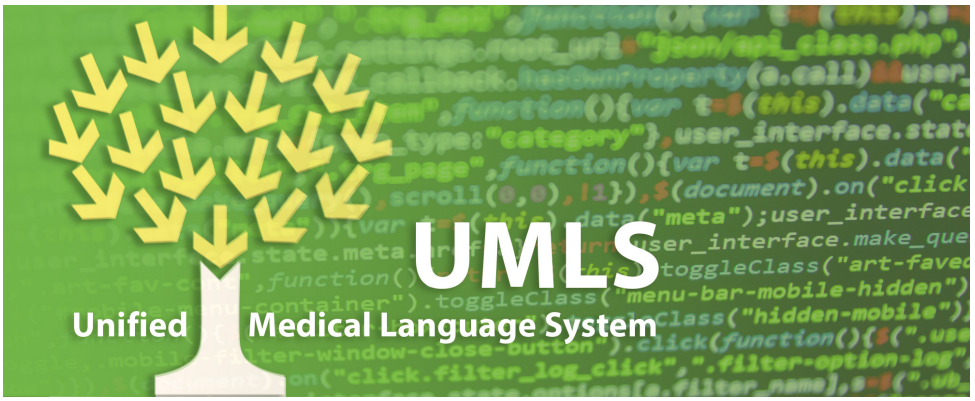
Alexandre JUAN - Promotion 2024
3rd year student at EPITECH
3 Place Paul Bec
34000 Montpellier

Internship Report

Machine-learning based automatic assignment of Semantic Types to ontology concepts



Internship from 04/01/2022 to 07/31/2022 in Montpellier, 2 Place Pierre Viala 34060, France, in MISTEA research lab

Ack

First of all, I would like to thank the Montpellier INRAE MISTEA department (Mathematics, Informatic and STatistic for the Environment and Agronomy) for their confidence. They trusted me in summer 2021 giving me the opportunity to work on the OpenSilex project. I learned a lot and upgraded my hard skills in the semantic web domain. Thanks to this experience, I could apply to a Machine Learning intern offer in the MISTEA department.

Special thanks to my internship supervisor, Clement Jonquet, who gave me the opportunity to work for the first time in Artificial Intelligence. Moreover, he trusted me, letting me be independent enough in my research without leaving me alone.

Finally, I address my thanks to all my colleagues who made me enjoy my days more than they realize. Thank you very much!

1- Internship

Introduction

Before going into the project objectives, it is very important to understand what **UMLS** is and to define precisely its rich vocabulary.

The Unified Medical Language System, or UMLS, brings together numerous vocabularies, terminologies and classifications in the biomedical field. This database allows interoperability between computer systems which rely on the terminologies. This set of terminologies was created in 1986 by the National Library of Medecine (USA), or **NLM**, and is still updated annually. After more than 35 years of existence, UMLS has currently more than 4.5 million different concepts. The UMLS has been populated over the years with data from over 220 different sources, some in foreign languages. So, the UMLS is not only made of English vocabularies even if English is the majority (our work is dedicated to the English part of UMLS, constituting the vast majority of the research data).

The UMLS is composed by three differents parts:

- the **Metathesaurus** (list of all concepts coming from each source and integrated in a unique (meta)thesaurus).
- the **Semantic Network** (categories and relationships of concepts).
- the Lexical Tools (lexical information but this part is ignored during all the project).

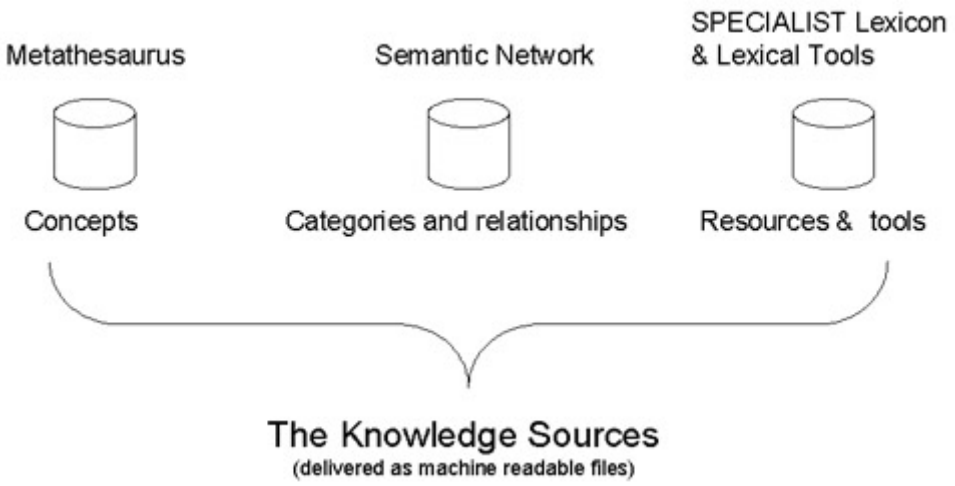


Figure 1. The UMLS diagram representation
https://www.nlm.nih.gov/research/umls/new_users/online_learning/OVR_001.html.

The Semantic Network allows to tag the concepts in the Metathesaurus with a high level dimension. This new dimension allows the scientist to categorize concepts such as diseases or molecules more simply.

To illustrate this idea, here is a diagram where all the concepts related to the heart are shown. And each of these concepts is linked to a semantic type. So there are several hierarchies in the UMLS.

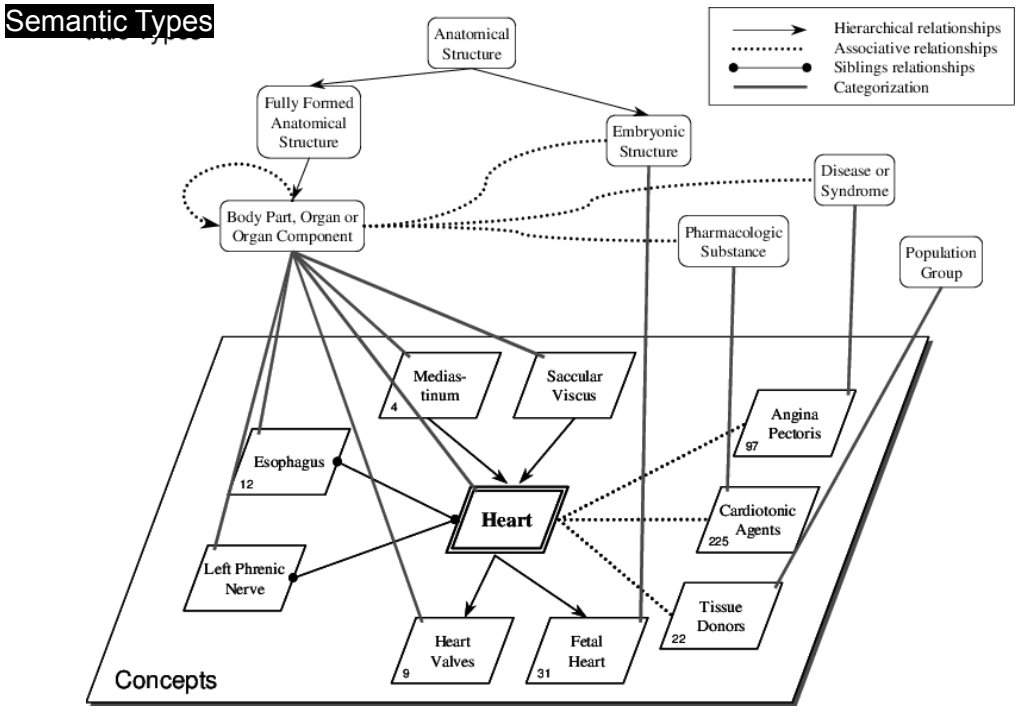


*Figure 2. **Semantic space** for the concept " Heart " (partial representation)*
https://www.researchgate.net/figure/Semantic-space-for-the-concept-Heart-partial-representation-Numbers-refer-to-the_fig1_242544690.

