Rapport de Stage M2 TAL (2020-2021): Analyse Sémantique

Entreprise:

SNCF Réseau
10 rue Camille Moke
93200 Saint-Denis

Etudiante:

Amer Yara

Sommaire:

Introduction:

J'effectue depuis le 29/10/20, une alternance au sein du Smart Studio de SNCF Réseau, en tant qu'analyste sémantique. Cette alternance dure jusqu'au 13/09/21, soit environ 11 mois. Elle s'inscrit dans mes études en Traitement Automatique des Langues, et s'effectue en parallèle de mon Master 2 à l'université de Paris - Nanterre.

SNCF Réseau, filiale de la SNCF, est une entreprise publique chargée de la gestion du réseau ferroviaire français. Elle possède un organisme appelé "Smart Studio", spécialisé dans l'intelligence artificielle et l'analyse de données. Mes missions en tant qu'analyste sémantique consistent notamment en l'analyse d'enquêtes de satisfaction ainsi que des rapports d'incidents, à la fois grâce à l'utilisation de logiciels dédiés et grâce au développement informatique. Toutes les missions effectuées se sont faites en binôme, avec Martin Salud, également alternant en analyse sémantique et étudiant en M2 TAL à l'université de Paris - Nanterre.

L'analyse sémantique se donne pour but de comprendre les mots dans leur contexte, donc d'en faire sens. Elle cherche ainsi à tirer la signification d'un corpus, en analysant ses éléments textuels. Il s'agit aujourd'hui de l'un des domaines majeurs du TAL dans le cadre de l'entreprise et possède diverses applications (traduction, analyse d'opinions, analyse prédictive, agents conversationnels etc…). L'analyse sémantique fait usage à la fois d'une approche linguistique mais aussi statistique à travers le machine learning.

Ce rapport de stage vise à explorer les applications concrètes de l'analyse sémantique dans un cadre professionnel, notamment ici dans le cadre d'une entreprise ferroviaire, donc possédant des besoins et un vocabulaire qui lui sont propres. Nous verrons donc les problématiques, les outils et les méthodologies propres à l'analyse sémantique au sein de SNCF Réseau.

A-) L’entreprise:

1-) La SNCF:

La SNCF (Société Nationale des Chemins de Fers Français) est l’entreprise ferroviaire publique française, officiellement créée en 1937. Depuis janvier 2020, il s’agit d’une société anonyme à capitaux publics. Elle est constituée d’une société mère assurant la direction de ses cinq filiales: SNCF Réseau, SNCF Voyageurs, Rail Logistics Europe, Geodis et Keolis.

2-) SNCF Réseau:

SNCF Réseau, filiale de la SNCF, est le principal gestionnaire ferroviaire français, l’entreprise est chargée de l'ingénierie et de la maintenance du réseau ferré, ainsi que de la construction de nouvelles lignes ferroviaires et de la gestion de la circulation des trains. SNCF Réseau compte environ 52 000 employés en 2015, soit un tiers des effectifs de la SNCF.

3-) Le Smart Studio:

Le Smart Studio est un pôle dédié à l’intelligence artificielle au sein de SNCF Réseau. Il accompagne les différents enjeux du métier ferroviaire, en proposant des solutions technologiques. Avec l’Immersive Studio, qui est quant à lui dédié à la réalité virtuelle et augmentée, le Smart Studio fait partie du pôle Next Techs.

Il propose l’offre Smart Langage, qui fait usage du TAL pour l’analyse de différents supports tels que les enquêtes de satisfaction ou des rapports d’incident, mais aussi afin de développer des technologies de synthèse vocale ou bien faciliter la communication entre conducteurs de différents pays.

L’offre Smart Maintenance vise à effectuer de la maintenance prédictive, en détectant des comportements anormaux, en anticipant les perturbations, etc… Pour cela, cette offre fait usage de nouvelles technologies telles que les data sciences ou le blockchain.

L’équipe du Smart Studio est composée de Kamel Bentchikou, responsable du pôle Next Techs, Enora Goulard, son adjointe, ainsi que de Martin Salud et moi-même, alternants en analyse sémantique.

B-) Le cadre du stage:

1-) L'analyse sémantique:

Les activités de ce stage s'inscrivent dans le domaine de l'analyse sémantique. L'analyse sémantique permet d'établir la signification d'un texte, en analysant ses éléments textuels. Elle fait donc sens des mots dans un contexte donné. Elle s'effectue grâce à une analyse lexicale (au niveau des mots) et au niveau syntaxique (juxtaposition des mots entre eux). Elle permet une compréhension plus approfondie d'un corpus, comme ici les rapports d'incidents. Elle peut s'allier et compléter d'autres méthodes telles que le machine learning, pour ainsi créer des outils de prédiction comme pour l'outil développé dans le cadre du projet Trinôme Panto, qui sera explicité plus en détails dans le cadre de ce rapport de stage.

Dans le cadre d'une analyse d'opinions, comme celle effectuée lors de ce stage pour le projet DGNUM, elle prend en compte les verbatims, donc les termes positivement ou négativement connotés. Ces derniers peuvent être sollicités (dans le cadre d'un questionnaire de satisfaction) ou bien spontanés. Elle permet donc de mieux évaluer le besoin client et de répondre plus convenablement à ce dernier.

Afin de répondre à l'offre d'analyse sémantique, dans le cadre de l'entreprise, des outils permettant une analyse sémantique semi-automatisée existent sur le marché, comme par exemple Proxem Studio.

2-) Les outils:

a -) Proxem Studio:

Proxem Studio est un logiciel d'analyse sémantique destiné à un cadre d'entreprise. Il permet la collecte et l'analyse de données textuelles (Text Mining), dans une perspective de compréhension du langage naturel (Natural Language Understanding). Je vais présenter les fonctionnalités dont nous avons principalement fait usage durant cette alternance.

Proxem Studio possède une fonctionnalité "Connect Files" permettant d'importer des fichiers à analyser, sous format excel ou PDF. Les informations comprises dans les colonnes doivent être désignées par l'utilisateur en tant que texte ou bien en tant que métadonnées, afin d'être traitées par le logiciel.

Proxem Studio permet l'annotation des documents, à travers des requêtes rédigées dans une grammaire locale, donc une grammaire propre au logiciel. Cette dernière s'apparente aux expressions régulières.

Mémo

Requête	Description
`.`	Recherche n'importe quel mot **NB :** ne peut pas être utilisé en début ou fin de grammaire (ie pas à côté d'une accolade), seulement entouré par d'autres éléments
`?`	Rend un mot ou groupe de mots optionnel
`+`	Recherche un mot ou groupe de mots apparaissant une ou plusieurs fois
`*`	Recherche un mot ou groupe de mots apparaissant zéro, une ou plusieurs fois

`.+`
`.*`
`.?`
	Recherche :
	• n'importe quel groupe de mots
	• idem, mais de manière optionnelle
	• un mot optionnel

`{2}`
`{2,4}`
`{2,}`
	Recherche un mot ou groupe de mots apparaissant
	• deux fois
	• entre deux et quatre fois
	• au moins deux fois

Figure 1 - Grammaire de Proxem selon la documentation officielle

L'utilisateur identifie et nomme des concepts, auxquels il associe des règles ou bien requêtes. Ces concepts peuvent être groupés, et rangés en sous-groupes.

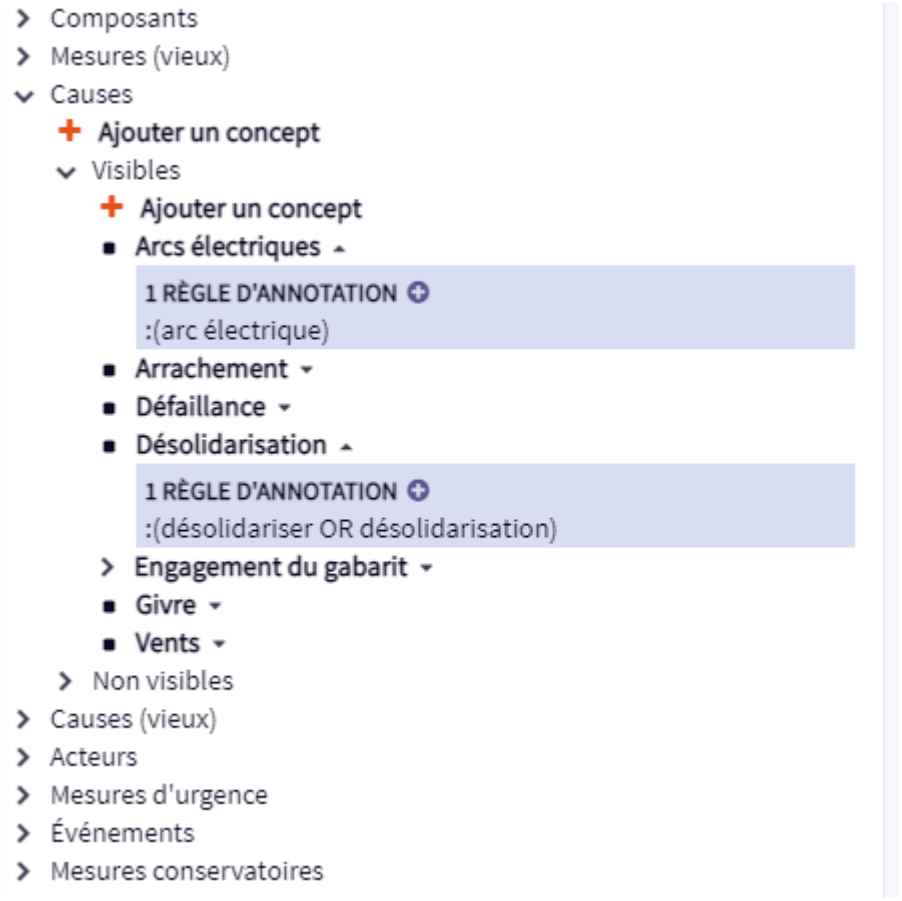


Figure 2 - Première tentative de catégories, concepts et requêtes pour le projet Trinôme Panto, sur Proxem Studio

Une fois ces concepts construits, Proxem Studio permet l'exploration et la visualisation des données. Il est possible de créer un tableau de bord, afin de visualiser les informations qui nous intéressent. On peut générer un graphique sous diverses formes (diagramme en bâtons, diagramme circulaire, diagramme linéaire, TreeMap etc…) où l'on peut croiser les différentes catégories de concepts ainsi que les métadonnées, pour comprendre leur interaction et leur corrélation statistique.

