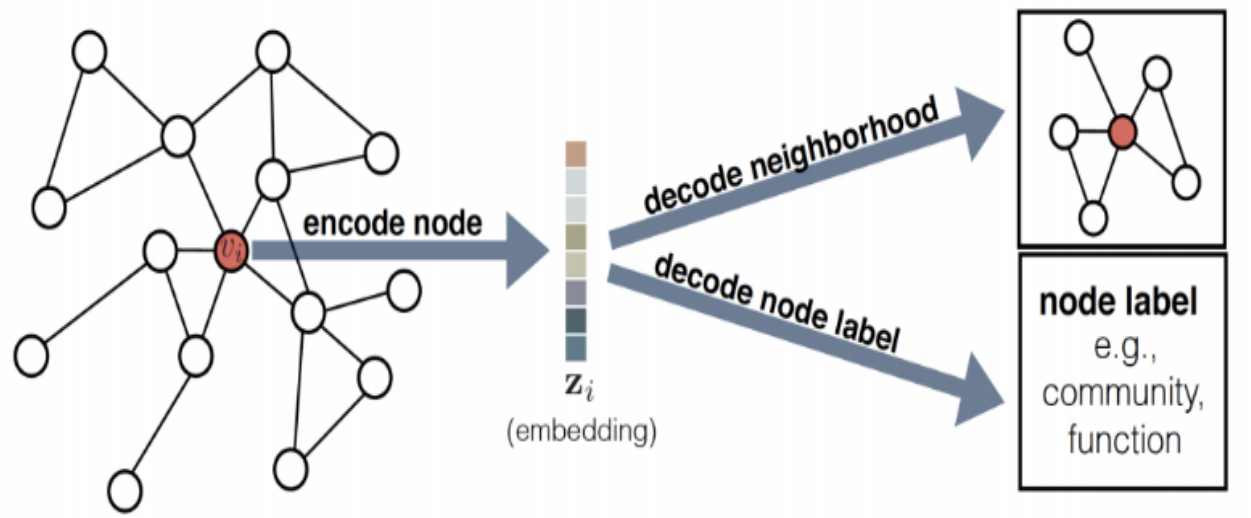


Figure 10: Node embedding

8.2.2 Centrality:

Centrality algorithms are used to determine the importance of distinct nodes in a network.

We have used the Degree Centrality algorithm, as it works well in directed and undirected relationships.

degree centrality measures the number of incoming or outgoing (or both) relationships from a node, depending on the orientation of a relationship projection.

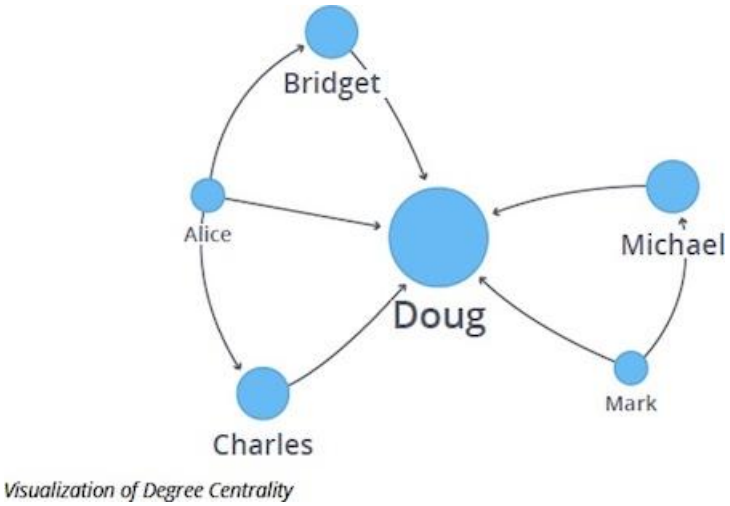


Figure 11: Example of Centrality

8.2.3 Community detection:

Community detection algorithms are used to evaluate how groups of nodes are clustered or partitioned, as well as their tendency to strengthen or break apart.

We have used the Label propagation algorithm, as it works well for directed and undirected relationships.

The Label Propagation algorithm (LPA) is a fast algorithm for finding communities in a graph. It detects these communities using network structure alone as its guide, and doesn't require a pre-defined objective function or prior information about the communities.

Now we have added all the properties that may help the algorithm more and more, next step is to calculate combined features using these properties.

8.3 Combined Features:

We are using the properties that are created on nodes, and combine them in each potential link According to one of the following functions we choose for every property:

L2	$f = [(s_1 - t_1)^2, (s_2 - t_2)^2, ..., (s_d - t_d)^2]$
HADAMARD	$f = [s_1 * t_1, s_2 * t_2, ..., s_d * t_d]$
COSINE	$f = \frac{\sum_{i=1}^{d} s_i t_i}{\sqrt{\sum_{i=1}^{d} s_i^2}\sqrt{\sum_{i=1}^{d} t_i^2}}$

Figure 12: Similarity algorithms.

8.4 Training and prediction:

Now all of the pipeline is ready, we trained algorithms in the dataset and after making hyper tuning we found that the RF model behaves best in all use cases, now we need to test the models in all scenarios:

- First scenario: we use only the 130 relation data set, which is the result of using only use cases that have figures on it.
- Second scenarios: we use the full representation of the reference code, that have +500 relations and +300 Nodes.
- Third scenario: we use the second scenario but we add the functions of the actors as properties in these nodes.

And the following was the results in each try:

Representation	Best Model	Test Accuracy (AUCPR)
Scenario 1	RF	88 %
Scenario 2	RF	73.6%
Scenario 3	RF	86 %

Table 8: Results

Now it's obvious that the third scenario is the best scenario, and when we made the prediction, we found that 6954 new links with more than 99% of probability to be a true relation.

9 Conclusion and final mark:

We can conclude from all of our findings that we found a way to parse the document and text, also the draw.io is the best tool that has a lot of customizations that can help us in making more representative figures.

Best algorithm for link prediction is the Random Forest model with 86% AUCPR accuracy in the full reference code, and also we need to find a more enhanced way to represent the use cases in the reference code, i.e. to convert all functions to be properties.

We need a way to include parsable shapes inside the document of use cases instead of images, to help automate the process further.

Also we need more representative figures or an algorithm for example and NLP to convert the descriptions of use cases into the reference code.