The Semantic Network (STY) is in particular composed of Type Unique Identifier (**TUI**) which are themselves classed in the Group of Unique Identifier (**GUI**).
As explained earlier, the UMLS is fed with vocabulary graphs from different sources. In order to agree between all the concepts, the UMLS has defined the Concept Unique Identifier (**CUI**), allowing us to navigate between terminologies despite the different sources.

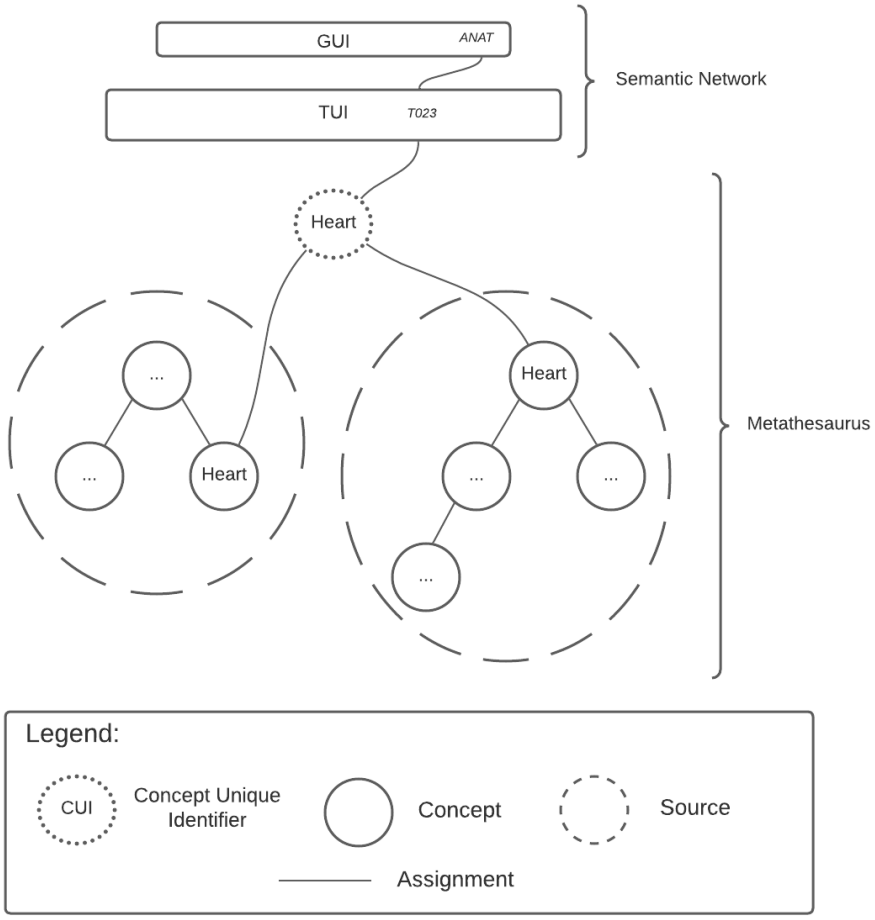


Figure 3. Schema illustrating the Metathesaurus and the Semantic Network with the "Heart" (CUI = C0018787) ***assignments*** *concept as an example.*

The goal of the project is to create a Machine Learning algorithm that automatically predicts, on a new data graph / **ontologies**, the TUI & GUI of each concept. For this purpose, we use the supervised Machine Learning method that consists of training our algorithm on terminologies already tagged and testing it with other terminologies and checking if the predictions are correct or not.

Here is an analogy to better understand the goal of the project:
We have lots of animal pictures from different social networks. The goal is, from these photos already assigned to an animal, to create an algorithm, using supervised Machine Learning, that is able to determine, from a new photo, which animal is in it.

Replace the animals with medical concepts that need to be tagged and you have a summary of the project's objective.

<u>We hypothesize that the model will predict the type of new terminologies.</u>

Several UMLS ontologies are not tagged. The model would perfectly fit with this problem. The long-term objective is to use the UMLS Tag Assignment model to be exploited and used in OntoPortal, more precisely in BioPortal, which is an alliance of researchers dedicated to promoting semantic services in scientific research based on shared domain-specific ontologies managed with the collaboratively developed OntoPortal software. BioPortal, developed by Stanford University, is a platform to easily find biomedical ontologies (from the Unified Medical Language System).

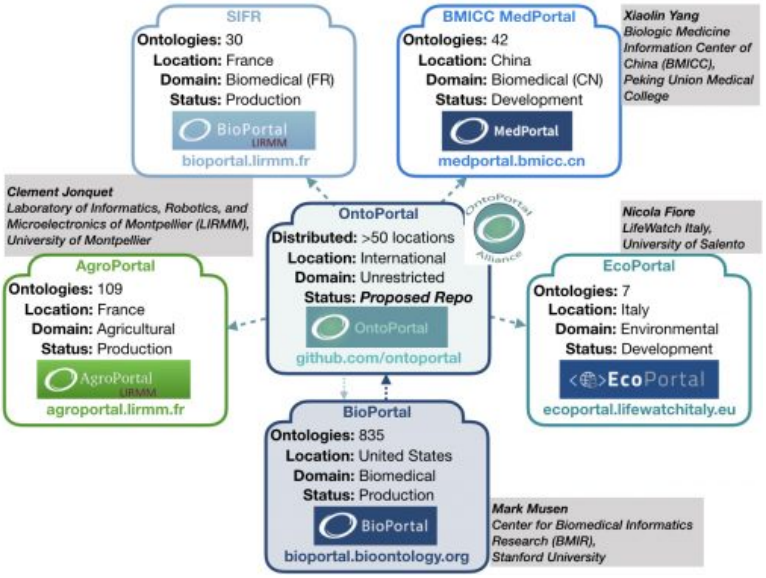


*Figure 4. Members and Systems of the **OntoPortal Alliance** https://ontoportal.org/.*

But behind the project's goal lies another approach. If we manage to obtain a fairly accurate model, it would assert the following hypothesis: "It is possible to create a semantic prediction model on a graph of ontology data". This would mean that we can create a semantic prediction model on a graph of ontology data in another context than biomedical. And what interests INRAE is to acquire a predictive model in the same style as this one for AgroPortal. AgroPortal is a repository of vocabularies and ontologies in the field of agronomy.

Feature selection process

In order to generate the most accurate model, it must be fed with essential features that are correlated to the type. Currently, the list of features used with the explanation of why we use them:

- Labels (as Bag-of-Words): The labels represent the most used words for CUI. This means that they have a great influence on which type (TUI/GUI) it is.
  Example: Assuming the vector size of all different labels is linked with the number of unique labels, let's say for example a bag-of-words vector like [illness, heart, …, melanoma]. The concept with "Heart" as a label will have [0, 1, …, 0] as a label bag-of-words vector.