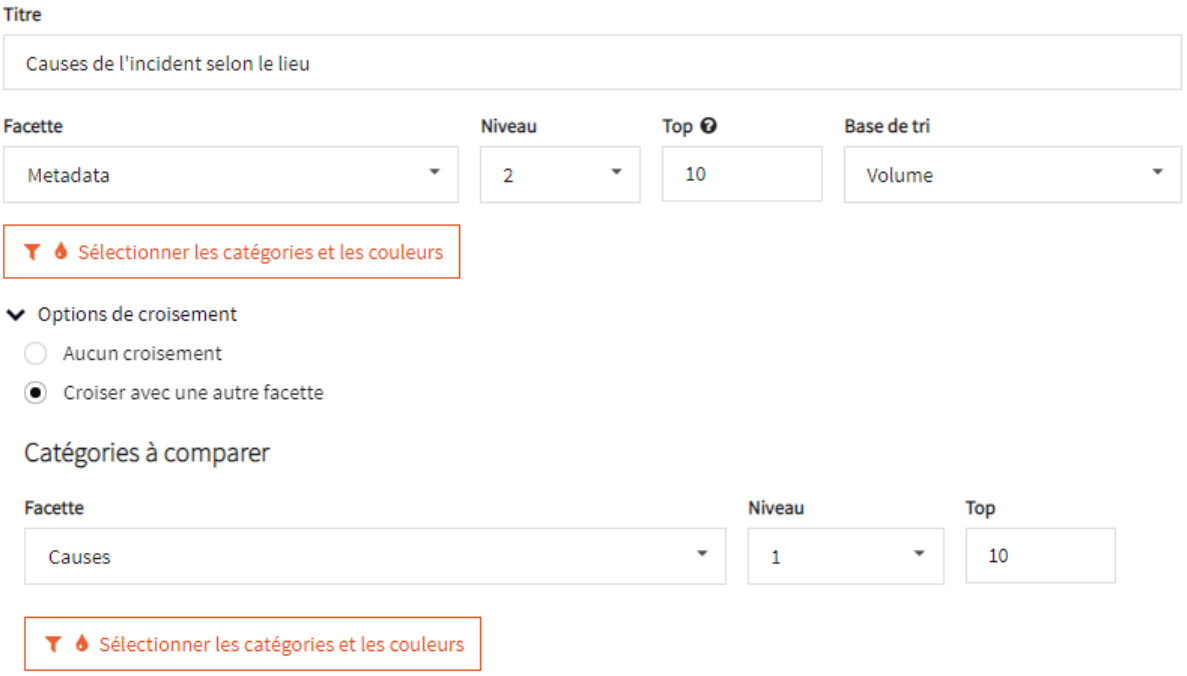


Figure 3 - Interface de la fonctionnalité "Explore", permettant de générer des graphes

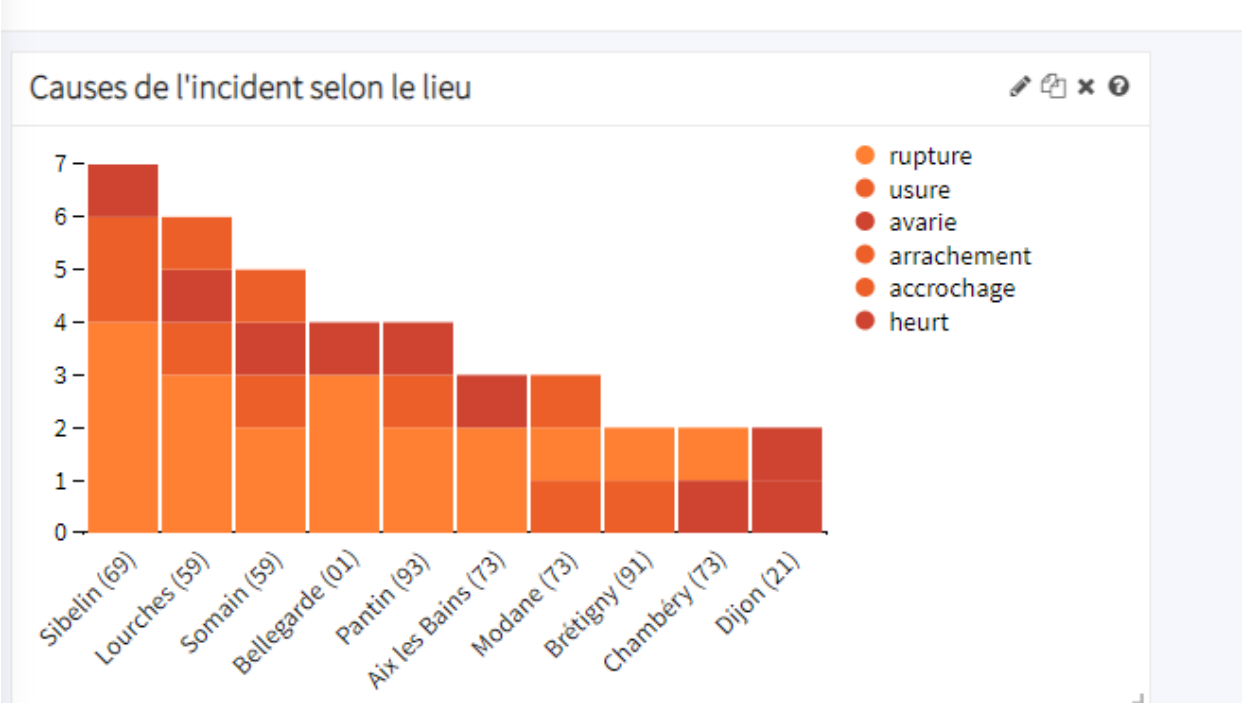


Figure 4 - Diagrammes montrant le causes de l'incident selon le lieu (département), issu du projet Trinôme Panto

Les diagrammes générés permettent une visualisation claire et accessible, que l'on peut donc présenter par la suite.

b-) Les bibliothèques Python:

Dans le cadre de cette alternance, nous avons également développé nos propres outils, grâce au langage Python. Nous avons notamment programmé grâce à l'IDE IDLE ou bien Google Colab. Le Python présente l'avantage de nous être familier, grâce à nos études en TAL, mais aussi celui d'inclure de nombreuses bibliothèques d'analyse linguistique et de machine learning.

Par exemple, nous avons utilisé la bibliothèque Stanza pour la lemmatisation et la tokénisation. Stanza est une bibliothèque offrant des fonctionnalités d'analyse linguistique dans une diversité de langues (par exemple le POS tagging, l'analyse des dépendances, la reconnaissance d'entités nommées etc…).

Pour la manipulation des fichiers CSV, nous avons utilisé Pandas. Afin d'effectuer des tâches de machine learning, nous avons utilisé les fonctionnalités et les modèles de classification multilabel offerts par Scikit Learn. Scikit Learn est une bibliothèque libre, destinée à l'apprentissage automatique, proposant de nombreuses bibliothèques d'algorithmes de classification. Afin de déployer nos scripts, nous avons utilisé la bibliothèque Flask.Il s'agit d'un framework permettant de développer des applications en Python, sans que l'utilisateur ne doive se soucier des opérations de bas niveau tels que le protocole informatique. La bibliothèque comprend une API permettant la communication entre le script Python et un serveur.

c-) Coggle:

Coggle est une application web gratuite, offrant des outils de mindmapping. Elle permet de produire des documents structurés de façon hiérarchique, sous forme d'arborescence. On a également l'option d'ajouter des icônes (par exemple pour dénoter l'importance d'une tâche) ou bien des images. Coggle permet une édition collaborative, et nous a donc permis d'établir des ontologies avec par exemple, des termes reliés à l'incidentologie ferroviaire, mais aussi d'organiser nos tâches de travail afin d'établir une feuille de route.

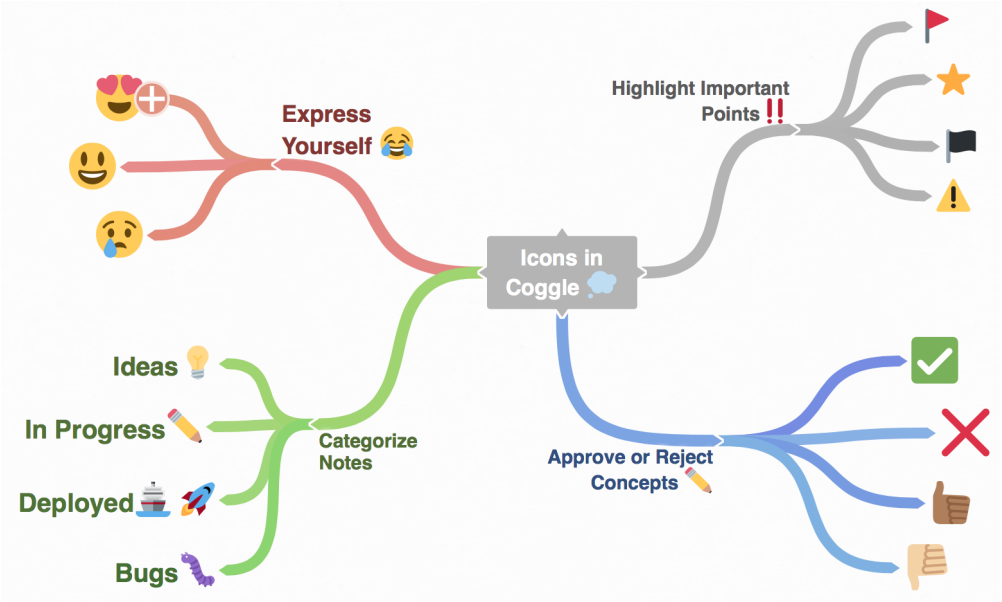


Figure 5 - Exemple d'arborescence type, publié sur le site de Coggle

d-) Microsoft Teams:

Microsoft Teams est une application de communication collaborative. Le service intègre la suite Microsoft Office 365 ainsi que d'autres extensions. Au-delà des fonctionnalités de communication offertes par cette application, nous avons souvent utilisé le SharePoint, afin de partager des documents, ainsi que le planner afin d'organiser nos tâches.

C-) Les activités du stage:

1- Projet DGNUM:

L'enquête DGNUM, menée par le service DGNUM (Direction Générale du Numérique), est une enquête de satisfaction annuelle à destination des agents, concernant les applications utilisées au sein de la SNCF. Le but d'une analyse sémantique semi-automatisée à l'aide du logiciel Proxem Studio est de pouvoir en extraire des verbatims qui sont difficilement analysables et exploitables manuellement. Il s'agit donc d'une analyse d'opinions effectuée à grande échelle. L'analyse de ces verbatims vise à faire remonter les grandes thématiques, les axes d'améliorations, les services concernés, etc...