- Definition (as **Word Embedding**), or Def: Most of CUI have definitions. The relationship between the definition words and the Semantic Network class should exist. In addition to Word Embedding, we use another binary feature named "Has_Definition" to indicate to the model if the CUI has or not a definition.
  Example: "Heart" definition in MSH is "The hollow, muscular organ that maintains the circulation of the blood". The Machine Learning algorithm, using the Gensim library, trains the word embedding model (Word2Vec) with every definition of all concepts and gives us the vector of the "Heart" definition in the Word2Vec trained model.

- Sources (as one-hot encoding), or SAB: As explained previously, a CUI can have several different sources. We know that there are sources that are specialized in certain domaines (disease, living organisms, chemist, etc.). Knowing the CUI sources could categorize CUI into Semantic Network easily.
  Example: "Heart" has SNOMEDCT_US and MSH as sources. For example, if the source one-hot encoding vector is [SNOMEDCT_US, WHOART, …, MSH], the "Heart" concept would have the source vector like [1, 0, …, 1].

- Parents_Types (as Bag-of-Words): As CUIs are arranged as graphs/trees, they have at least one parent (none if the CUI is at the very top of the tree). Knowing that some CUIs have multiple sources, this means that CUIs can have multiple parents from multiple different sources (and can have multiple types [TUI/GUI] but our algorithm predicts only one answer). This information is important since the parents' types are inherited by the children (inherited does not mean that the son CUI will have the same type as the parent CUI. The types are also in tree form with a hierarchy. This means that the child CUI can be typed by a child type of the parent CUI type).
  Example: "Heart", CUI = C0018787, has "Cardiovascular system", CUI = C0007226, as parent which is assigned "Body System", TUI = T022. For example, if the bag-of-words vector is like […, T021, T022, T023, …], the "Heart"'s parents type vector would be […, 0, 1, 0, …].

We also tried to implement other features that, in the end, were not kept during the selection of the features during the evolution of the project:

- The number of children: This information approximates the depth in the CUI data graph.
- Number of ancestors: Like the number of children, this feature would have told us whether the CUI was more towards the top of the tree, or more towards the bottom.

This information was not kept because of its inefficiency on the accuracy of the model.

Architecture

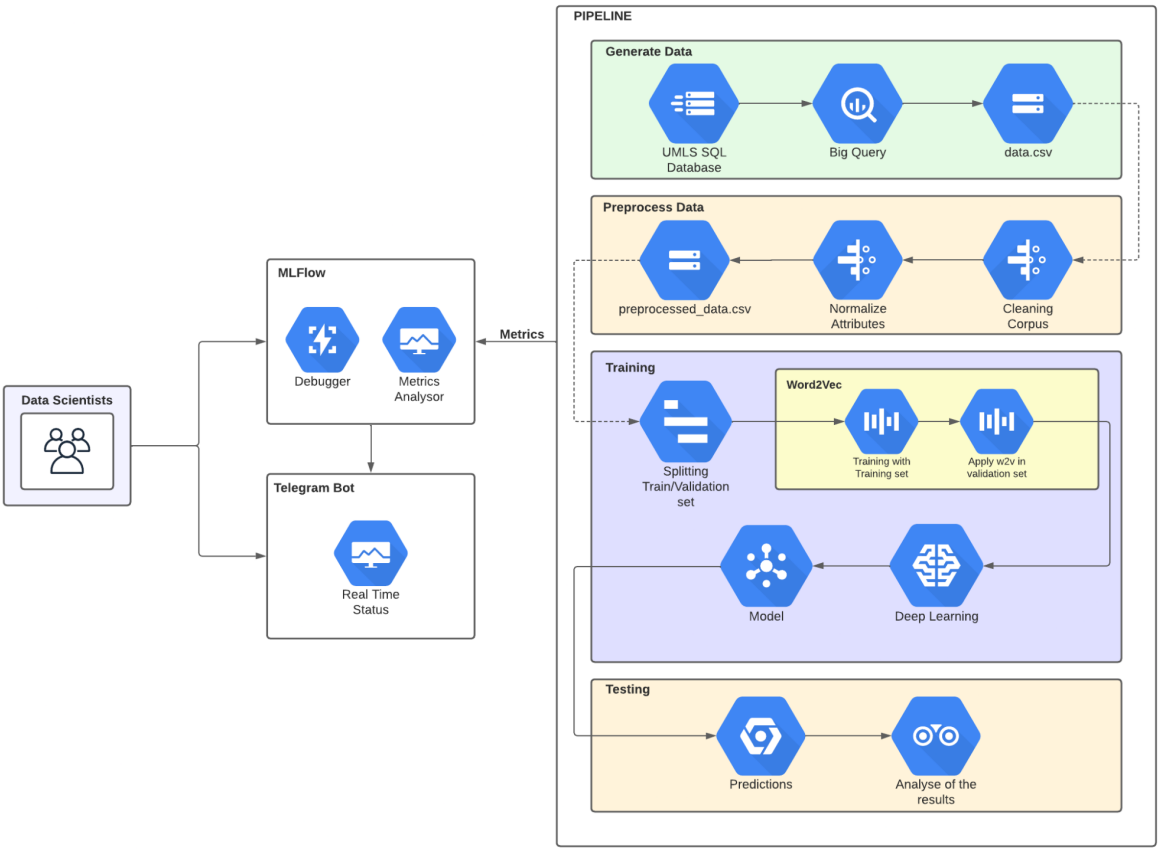


*Figure 5. Diagram to visualize the project's **architecture**.*

The most important part of the architecture is the **pipeline**. The pipeline is composed in 3 parts:

- Generate Data

  This is the first part of the pipeline. Its role is to retrieve the important data from the UMLS directly via the locally installed UMLS database via an installer available on the NLM site. We retrieve each CUI with its labels, its definitions, its sources, its TUI, its GUI and the TUI and GUI of its parent. All these data are stored in a file named data.csv.

- Preprocess Data

  This part follows the first one and requires data.csv to be executed. In this part, we have to process the missing data, normalize the numerical data and clean the textual data, especially the definitions. The latter may have symbolic links, useless words and all sorts of punctuation that are useless for the training of our algorithm. The preprocessing generates a new table of data saved in a file named preprocessed_data.csv.

- Training & Testing

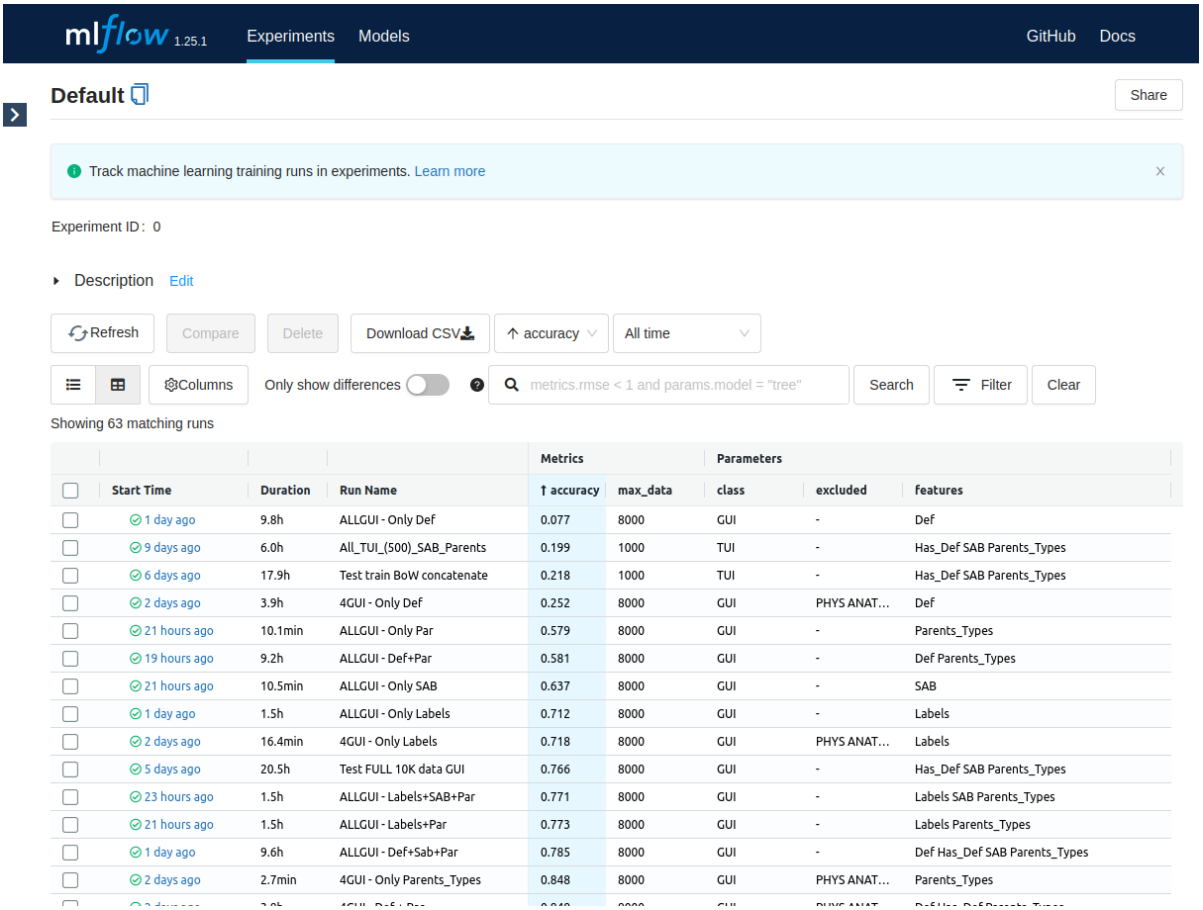
  The dataset is retrieved from the preprocessed_data.csv and is divided into 2 parts: a training set and a testing set. On the definitions of the training set, we train a Word Embedding model, from the Gensim library, and then we apply this model on the testing set (we used Word2Vec technology to resolve the sentences parameters problem. Word2Vec is not anymore state-of-the-art but it is tested in multiple kinds of the same projects with sentences used). After that, we generate a model defined according to the configuration of the config.yaml. This model, at the moment, is based on a complex and deep neural network. The model is generated using the Keras library. After training the model with the training set, the model will then predict its results on the testing set. We retrieve these predictions and see if these predictions are good or not.

The pipeline lifecycle is orchestrated by MLFlow, an open source platform that can be used to manage the lifecycle's runs of our Machine Learning algorithm. MLFlow allows us to save the information of each launched pipeline (pipeline configuration, precisions, tested features, training history, model graph, etc.), as well as to save the models each time.

In addition to MLFlow, there is a Telegram Bot that sends notifications when a pipeline succeeds or fails. It also allows to manage the number of pipelines running on the server and to know the status of the runs.

As mentioned before, the configuration of the pipeline can be customized directly in the config.yaml file. In this file, the number of classes used in the dataset can be changed, to select the features you want to experiment with, to modify the Machine Learning model etc. This file is very important because it allows you to customize the pipeline in a very complete way, both in the preprocessing and in the training part.

All the code of the project is open source and is available on Github. In this repository is a README.md listing the steps to get the project up and running. The project has been fully developed under Python version 3.8 and uses a Continuous Integration, with Github Action, to keep a continuous cleanliness report of the code.



				Metrics		Parameters		
	Start Time	Duration	Run Name	↑ accuracy	max_data	class	excluded	features
	1 day ago	9.8h	ALLGUI - Only Def	0.077	8000	GUI	-	Def
	9 days ago	6.0h	All_TUI_(500)_SAB_Parents	0.199	1000	TUI	-	Has_Def SAB Parents_Types
	6 days ago	17.9h	Test train BoW concatenate	0.218	1000	TUI	-	Has_Def SAB Parents_Types
	2 days ago	3.9h	4GUI - Only Def	0.252	8000	GUI	PHYS ANAT...	Def
	21 hours ago	10.1min	ALLGUI - Only Par	0.579	8000	GUI	-	Parents_Types
	19 hours ago	9.2h	ALLGUI - Def+Par	0.581	8000	GUI	-	Def Parents_Types
	21 hours ago	10.5min	ALLGUI - Only SAB	0.637	8000	GUI	-	SAB
	1 day ago	1.5h	ALLGUI - Only Labels	0.712	8000	GUI	-	Labels
	2 days ago	16.4min	4GUI - Only Labels	0.718	8000	GUI	PHYS ANAT...	Labels
	5 days ago	20.5h	Test FULL 10K data GUI	0.766	8000	GUI	-	Has_Def SAB Parents_Types
	23 hours ago	1.5h	ALLGUI - Labels+SAB+Par	0.771	8000	GUI	-	Labels SAB Parents_Types
	21 hours ago	1.5h	ALLGUI - Labels+Par	0.773	8000	GUI	-	Labels Parents_Types
	1 day ago	9.6h	ALLGUI - Def+Sab+Par	0.785	8000	GUI	-	Def Has_Def SAB Parents_Types
	2 days ago	2.7min	4GUI - Only Parents_Types	0.848	8000	GUI	PHYS ANAT...	Parents_Types

*Figure 6. **MLFlow** user interface (web interface that interacts with the pipeline via the python API library). We can find out if a run succeeded or failed, how long it took, each **configuration** per run and all generated **artifacts**.*