	A6	APPLI_ORDR	APPLICATION	B2_B9	B6_B12	B7_B13	C11	C1
ə	Cadre	APPLI 1						
	Agent de maî	APPLI 1	DURANDAL2	Plutôt satisfait(e)				
	Agent d'exéc	APPLI 1						
	Cadre	APPLI 1	CAPITOLE	Plutôt pas satisfait(e)		l'ancienneté de la GEP CAPITOLE		
	Agent de maî	APPLI 1	SPOT MOBILE	Plutôt pas satisfait(e)	Rien	Le temps de synchronisation		
	Cadre	APPLI 1	CADA	Plutôt satisfait(e)				
ə	Cadre	APPLI 1	GAIASCOPE	Tout à fait satisfait(e)	Son utilité pr	Quelques foi	Très chaleureux a	Pa
	Agent d'exéc	APPLI 1	E-HOUAT	Plutôt satisfait(e)	La quantité d	Interface trop vieille et pas assez s		
ə		APPLI 1						
ə	Agent de maî	APPLI 1						
	Cadre	APPLI 1	SEISM	Plutôt satisfait(e)	Permet de ce	N'est pas tou	disponibilité, effic	tr
	Cadre	APPLI 1	WIFI1	Plutôt pas satisfait(e)	Le format ex	Absence de l	Rapidité, efficacité e	
ə		APPLI 1	PAPO	Plutôt satisfait(e)	La stabilité, la performanc		Disponibilité, écoute	
	Agent de maî	APPLI 1	TSP1	Plutôt pas satisfait(e)				
	Agent d'exéc	APPLI 1	ODICEO	Plutôt satisfait(e)				
	Cadre	APPLI 1	GEOPRISM	Plutôt satisfait(e)				
ə	Cadre	APPLI 1	GESICO	Tout à fait satisfait(e)	La bascule so	Les déconnexions intempestives q		
		APPLI 1	CADA	Tout à fait satisfait(e)				
	Agent de maî	APPLI 1						
	Agent d'exéc	APPLI 1	SPOT MOBILE	Plutôt pas satisfait(e)	La simplicité	Les mises à jours trop longue		
	Agent de maî	APPLI 1	CONTACT	Plutôt satisfait(e)				
	Cadre	APPLI 1	DEFRAIL 2	Plutôt satisfait(e)	fluidité	redondance	-	-
ə	Cadre	APPLI 1						
ə	Cadre	APPLI 1	SEISM	Tout à fait satisfait(e)	le fait de pou	la gestion de	etre facilement jo	le
	Cadre	APPLI 1	DURANDAL2	Plutôt satisfait(e)		présentation qui n'est pas sous fo		
ə	Cadre	APPLI 1						
	Agent de maî	APPLI 1	CONTACT	Plutôt satisfait(e)				
	Agent d'exéc	APPLI 1						
	Agent d'exéc	APPLI 1	OTARIE	Plutôt satisfait(e)	l'accès intuitif à certaines informations.			
	Agent d'exéc	APPLI 1						
	Agent d'exéc	APPLI 1						
ə	Cadre	APPLI 1						

Figure 6 - Échantillon du fichier CSV comprenant l'enquête de satisfaction DGNUM

Grâce à la documentation fournie, notamment “Le Baromètre de Satisfaction Interne”, nous avons pu établir une ontologie.

PROG : E1/E2 sur le même écran
A TOUS
E1. Savez-vous qui sont vos Interlocuteurs de proximité en matière de Système d'Information ?

1	Oui
2	Non

SI E1=1
E2. Parmi ces interlocuteurs de proximité SI, lesquels connaissez-vous particulièrement ?
Plusieurs réponses possibles

1	Relais Régionaux du Système d'Information (RSI)	
2	Correspondants en Système d'Information d'Entité (CSIE)	
3	Gestionnaires de flotte mobile	
4	Autres correspondants	***Ouvert*** *Merci de préciser*

Figure 7 - Échantillon du Baromètre de Satisfaction Interne, sur les interlocuteurs de proximité et SI

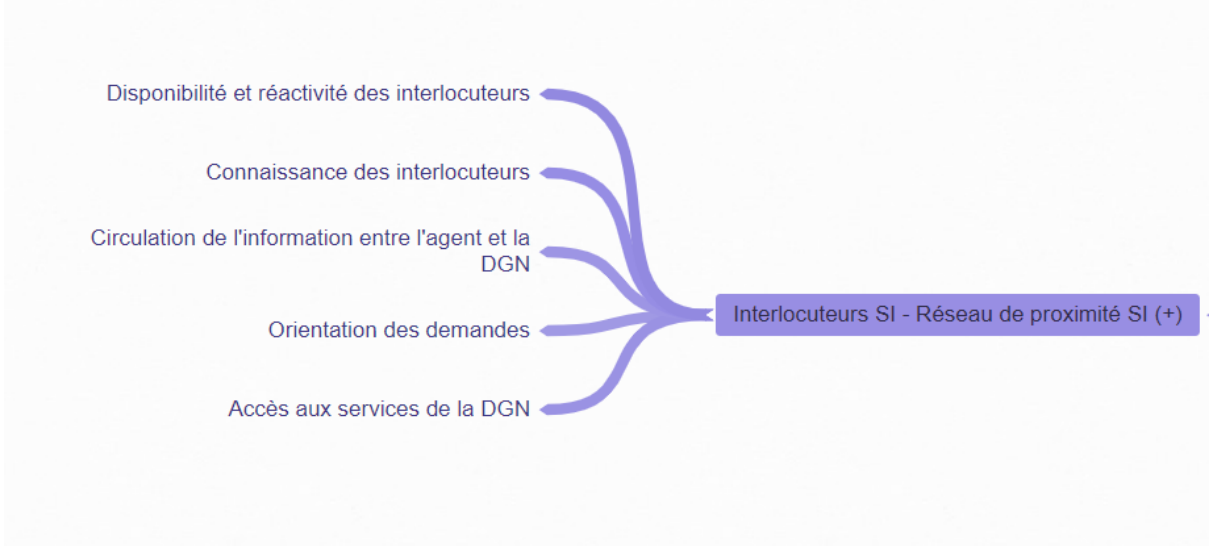


Figure 8 - Echantillon de l'ontologie de l'enquête DGNUM, montrant les Interlocuteurs de proximité et SI

Cette ontologie comprend les différentes catégories du questionnaire de satisfaction. Une colonne dans le csv correspond à une catégorie dans l'ontologie et à une catégorie de concepts sur Proxem (par exemple, ici interlocuteurs SI - réseau de proximité SI). Les branches dérivées d'une catégorie donnée dans l'ontologie correspondent à son contenu, et comprennent donc les verbatims qui nous intéressent.

On transpose les verbatims de l'ontologie sous forme de concepts grâce à la fonctionnalité "Annotate" de Proxem Studio.

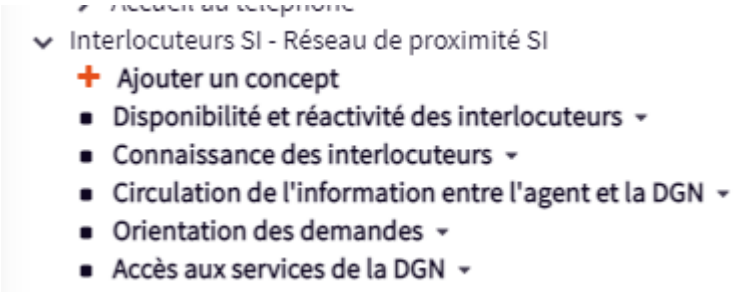


Figure 9 - Echantillon des catégories de concepts sur Proxem Studio

Grâce à la syntaxe de Proxem, nous formulons des requêtes afin d'identifier les verbatims dans le document:

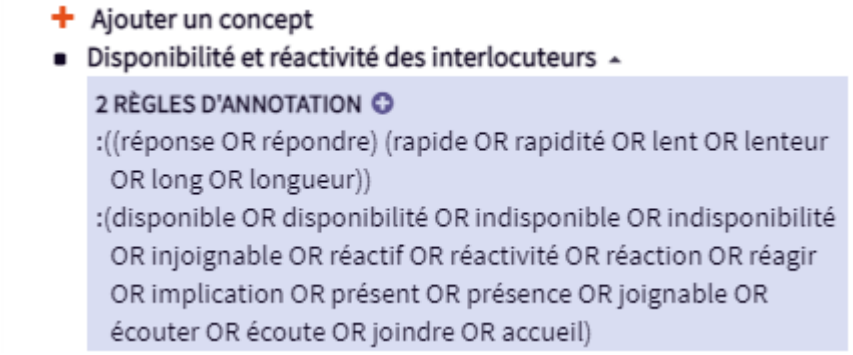


Figure 10 - Echantillon des requêtes sur Proxem Studio

Enfin, grâce à la fonctionnalité "Explore", nous pouvons visualiser les données sous forme de graphes. Grâce au questionnaire, nous pouvons également déterminer la connotation des verbatims (par exemple, pour la question " connaissez-vous vos interlocuteurs de proximité ?" le fait de les connaître est positivement connoté). En créant des graphes, nous avons la possibilité de croiser des catégories, et de peupler une catégorie donnée avec les verbatims désirés.

Ce diagramme en bâtons montre les verbatims positifs concernant la performance des applications utilisées par le métier. On peut y voir que l'ergonomie est le plus souvent mentionnée positivement.