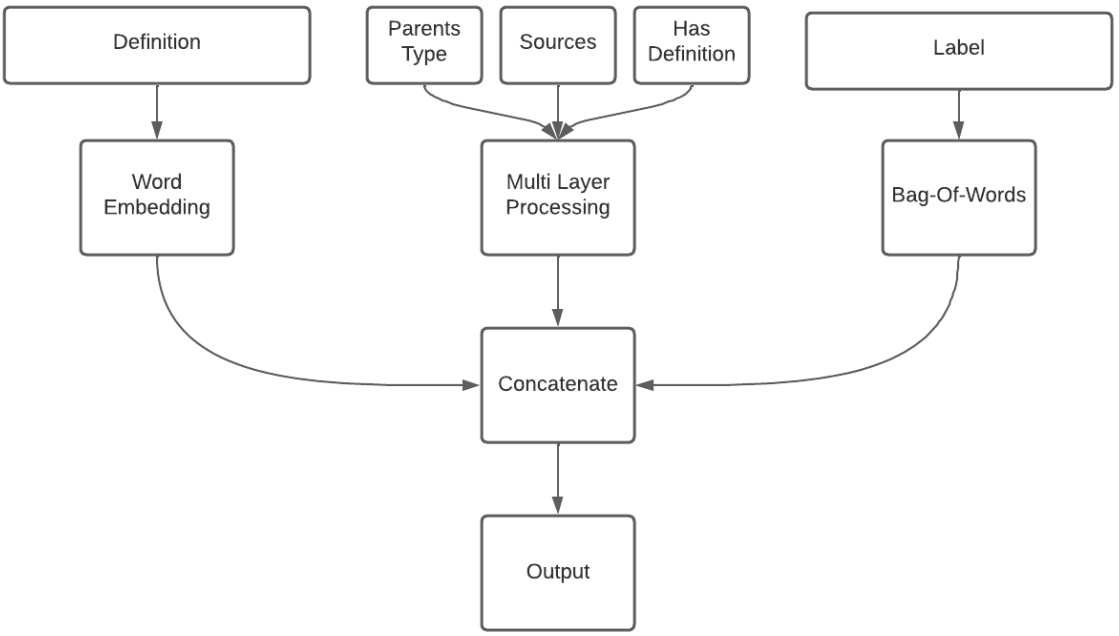


*Figure 7. Schema representing the **Machine Learning model**.*

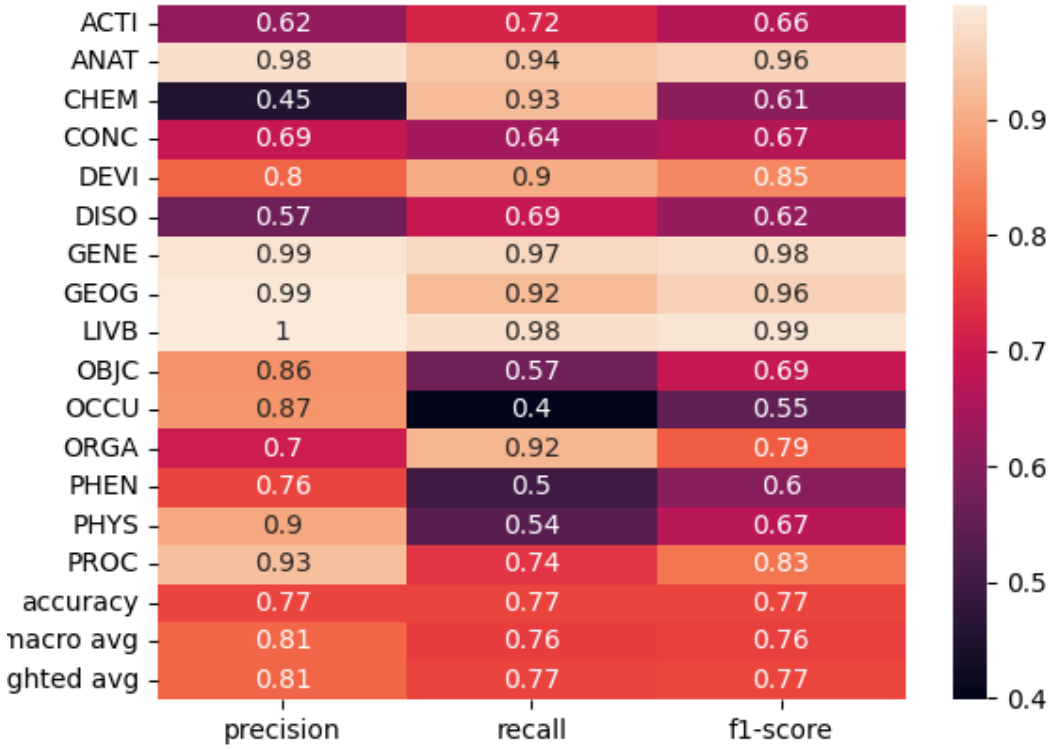


*Figure 8. Example of **predictions model results** (saved for each run in MLFlow). The results are represented by plot rectangular data as a color-encoded matrix. In the left corner, there are all the **classes tested** (in this case, it's GUI). Bottom, the **precision**, **recall** and **f1-score** are indicated. These scores measure the reliability of the prediction model according to the different classes evaluated.*

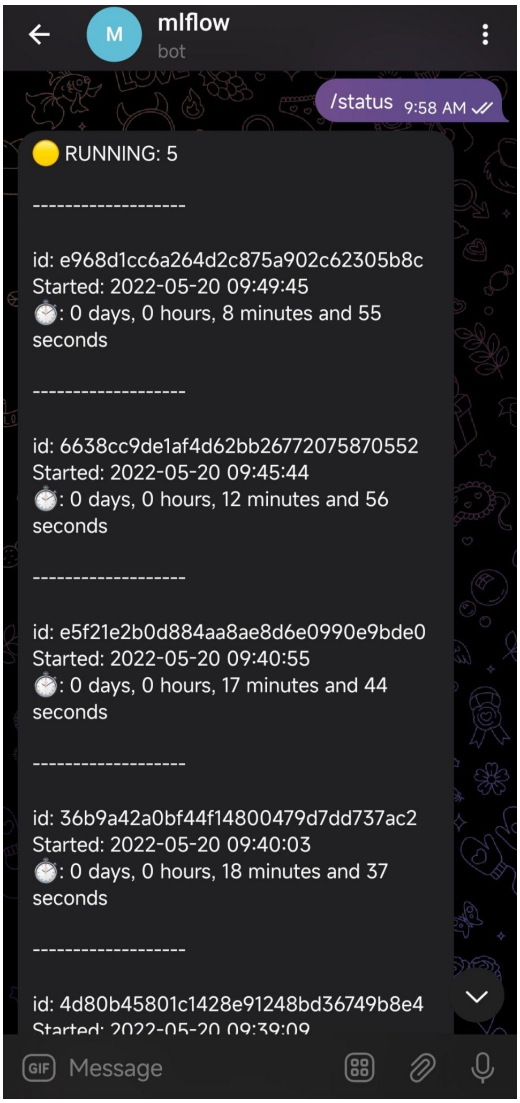


*Figure 9. Chat with the **Telegram Bot** to get **real-time information** on your phone.*

The UMLS being a rather large database, we had to set up a pipeline run architecture on 2 different servers:

- INRAE server: One PC, which is continuously switched on, is used as a server, especially for the generate data and preprocessing parts, since it holds the installation of the complete UMLS database (MySQL) locally. The computer is not very powerful so the training part of the Deep Learning model is done by the other server.
- Shared computing server managed by the University of Montpellier: A partition on this calc server has been allocated to us for the UMLS Tag Assignator project. The server has one node with 28 cores, 128 GB of RAM and 3 TB of unsaved storage. Despite the large resources available from the compute server, the UMLS database is far too large to be fully trained.

As the entire project code is hosted on Github, it allows data scientists to add features / fix errors from any computer. Then, via SSH, they connect to the remote servers in order to update the remote code and to be able to update the generate_data.csv, preprocessed_data.csv as well as the pipeline configuration itself.

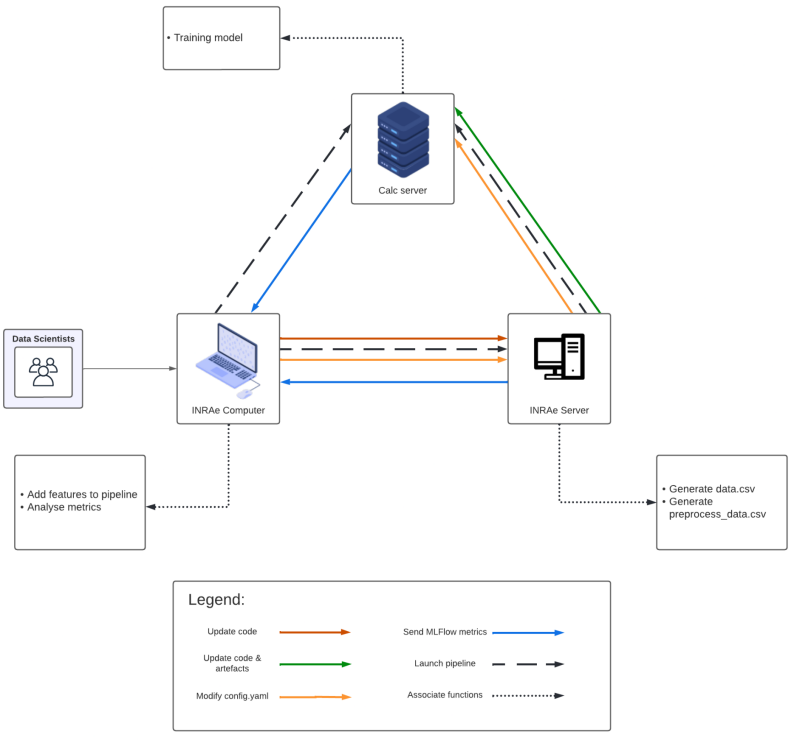


*Figure 10. Architecture of the **server network**. INRAE computer is the most used to debug and code while Calc server is the one with enough **resources** to launch the runs with custom parameters.*

Experiments

After developing the first version of our model, to validate or invalidate our hypothesis, we runned a dozen experiments. We could easily do this thanks to the generic architecture of the pipeline, changing the parameters quickly. We decided to describe 4 of them to show the progress of the progress.

The Semantic Network has a lot of different types but few types have much more data than others. The following diagrams show the imbalanced distribution.

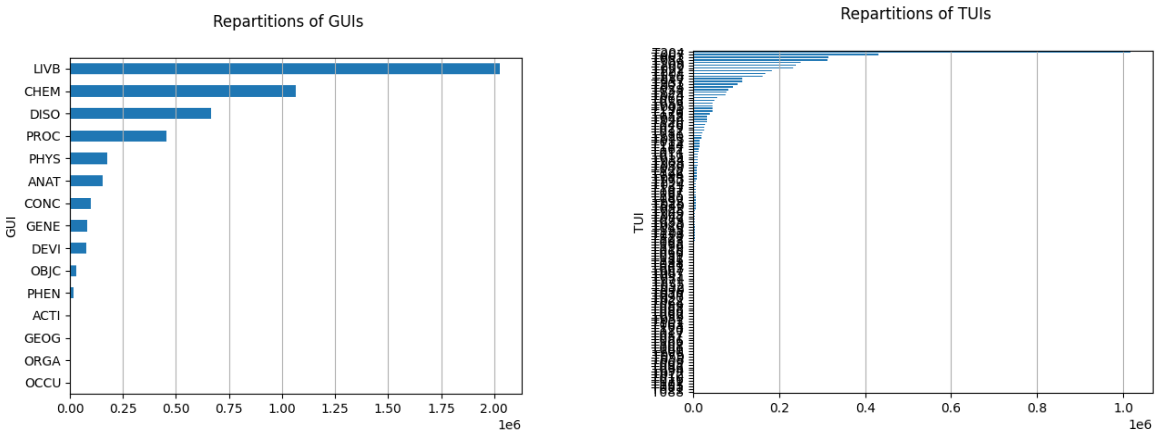


*Figure 11. **Imbalance data** representation for GUI and TUI. GUI has 15 classes while TUI has 127 (TUI representation is not perfectly readable due to the excessive number of classes)*

To avoid the imbalance of data, our dataset is regulated, balanced, from the start of the training set so that all classes have almost the same number of training and testing dataset. The “Maximum data per type” parameter represents the limit of the data per class.

And we need to select the “Maximum data per type” limit with multiple conditions. If we increase the parameter too much, let’s say 80K, the model is going to pick for every class a maximum 80K of data, but there are classes that have less than 80K data. So, the model will end with an imbalanced data. Moreover, as explained previously, the calc server can’t exceed a limit of maximum data per class depending on the number of classes analyzed. And reducing the value of the “Maximum data per type” is not a good idea because we want that our model trains with the maximum data possible to predict with the highest possible accuracy.

Experiment 1

Start date of the experiment: June 2nd 2022
End date of the experiment: June 9th 2022
Experiment goals:

- Identify essentials features to develop the most accurate model possible
- Observe the differences in results between a small sample of more complete types and a larger but unevenly distributed sample

Hypothesis: There is a significant difference of results between predicting 4 classes and predicting all GUI classes.

Experiment parameters:

- Training set : 70% / Testing set : 30%
- Maximum data per type : 8.000
- Semantic Network part studied: GUI
- Short type dataset: composed with “CHEM”, “DISO”, “LIVB” & “PROC” GUI
- Word Embeddings params: vector size=700, window=20, sequence length=100, epochs=40
- Features compared: Def=Definition + Has_Definition / Labels / SAB=Sources / Parents=Parents type
- We collected the f1-score for the diagram results

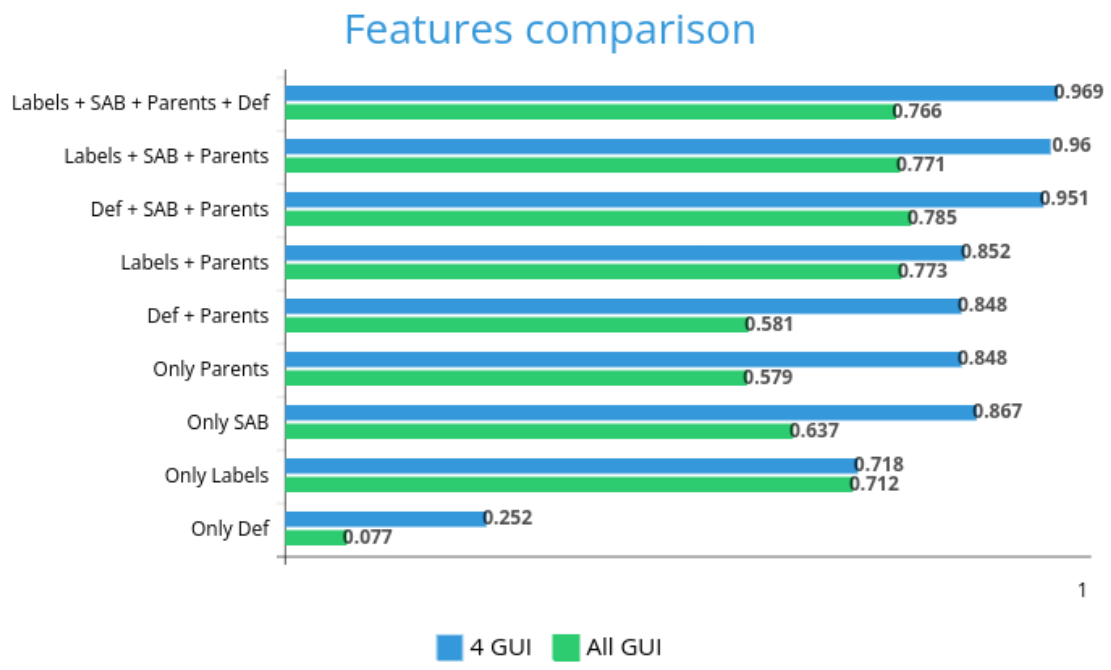


*Figure 12. GUI **Features comparison**. The results are from 0 to 1.*

The comments that we can make:

- The model is more precise with 4GUI than All GUI. The best f1-score of the model to predict with 4 GUI classes is closed to 97%.
- Despite all the features used in this experience, the model can't predict above the 80% of f1-score with all GUI.
- The Def feature seems to be useless because the f1-score of the model trained only with this feature is close to the random precision (1/4 and 1/15).
- There is a 20 percent difference between 4 GUI and ALL GUI. Our hypothesis is true if we consider a significant difference be 20 percent or more.
- With this experiment we learned, exploring the minority class results, that several unrepresented classes in the Semantic Network are predicted with higher accuracy than more represented classes. For example, in this results picture, GEOG GUI class has way better results than CHEM GUI class which is more represented in the Semantic Network (you can see here). This phenomenon can be explained by the fact that the GEOG CUIs have "discriminatory" informations which allow the model to achieve this results.