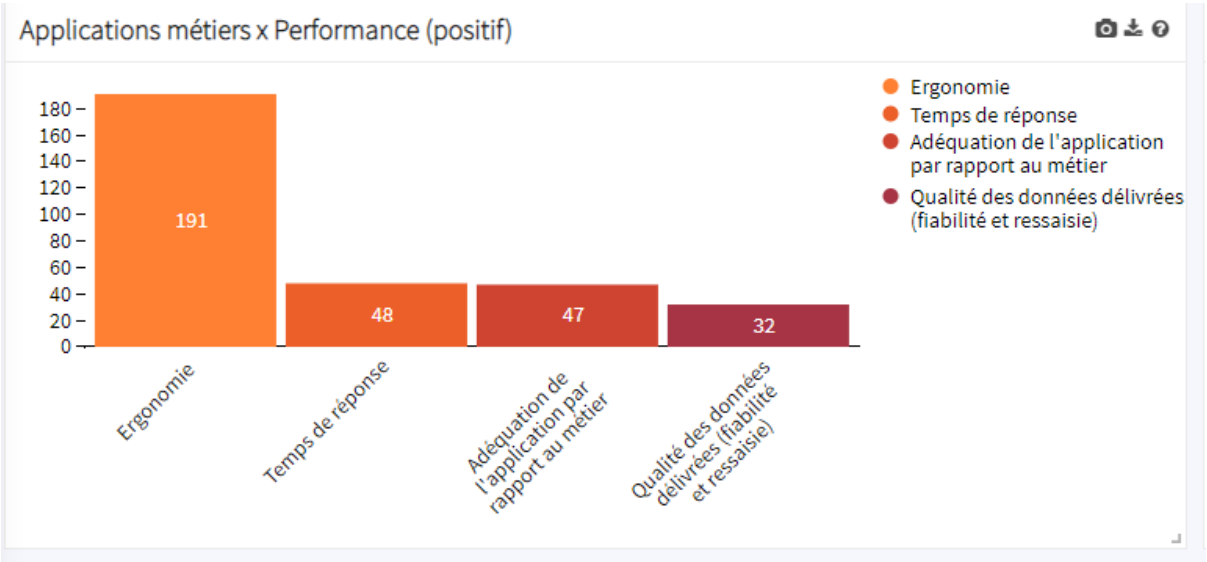


Figure 11 - Diagramme en bâtons montrant la performance positive des applications

Ce diagramme montre, par contraste, les verbatims négatifs relatifs à la performance des applications. On voit qu'ici encore, l'ergonomie ressort le plus, mais c'est probablement dû à un nombre d'occurrences plus élevé de verbatims relatifs à l'ergonomie, d'où la nécessité de comparer les deux diagrammes. En revanche, là où le temps de réponse apparaissait très peu de façon positive, il apparaît de façon marquée quand il s'agit de verbatims négatifs. On peut en déduire que le temps de réponse est un point faible des applications.

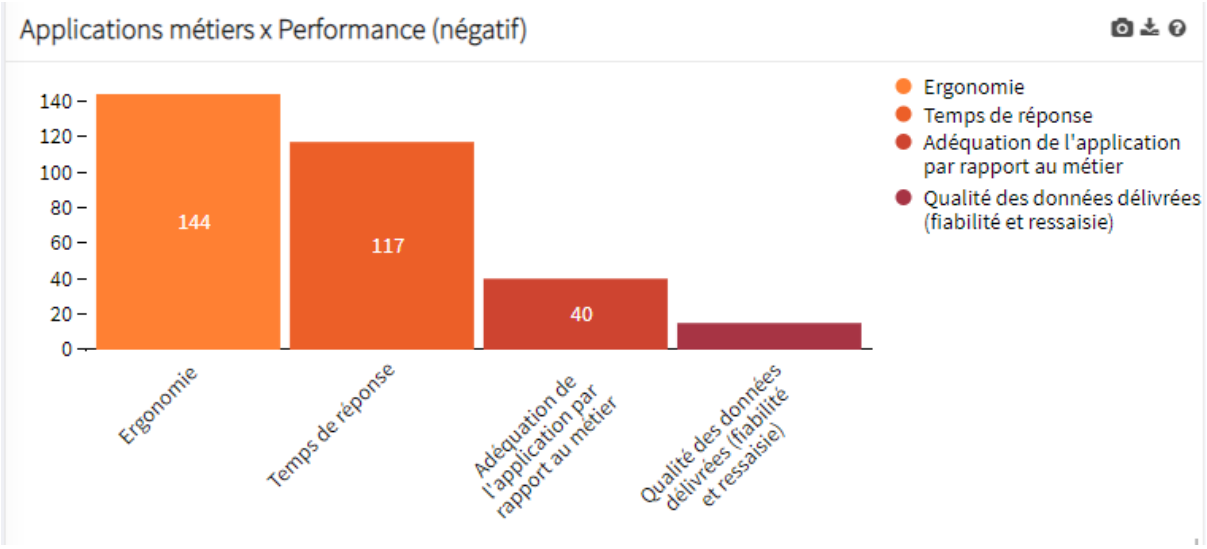


Figure 12 - Diagramme en bâtons montrant la performance négative des applications

Dans ce diagramme, on se focalise sur les verbatims positifs relatifs à l'accueil téléphonique du support de l'application ASU. On peut voir que le point positif qui ressort le plus, est la disponibilité d'un interlocuteur.

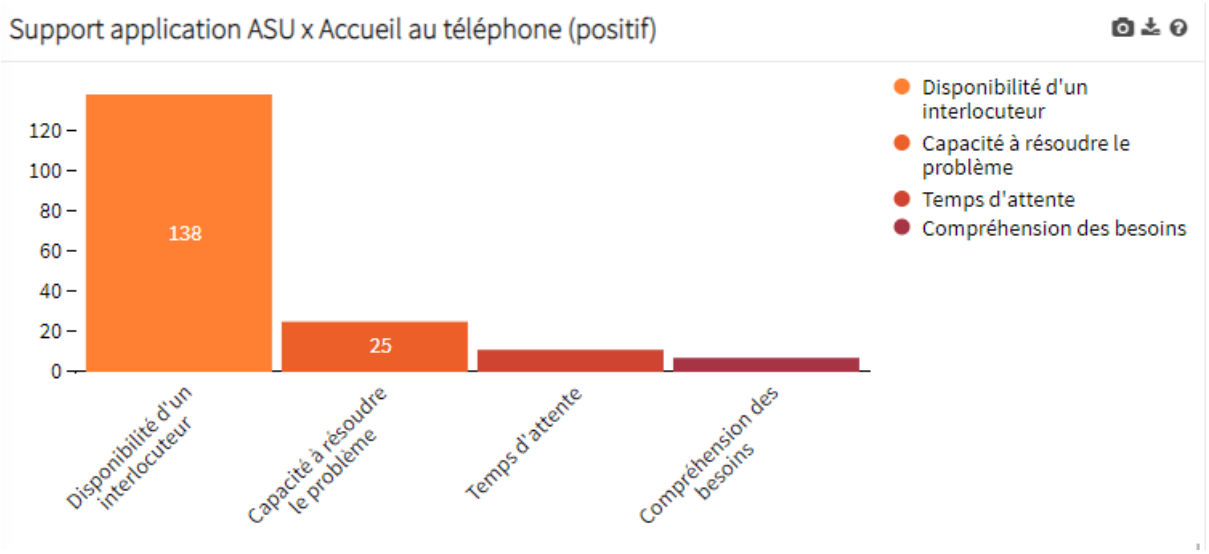


Figure 13 - Diagramme en bâtons montrant la performance positive du support application ASU, au niveau de l'accueil au téléphone

Ici, l'on se focalise sur les verbatims négatifs. La capacité à résoudre un problème, ressort comme l'aspect négatif majeur.

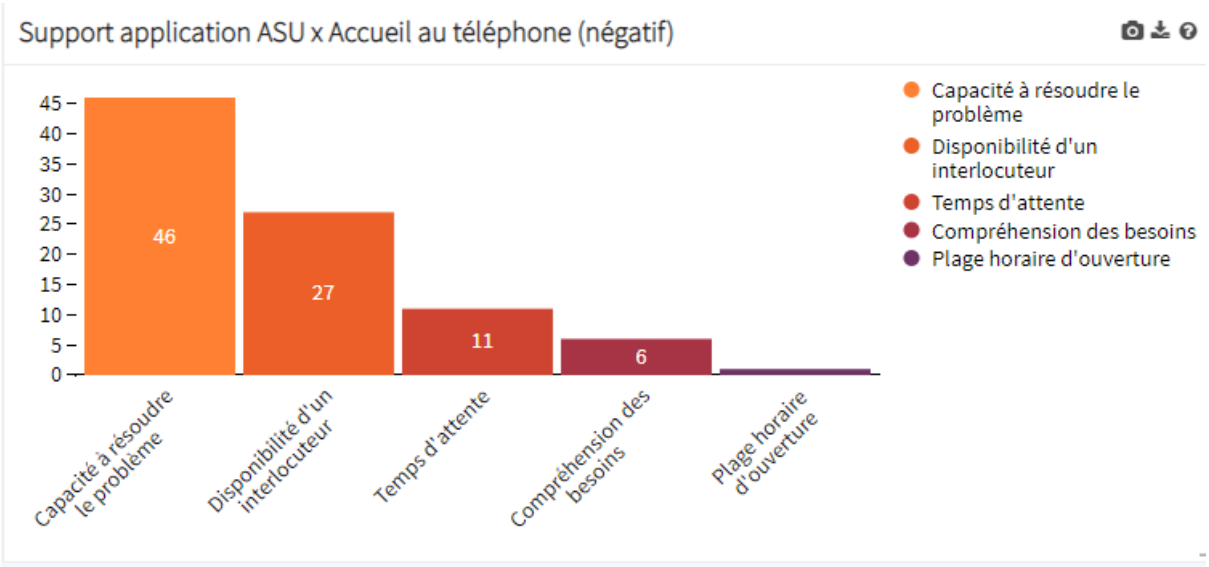


Figure 14 - Diagramme en bâtons montrant la performance négative du support application ASU, au niveau de l'accueil au téléphone

2-) Projet Trinôme Panto:

Ce projet concerne l'analyse des rapports d'incidentologie caténaire. La caténaire est un ensemble de câbles et de fils conducteurs destinés à alimenter les moyens de transports électriques. Les rapports sont rédigés sous format PDF, et il existe deux types de rapports, les fichiers FI (fiches d'incidents) et les fichiers RAC (rapports d'analyse causale). Ces rapports comprennent le descriptif de l'incident (lieu, date,

heure, composition de l'engin moteur etc…) ainsi que différentes catégories permettant d'évaluer l'incident (contexte, causes, mesures prises, risques etc…).

RAPPORT D'ANALYSE CAUSALE n°33309

Nature de l'événement : Incident caténaire, sans affaissement du fil de contact, mais engagement du gabarit, suite à pendule défaillant découvert par ADC. Sans conséquence.

1. DESCRIPTIF

Date :	27/04/21	***Heure :***	18 :15	***Lieu :***	Perrigny Poste 2 (21)
N° de ligne :	860000	***Voie :***	VP	***Régime d'exploitation :***	DV double voie
Circulation :	733470	***Activité :***	Voyageurs,	***Exploitant :***	SNCF Voyageurs,
Sous-traitant :	Non concerné,	***Composition :***	Engin moteur : Z51621, Parcours : Seurre - Dijon Ville.		
Particularités des lieux :	La ligne concernée par l'évènement est équipée d'installations fixes de traction électrique alimentées en courant continu : 1500 volts.				

2. CIRCONSTANCES

Contexte :

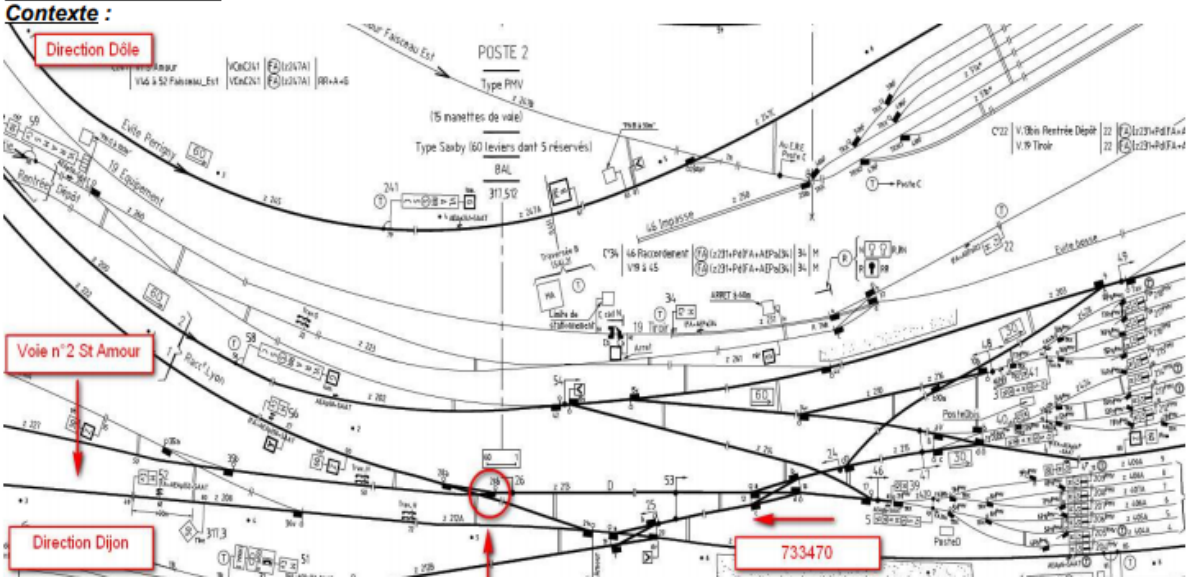


Figure 15 - Échantillon d'un fichier RAC

Nous avons effectué une première analyse sémantique avec le logiciel Proxem. Le but initial était d'obtenir des statistiques et des patterns, puis de pouvoir les visualiser.

Le fait de traiter l'ensemble de ces données manuellement est chronophage et coûteux, de plus le savoir technique est à pérenniser car les agents peuvent par exemple partir à la retraite. Notre étude sémantique rend ces corpus accessibles, et elle est plus rapide et économique.

a -) Analyse sémantique avec Proxem:

Afin de pouvoir traiter l'ensemble des fichiers avec Proxem, nous avons décidé d'établir des fichiers CSV comprenant l'ensemble des rapports, où une colonne

correspondrait à une catégorie. Nous avons généré un fichier CSV pour les rapports FI et pour les rapports RAC, respectivement.

```
desc = re.findall(r"Description de l'évènement.*Information immédiate", text)
for b in desc:
    desc = b.replace("Description de l'évènement :","").replace("Information immédiate","")
    dict_desc[num] = desc

action = re.findall(r"exploitants :.* Mesures conservatoires", text)
for b in action:
    action = b.replace("exploitants :","").replace(" Mesures conservatoires","")
    dict_action[num] = action

mesures_gi = re.findall(r"par le GI :.* Autres commentaires ", text)
for b in mesures_gi:
    mesures_gi = b.replace("par le GI :","").replace(" Autres commentaires ","")
    dict_mesuresgi[num] = mesures_gi

commentaire = re.findall(r"Autres commentaires.*Mesures conservatoires :", text)
for b in commentaire:
    commentaire = b.replace("Autres commentaires / mesures autres mises en oeuvre :","").replace("
    dict_commentaire[num] = commentaire
```

Figure 16 - Échantillon du script Python, permettant de scraper

Les fichiers ont été scrapés à l'aide de scripts Python, où l'on utilise notamment des expressions régulières afin d'identifier et de délimiter les catégories qui nous intéressent. Nous avons décidé de regrouper les lieux par région administrative, afin de pouvoir obtenir des statistiques plus pertinentes.

	C	D	E	F	G
1	Lieu	égion Administrat	Circonstances	Conséquences	C
2	Gien (45)	Centre-Val-de-Loi	Contexte : Le train FRET 75049	GravitÃ© des domma	
3	Javel (75)	Ile-de-France	Contexte : Les installations de s	GravitÃ© des domma	
4	Entre BÃ©zier	Occitanie	Le 09112011 vers 23h05 le train	GravitÃ© des domma	
5	Ecole Valentin	Bourgogne-Franc	Contexte : Pendant le mouveme	GravitÃ© des domma	
6	Arlac (33)	Nouvelle-Aquitain	A 19h28, le conducteur du TER 8	GravitÃ© des domma	
7	NÃ®mes (30)	Occitanie	Contexte : Le 28 mars 2017, une	GravitÃ© des domma	
8	Juvisy (91)	Ile-de-France	Contexte : Le train nÃ°7214223	GravitÃ© des domma	
9	Bailleul-Sir-Be	Hauts-de-France	Contexte : Le conducteur du trai	GravitÃ© des domma	
10	Libourne (33)	Nouvelle-Aquitain	Contexte : Des travaux sont en	GravitÃ© des domma	
11	Entre Magenta	Ile-de-France	Contexte : Section de ligne de do	GravitÃ© des domma	
12	Entre Gravigny	Ile-de-France	Contexte : Le train autotrain 245	GravitÃ© des domma	
13	Lourches (59)	Hauts-de-France	Le train SPAC 20 assure une mis	GravitÃ© des domma	
14	Avignon (84)	Nouvelle-Aquitain	Dans la matinÃ©e du mercredi 1	GravitÃ© des domma	
15	Ruffec (16)	Hauts-de-France	Contexte : Le train nÃ° 865646 d	GravitÃ© des domma	
16	Orrouy Glaigne	Nouvelle-Aquitain	Contexte : Le train de IEF SNCF	GravitÃ© des domma	
17	Entre Pauillac	Ile-de-France	Contexte : Le Train 141639 parti	GravitÃ© des domma	
18	Villeneuve Tria	Auvergne-RhÃ´ne	Contexte : Le train nÃ°712156 9	GravitÃ© des domma	

Figure 17 - Échantillon du fichier CSV de sortie, après avoir scrapé les rapports d'incident

Nous avons été confrontés à un premier problème concernant l'analyse sémantique, celui de la terminologie technique utilisée dans les rapports.
A l'aide de la documentation fournie (par exemple: le référentiel d'infrastructure, le lexique ferroviaire, la formation des Responsables Locaux Sécurité, etc…), nous avons commencé à établir une ontologie des termes sous forme de carte mentale, que nous avons étoffée au fur et à mesure, à l'aide de l'outil Coggle. La documentation nous a permis d'identifier les termes techniques et de pouvoir établir des catégories relatives à la réalité du métier. Par exemple, les conséquences peuvent être divisées en trois catégories: conséquences pantographe, conséquences caténaires et conséquences relatives aux personnes. Les mesures, quant à elles, peuvent être conservatoires ou d'urgence. Les mesures d'urgence consistent par exemple en l'alerte des secours, la coupure, la couverture d'obstacle etc...

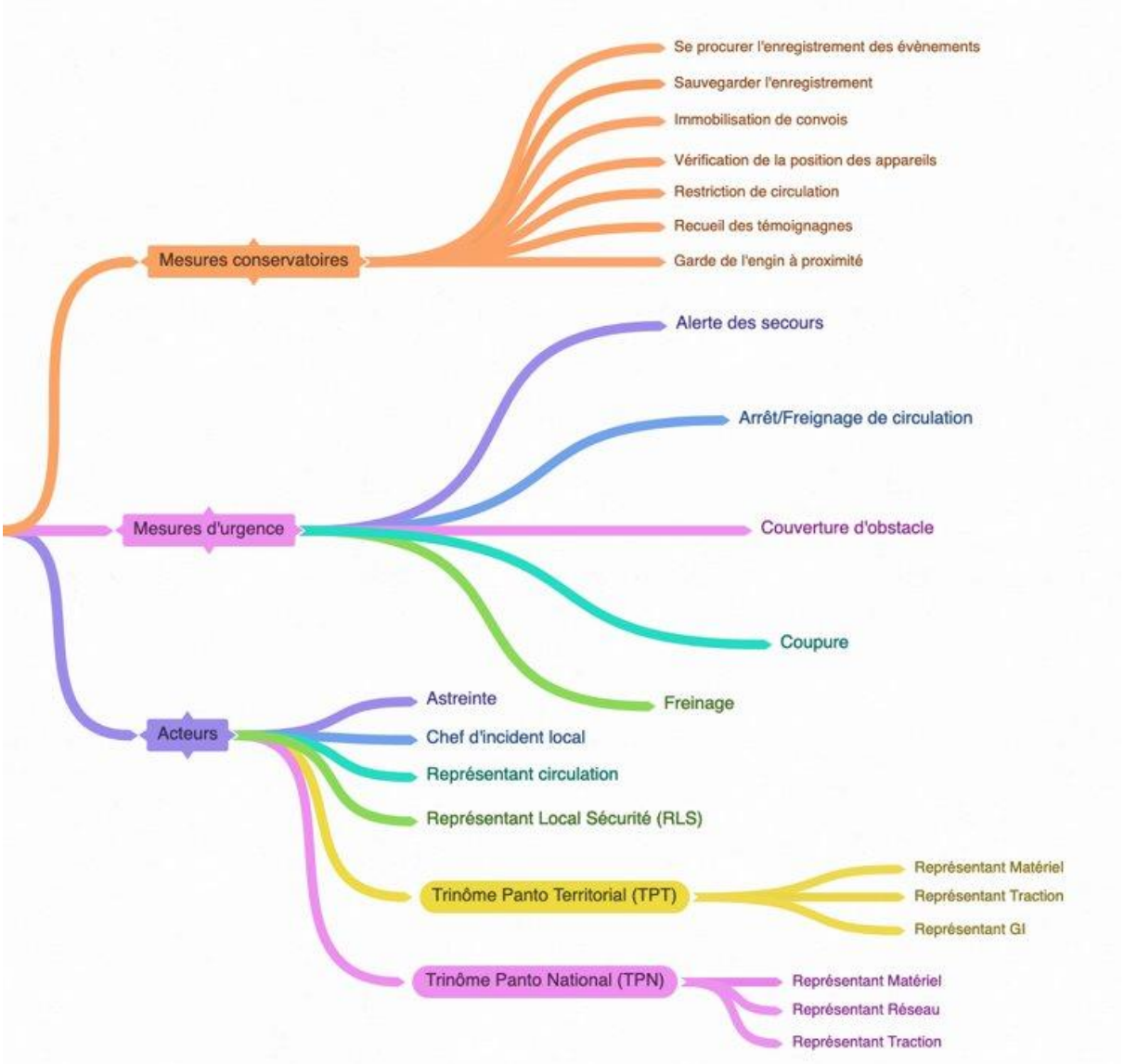


Figure 18 - Echantillon de l'ontologie sur l'incidentologie caténaire

Pour la catégorie “causes”, nous avons par exemple: végétation, faune, armement caténaire, engagement du gabarit, arc électrique etc...