What we are going to do:

- Create a new experiment where we are going to check the reason for the Def useless parameter (Experiment 2).

Experiment 2

Start date of the experiment: June 9th 2022
End date of the experiment: June 13th 2022
Experiment goals:

- Check the reason of the Def useless parameter

Hypothesis: The previous bad results of the Definition feature is due to an underfitting of the Word Embedding model.

In order to check if the bad results of the Def features (in the first experiment) are due to the not high enough parameters for training the Word Embedding model, we will launch a run on all GUIs, with only Def feature enabled, with Word Embedding custom parameters.

Note: the experiment took a little bit of time to compute due to the large amount of data that the Deep Learning model had to take with the Word Embedding params.

Experiment parameters:

- Training set : 70% / Testing set : 30%.
- Maximum data per type : 8.000.
- Semantic Network part studied: All GUI.

- Word Embeddings params: vector size=1000, window=50, sequence length=700, epochs=60.
- Features compared: Def=Definition.

Results:

0.078 of precision, so 0.78%

The comments that we can make:

- The word embedding parameters are not responsible for the bad Def results. We invalidate our hypothesis.
- The possible reasons of the bad Def results are:
    - Bad preprocessing on the definitions corpus.
    - Bad manipulation of the Word Embedding results to the Deep Learning model.
    - The definition is not a source of information to class CUI in the Semantic Network (probably not true).

What we are going to do:

- Deeply check the code to verify the manipulation of the W.E. results.
- Debug preprocessing corpus.
- Upgrade the preprocessing corpus to keep the most important informations (regarding the STY classification).

Experiment 3

Following the not unsatisfying prediction results with the GUI trained model, we can globally validate the starting hypothesis.
We can make a transition to TUI experiments with its own level of granularity. Thanks to our rigorous methodology and the generic architecture, the transition is very easy and allows us to establish the first tries for TUI prediction that were significantly bad.

Start date of the experiment: June 13th 2022
End date of the experiment: June 16th 2022
Experiment goals:

- Identify the precision of the model with the terms and semantic type identifiers (TUI).

Hypothesis: The results of the prediction with all TUI are significantly bad in contrast to the GUI because of the number of classes.

After experimenting the model prediction with GUI classes, the interesting thing is to know how the model is going to behave with many more classes (127 TUI instead of 15 GUI).

Experiment parameters:

- Training set : 70% / Testing set : 30%.
- Maximum data per type : 500.
- Semantic Network part studied: TUI.
- Short type dataset: composed with “T204”, “T007”, “T061”, “T033”, “T109”, “T200”, “T002”, “T121”, “T004”, “T116”, “T047”, “T037”, “T201”, “T023”, “T028” GUI.
- Word Embeddings params: vector size=700, window=20, sequence length=100, epochs=40.
- Features compared: Labels / SAB=Sources / Parents=Parents type.
- We collected the f1-score for the diagram results.
- We removed the Definition feature because at the moment we launched the experiment, the problem was still remaining.

Note: the maximum data per type is low because of our calculator server. It can’t support beyond 500 * 127 * resources_per_class.

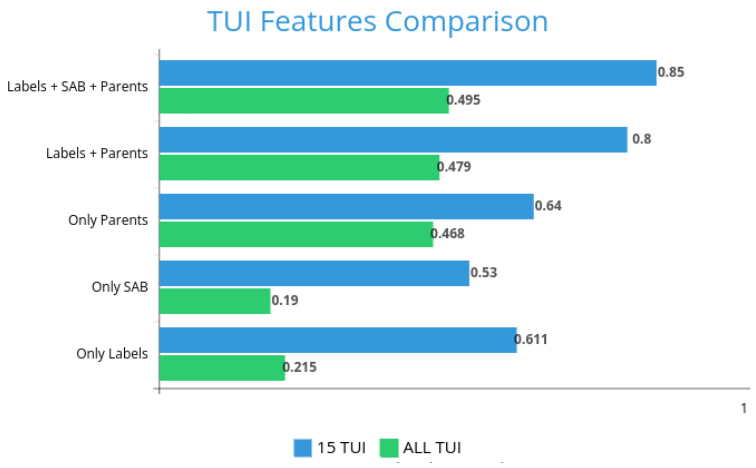


Figure 13. **TUI Features comparison**. The results are from 0 to 1.

The comments that we can make:

- We can see approximately the same results with the short dataset (15 TUI) as in experiment 1 with the large dataset (All GUI -> 15 GUI).
- The large dataset with all TUI (127) model predicts the correct value with 50% on average. Our starting hypothesis is invalidated.
- The Parents feature is the most important feature in our TUI prediction model.
- We trained the 15 most present TUI with only 500 concepts per type while even with the technical serveur ressources restriction we can increase the limit.

What we are going to do:

- Create a new experiment to look at the model prediction accuracy with the 30-40% most present TUI in the Semantic Network with a greater number of maximum data per class (this experiment will finish after the end of the writing of this paper).

Experiment 4

Start date of the experiment: June 15th 2022
End date of the experiment: June 21th 2022
Experiment goals:

- Identify the part of the correlation between the maximum data per class and the accuracy of the model with all GUI.

Hypothesis: The model is more accurate with more data. If we have a bigger calc server, the model would be more performant.

Experiment parameters:

- Training set : 70% / Testing set : 30%.
- Maximum data per type : from 5K to 9K.
- Semantic Network part studied: GUI.
- Word Embeddings params: vector size=700, window=20, sequence length=100, epochs=40.
- Features used: Def + Labels + Sources + Parents Types.
- We collected the f1-score for the diagram results.