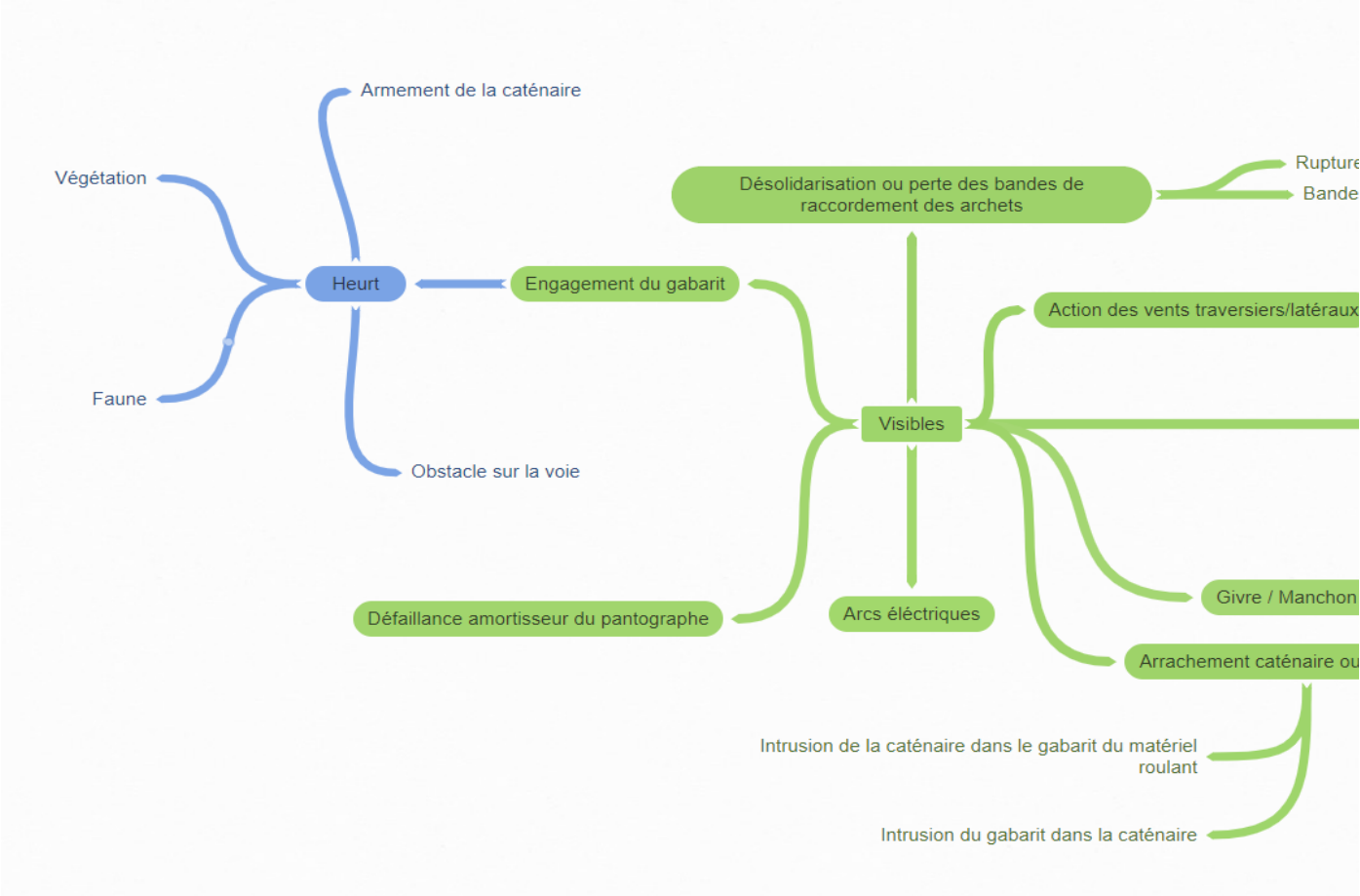
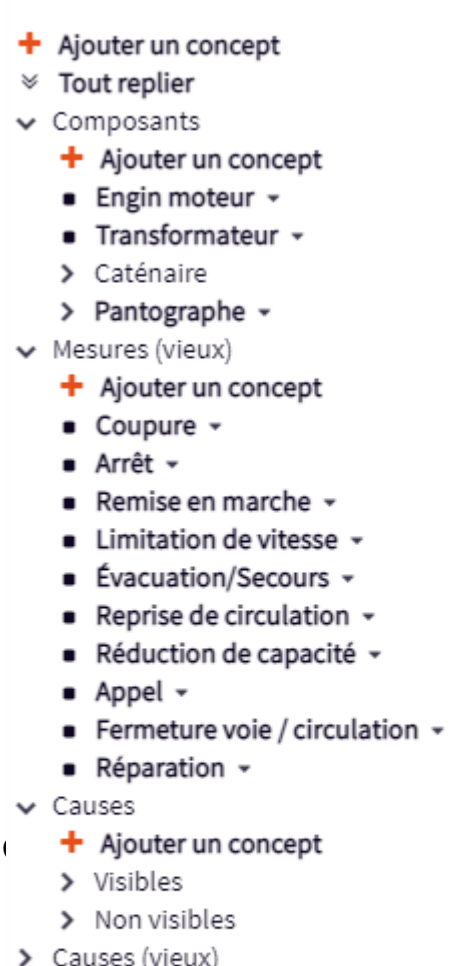


Figure 19 - Échantillon de l’ontologie sur l’incidentologie caténaire



Avec la fonctionalité “Annotate” de Proxem, nous avons ajouté les termes de l’ontologie en tant que concepts. Nous les avons triés par catégorie, conformément aux catégories établies par les rapports d’incident.

Figure 20 - Échantillon des catégories de concepts sur l’incidentologie caténaire

Nous avons ensuite généré des graphes grâce à la fonctionnalité “Explore”, afin de pouvoir visualiser des patterns visibles dans les rapports. Par exemple, nous avons généré un diagramme circulaire permettant de visualiser les causes d’incident sur l’ensemble du réseau, donc toutes régions confondues. L’on note que la cause la plus commune est l’arrachement, suivie de l’engagement du gabarit, et puis du vent.

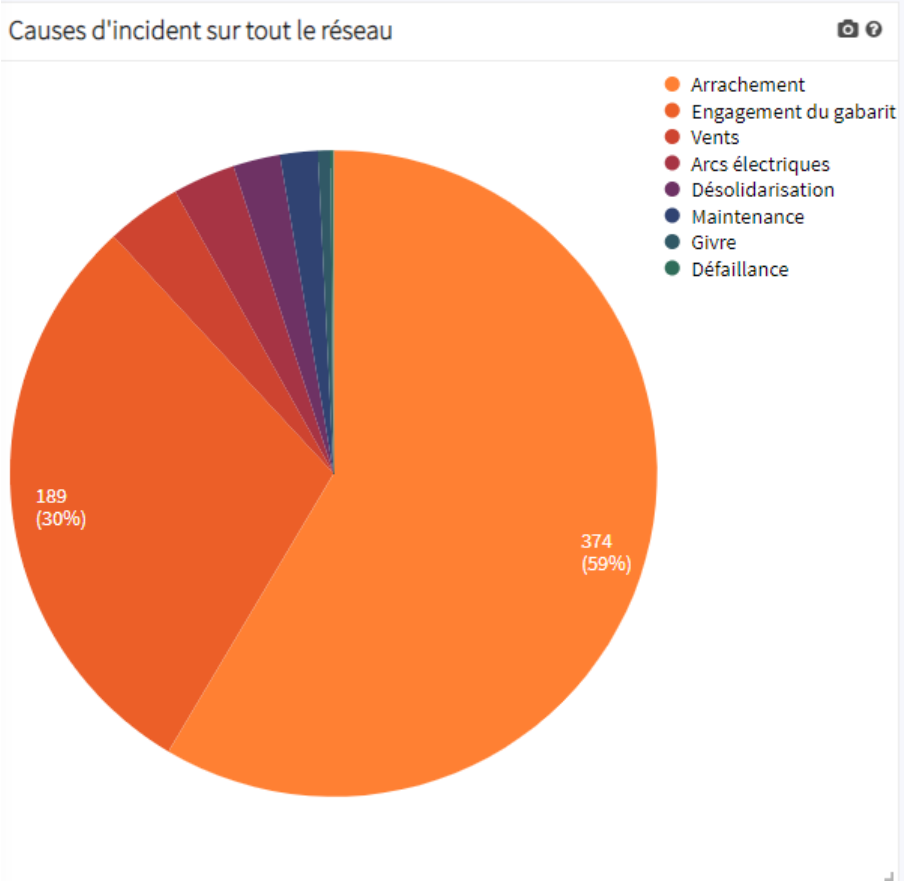


Figure 21 - Diagramme circulaire montrant les causes d’incidents sur tout le réseau ferroviaire

Ce graphe montre les pièces de la caténaire impliquées dans l’ensemble des incidents On peut constater que les composants les plus impliqués sont le fil de contact, suivis du pendule et du porteur.

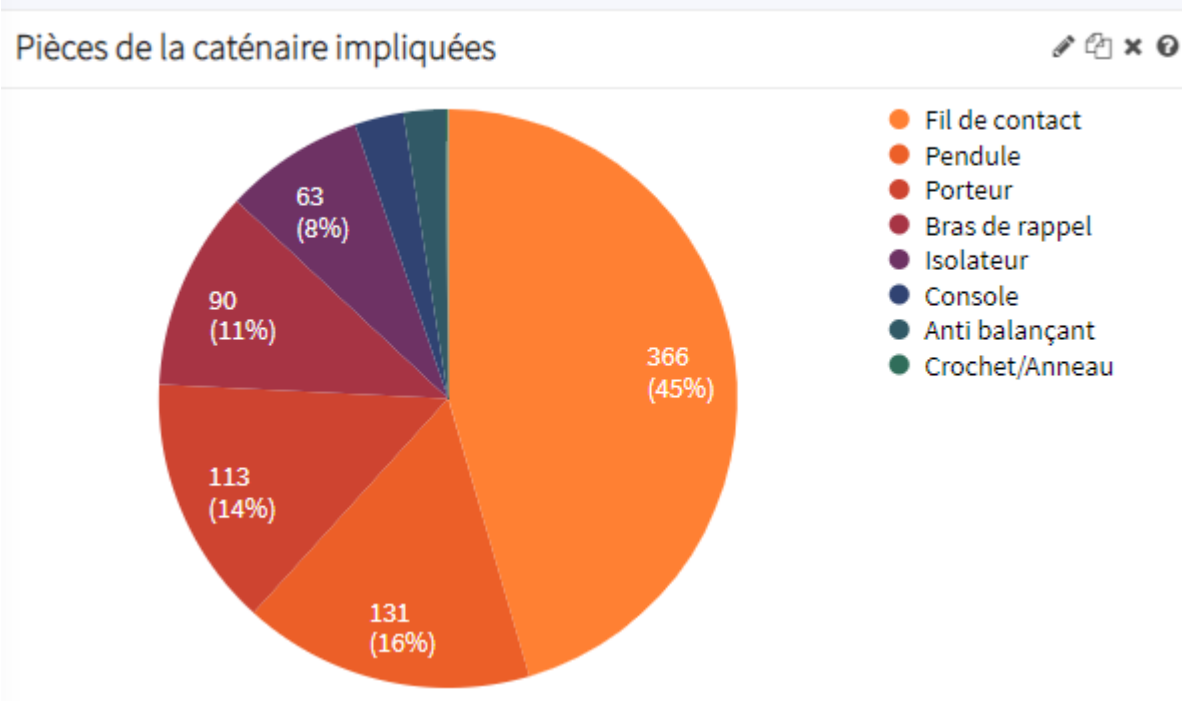


Figure 22 - Diagramme circulaire montrant les pièces de la caténaire impliquées dans les incidents

Ce diagramme en bâtons montre le découpage des types d'incidents par régions. On y retrouve l'arrachement en tant que type principal d'incident sur l'ensemble du réseau, sauf en Hauts-de-France, où le retournement est aussi commun que l'arrachement.

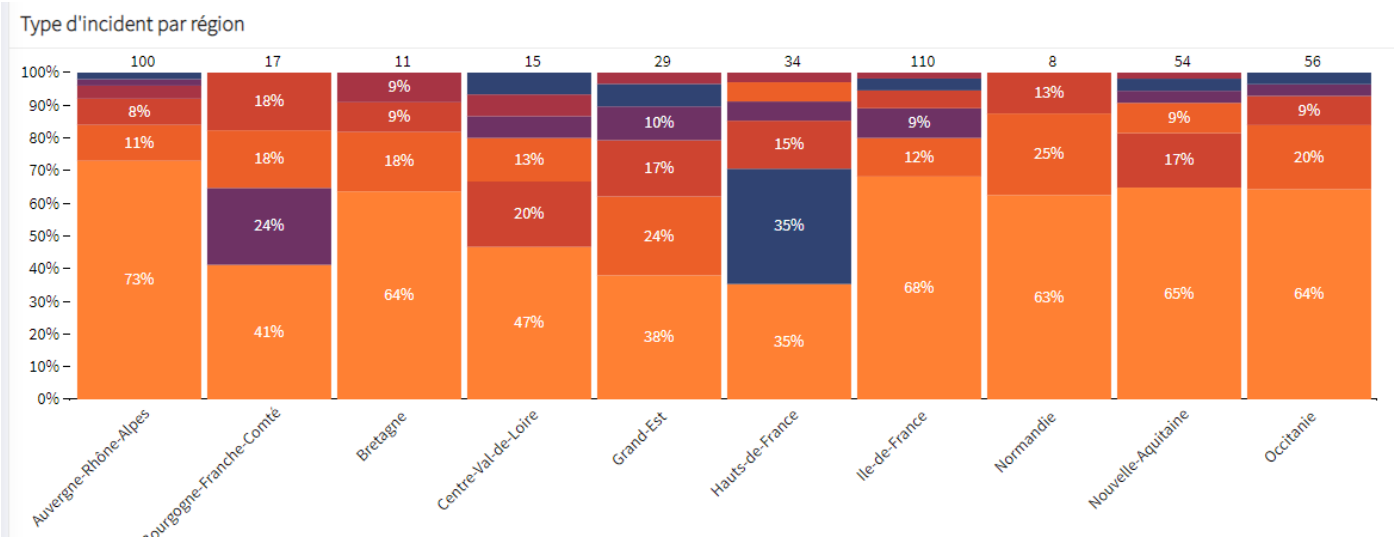


Figure 23.a - Diagramme en bâtons montrant le type d'incident selon la région

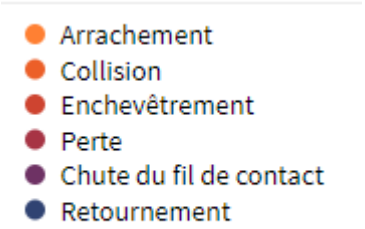


Figure 23.b - Diagramme en bâtons montrant le type d'incident selon la région

b-) Analyse Sémantique avec le Machine Learning:

Proxem nous permet d'avoir une vue d'ensemble du corpus, mais son utilisation demande un savoir-faire technique, et nous limite également à ses fonctionnalités internes. Nous nous sommes donc confrontés à une nouvelle problématique: celle de développer un outil accessible, permettant à un utilisateur lambda (dans ce cas, un agent) de pouvoir lui-même obtenir des prédictions sur les incidents. Cela nous permettrait d'automatiser l'analyse des rapports, et les agents pourraient à la fois obtenir des statistiques pertinentes mais aussi une aide à la décision lorsque confrontés à un incident. Par exemple, l'agent renseigne le lieu et le contexte de l'incident, il peut obtenir des prédictions sur les mesures à suivre, les intervenants à contacter, les risques potentiels etc... La base de données de l'outil pourrait être également nourrie, au fur et à mesure, avec de nouvelles données entrées par les agents, afin d'affiner ses prédictions.

i -) Le script Python:

Afin d'effectuer une analyse sémantique, on effectue d'abord une lemmatisation du corpus. Nous nous sommes focalisés sur les fichiers RAC, sachant que ce sont les rapports les plus complets et ceux qui servent de référence.

▾ Lemmatisation

```
[ ]  def analyze_column(column):
       global all_lemmes
       all_lemmes = []
       for mesure in column[0:300]:
         mesure_lemme =[]
       #newmesure = re.sub("\.|:|\?|\!|,|\.","",mesure)
       #mesure = newmesure
         doc = nlp(str(mesure))
         #print(*[f'word: {word.text+" "}\tlemma: {word.lemma}' for sent in doc.sentences for word in sent.wor
         for sent in doc.sentences:
           for word in sent.words:
               lemma = {word.lemma}
               lemma = re.sub("{'|'}","", str(lemma))
               mesure_lemme.append(lemma)
               lemmes =' '.join(mesure_lemme)
         all_lemmes.append(lemmes)
```

Figure 24 - Echantillon du script Python utilisé, ici on voit une fonction permettant la tokénisation et la lemmatisation du corpus grâce au module Stanza

Nous avons ensuite procédé à la labellisation du corpus. Nous avons utilisé les mêmes labels utilisés sur Proxem, les mêmes ayant été transposés sous forme d'ontologie. Nous avons utilisé les mêmes catégories. Une fonction a été créée pour chaque catégorie, permettant d'identifier les labels de cette même catégorie (donc en cherchant dans une colonne spécifique) à l'aide d'expressions régulières, ceci pour chaque rapport (correspondant à une liste sur le CSV). Nous obtenons pour chaque catégorie une liste de listes, où chaque sous-liste correspond à l'ensemble des labels identifiés.

+ Code + Texte

Labellisation

```
def causes():
  causes = df_RAC["Causes"]
  analyze_column(causes)

  global col_causes
  col_causes = ["arc électrique","arrachement","givre","vent","défailance",
                "désolidarisation","faune","végétation","armement caténaire","engaement du gabarit",
                "usure, mauvais serrage"]
  x = [[] for i in range(len(all_lemmes))]
  for item in all_lemmes:
    arc = re.search(r'arc éléctrique', item)
    if arc:
      x[all_lemmes.index(item)].append("arc électrique")
    arrachement = re.search(r'rupture|casser|arracher|arrachemment|rompre', item)
    if arrachement:
      x[all_lemmes.index(item)].append("arrachement")
    givre = re.search(r'givre', item)
    if givre:
      x[all_lemmes.index(item)].append("givre")
    vent = re.search(r'vent|rafale', item)
    if vent:
      x[all_lemmes.index(item)].append("vent")
    défaillance = re.search(r'défaillance', item)
    if défaillance:
      x[all_lemmes.index(item)].append("défaillance caténaire")
    désolidarisation = re.search(r'désolidariser|désolidarisation', item)
    if désolidarisation:
```

Figure 25 - Echantillon du script Python utilisé, ici on voit la labellisation d'une catégorie (causes)

Les listes obtenues sont ensuite converties en dataframes, et leur contenu est encodé en one-hot (encodage binaire en 0/1) afin que les données puissent ensuite être traitées par les algorithmes de classification. Les différentes catégories peuvent être combinées entre elles dans le dataframe, permettant donc d'effectuer un différent choix de catégories. Le contenu de ce dataframe est ensuite converti en array numpy, afin de pouvoir être traité par les algorithmes de classification.