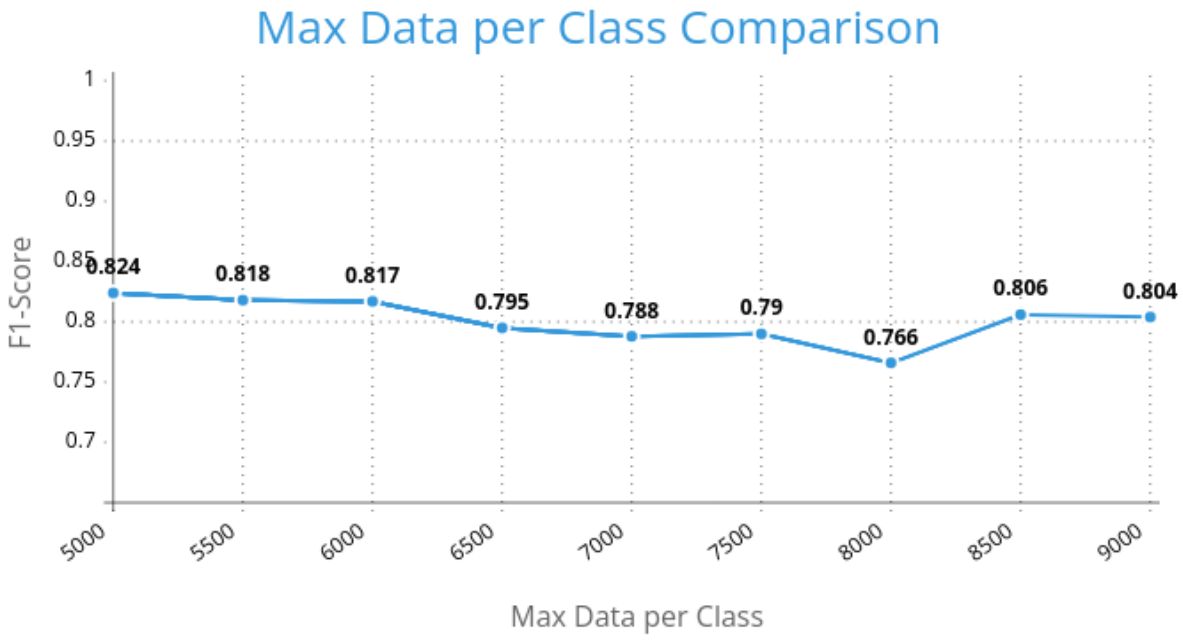


*Figure 14. Chart representing **Max Data per Class comparison**. The results are from 0 to 1.*

The comments that we can make:

- The experiment gives us irregular results.
- We know that increasing the maximum data per type can provoc multiple things (Experiments). But close to 9K of data per class still gives us 80% of good predictions.
- We invalidate our hypothesis. The quality of the data is more important than the quantity.
- The best accuracy is with 5K max data per class.

What we are going to do:

- For the future experiment, we will use 5K as the value for the “Max data per class” parameter.

Team Organisation

The subject of the project was written by Clement Jonquet (my tutor) in 2018. My internship allowed me to start this new project, i.e. on April 1st 2022. During the entirety of the project, Mr Jonquet was the project leader and guided me through the advanced experiments.

Mr. Jonquet gives a lot of autonomy and freedom to the development of the model until he gives the reins on the complete architecture of the pipeline. This can be a double-edged sword: you are free to apply such and such techniques as you want, but if the tools you have used are not adapted to the context of the project, you can lose considerable time. It is therefore important that there is a complete documentation phase on the technologies, the state-of-the-art techniques that exist to answer a problem.

Jonquet C. and me communicated through private channels like Slack or via mail because we are only two on the project. In addition to these discussions, we organized a meeting every week to expose the project avancement and to discuss the next goals bringing to the fore the problems encountered.

Conclusion

We worked on a classification method to assign terms to semantic types in UMLS. We developed a generic architecture and a model with tools that are popular and specific to our needs. The model configuration can be changed to let the model host as many features as you want. After several tests, we made a list enumerating the features that the model takes.

Through different experiments, we successfully established a model that can assign from the highest level of granularity (15 classes, with GUI) with a precision of 76%. Also, we experimented the training model from a lowest level of granularity (122 classes, with TUI) and we calculated its prediction around 50%.

These results allow us to validate our first hypothesis. The results are not perfect but quite good. The possibility to upgrade these results is vaste. Some points need further study like the calc server resources, the library, the list of features, the type of layers and more.

The next goals

As my internship continues even after I finish writing this paper (1 month), here are my and/or your future goals:

- Upgrading the definition feature working on these aspects:
    - Modify the Word Embedding libraries and complexity
    - Use another NLP technology to use CUI definitions as feature for our model
- Add visualization of the words proximity (Word Embedding) in a 2D or 3D space.
- Create a pipeline for production to prepare new graphs of data to be predicted by our model.
- Write a paper on our research results.

The direction that the “UMLS Tag Assignor” project will now depend on your next work. I hope that the project will continue in the same momentum to offer a solution of a quality and finish in perpetual evolution.

Glossaire

- **NLM**: National Library of Medecine (USA).
- **UMLS**: Unified Medical Language System, created and handled by the NLM.
- **Ontologies**: Set of terms specific to a concept, a science. Study of a concept, a science, vocabulary.
- **NLP**: Natural Language Processing, understanding and responding to text or voice data.
- **Metathesaurus**: It is the raw collections of concepts and terms from various controlled vocabularies and their relationships.
- **Semantic Network**: These are the overall categories that the information from the Metathesaurus is divided into, as a method of organization.
- **CUI**: A concept is a meaning. A meaning can have many different names. A key goal of Metathesaurus construction is to understand the intended meaning of each name in each source vocabulary and to link all the names from all of the source vocabularies that mean the same thing (the synonyms). CUI contains the letter C followed by seven numbers. For example: C0018681.
- **TUI, Terms and Semantic Type Identifiers**: Each Metathesaurus concept is assigned at least one semantic type along with its identifiers (TUI), that provide a consistent categorization of all concepts represented in the UMLS Metathesaurus. In all cases, the most specific semantic type available in the hierarchy is assigned to the concept. For example, the concept “Colonoscop” is categorized in ‘Diagnostic Procedure’ semantic type and assigned ‘T060’ identifier (TUI). There are overall 133 semantic types present in the UMLS.
- **GUI, Group Semantic**: The UMLS semantic network reduces the complexity of the Metathesaurus by grouping concepts according to the semantic types that have been assigned to them. For certain purposes, however, a set of semantic type groupings may be desirable. The following principles were used to design the groupings: semantic validity, parsimony, completeness, exclusivity, naturalness, and utility. For example, semantic types like ‘Diagnostic Procedure’,‘Laboratory Procedure’ and ‘Therapeutic or Preventive Procedure’ are grouped into ‘Procedure’.
- **Pipeline**: A pipeline is a collection of scripts or functions that transmit data in a series.
- **Word Embedding**: designates a set of machine learning techniques that aim to represent the words or sentences of a text by vectors of real numbers, described in a vector model.

2- Professional letter

From: Alexandre Juan
To: Mr Paradou
Subject: About the new project

Hello, Mr. Paradou,

I am aware of the company's new project and I would like to contact you to express my interest in joining its work team. I've had the opportunity to work for the company over the last ten years, and I'd like to continue working for the company. The new artificial intelligence project that the company wishes to launch would be the ideal project to rekindle my interest.

My point of view is that I am more than capable of doing the project as it requires specific experience in programming and computing, something I have had during the last few years. I think that I am the right person for this job, as I have been working on many projects around the same goals, and my application on the topic has been praised by all my partners and team leaders. I master the relevant languages that are going to be used for this work, and I know for sure that these skills will help me cope with the project requirements.

My positive approach and straightforward work method, I am sure, would be an asset to the company's project, as well as my capacity for high motivation and ability to handle even high stress levels without losing focus. Working on it will not be a problem for me as I enjoy challenges and see obstacles as part of my life, which is something I have had to learn over the years

I hope to be given the opportunity to speak to you personally in order to clarify any questions and concerns that you might have about my professional capabilities.

Sincere greetings

Alexandre Juan