	arc électrique	arrachement	givre	vent	défailance	désolidarisation	arrêt	freinage	coupure	alerte	couverture
0	1	1	0	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
2	1	1	0	0	0	0	1	0	1	1	0
3	0	0	0	0	0	0	0	0	0	0	0
4	1	0	0	0	0	0	1	0	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...
295	0	0	0	0	0	0	1	0	1	1	0
296	1	1	0	0	0	0	1	0	1	1	0
297	1	0	0	0	0	0	0	0	0	0	0
298	1	0	0	0	0	0	0	0	0	0	0

Figure 26 - Exemple de dataframe généré

	Auvergne-Rhône-Alpes	Bourgogne-Franche-Comté	Bretagne	Centre-Val-de-Loire	Corse	Grand-Est	Hauts-de-France	Ile-de-France	N
0	0	0	0	1	0	0	0	0	
1	0	0	0	0	0	0	0	1	
2	0	0	0	0	0	0	0	0	
3	0	1	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	
295	0	0	0	1	0	0	0	0	
296	0	0	0	0	0	0	1	0	
297	0	0	0	0	0	0	0	1	
298	0	0	0	0	0	0	0	1	
299	0	0	0	0	0	0	0	1	

300 rows × 13 columns

Figure 27 - Exemple de dataframe généré

Nous avons ensuite essayé plusieurs classifieurs multilabels (MLKNN, MLPclassifier, BRKNNAclassifier, LabelPowerset etc…) de Scikit learn. Nous nous sommes focalisés sur la mesure de hamming loss afin d'évaluer leur efficacité, cette mesure étant particulièrement adaptée à la classification mutlilabel. Elle permet d'évaluer le taux de labels individuels incorrectement prédits. Une mesure basse indique donc une classification pertinente.

```
from skmultilearn.adapt import MLkNN
from sklearn.metrics import accuracy_score

mlknn_classifier = MLkNN()
mlknn_classifier.fit(X_train, y_train)

predicted = mlknn_classifier.predict(X_test)

print(accuracy_score(y_test, predicted))
print(hamming_loss(y_test, predicted))
```

```
0.22
0.2613333333333333
```

Figure 28 - Échantillon du script, où l'on applique l'algorithme mlknn à nos données

Nous sauvegardons ensuite le modèle avec joblib et nous le testons sur le corpus, en faisant varier les catégories. Le script permet de générer un résultat final avec une liste de labels et leur taux de certitude en %, grâce à la fonction predict_proba() de scikit learn.

```
zipped = list(zip(colnameslist, flatten))
res = sorted(zipped, key = lambda x: x[1], reverse = True)

print(res)

dfresults = pd.DataFrame(res, columns = ['Label', 'Probabilité'])


dfresults.to_csv(r'predictionresults.csv', index= False)
```

Figure 29 - Échantillon du script, où l'on obtient une liste des labels prédits avec leur taux de certitude

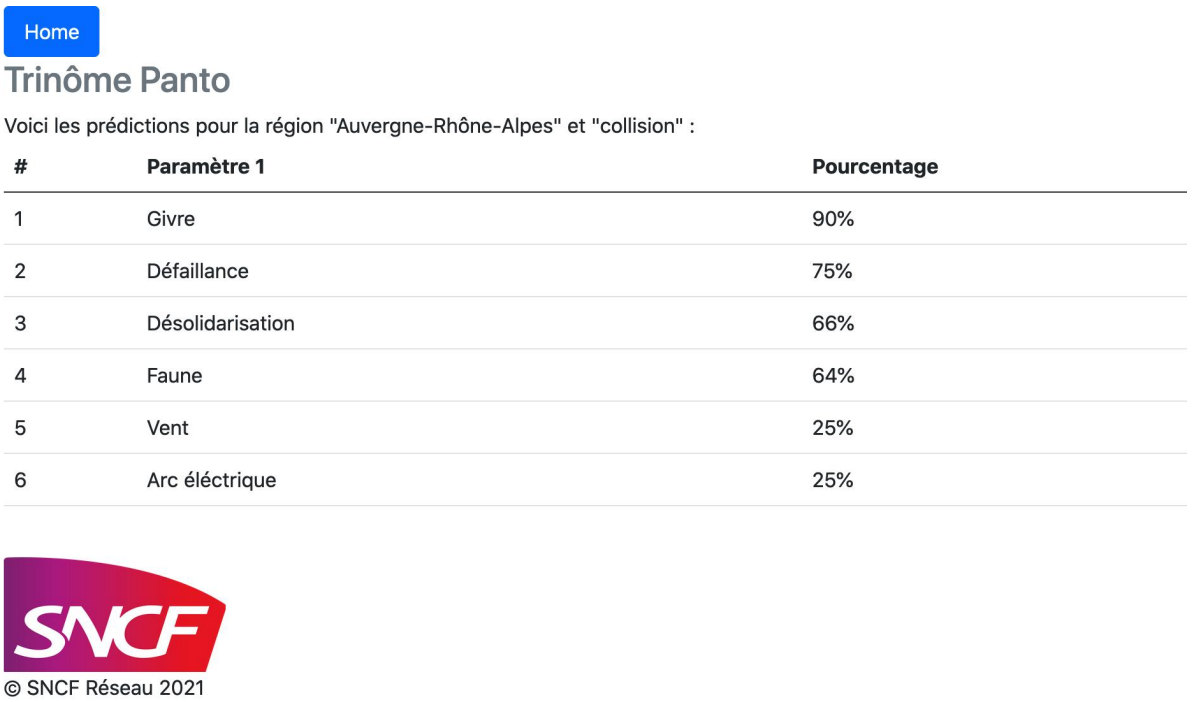


Figure 30 - Exemple de labels générés

Afin d'affiner les concepts, nous avons extrait les mots-clés à partir de chaque catégorie. Les labels initialement pris en compte viennent de la documentation, des synonymes ou d'autres labels peuvent être identifiés à partir du corpus lui-même. Nous avons utilisé NLTK pour exclure les stopwords de la liste des mots-clés et effectuer la tokénisation.

```
[ ]  import nltk
     nltk.download("stopwords")
     from nltk.corpus import stopwords
     from nltk.tokenize import word_tokenize

     stop_words = set(stopwords.words("french"))
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
def causes():
  causes = df_RAC["Causes"]
  analyze_column(causes)
  sentences = '-'.join(all_lemmes)
  total_words = sentences.split()
  total_word_length = len(total_words)
  print(total_word_length)
  tf_score = {}
  for each_word in total_words:
    each_word = each_word.replace('.','')
    if each_word not in stop_words:
        if each_word in tf_score:
            tf_score[each_word] += 1
        else:
            tf_score[each_word] = 1

# Dividing by total_word_length for each dictionary element
  tf_score.update((x, y/int(total_word_length)) for x, y in tf_score.items())
  print(tf_score)
```

Figure 31 - Échantillon du script d'extraction des mots-clés

On obtient une liste des mots-clés, pour chaque catégorie. Les mots-clés sont ajoutés à un fichier texte, pour être nettoyés et triés, afin de nourrir l'ontologie pré-existente.

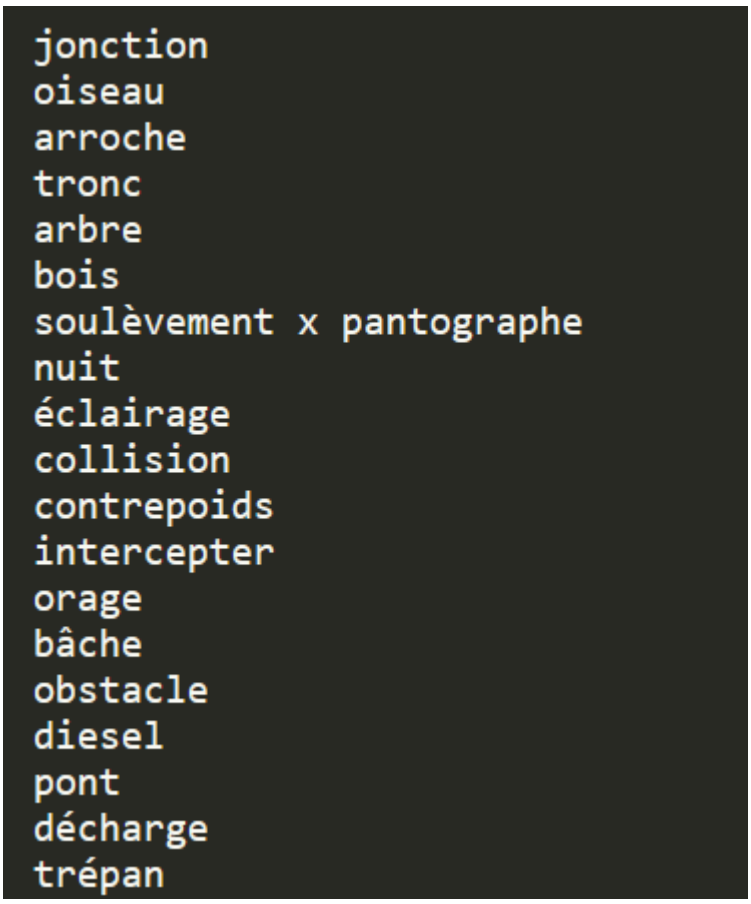


Figure 32 - Exemple de mots-clés générés

ii -) L'interface graphique:

Afin de rendre l'application accessible aux utilisateurs, et de pouvoir la déployer sur le serveur de l'entreprise, nous avons ajouté une interface graphique grâce à Flask, et nous avons d'abord déployé l'application sur notre serveur local.
Nous avons intégré un template HTML à l'application.

```
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/css/bootstrap.min.css" i
    ">
    <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='custom.css') }}">

    {% if title %}
      <title>{{ title }}</title>

    {% else %}
      <title>No title</title>

    {% endif %}
  </head>

  <body>
    <!-- Navigation bar -->
    <header class="site-header">
      <nav class="navbar navbar-expand-md navbar-dark bg-steel fixed-top">
        <div class="container">
          <a class="navbar-brand mr-4" href="/">Trinôme Panto</a>
          <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarToggle" a
            <span class="navbar-toggler-icon"></span>
          </button>
          <div class="collapse navbar-collapse" id="navbarToggle">
            <div class="navbar-nav mr-auto">
              <a class="nav-item nav-link" href="{{ url_for('home')}}">Home</a>
            </div>
          </div>
        </div>
      </nav>
    </header>
```

Figure 33 - Échantillon du script HTML

Le lien entre le template HTML et le script back-end en Python s'effectue avec la bibliothèque Flask. Flask permet de rendre le template dynamique, lui permettant donc de prendre en compte les données entrées par l'utilisateur en temps réel. Par exemple, la fonction POST permet d'envoyer les données d'un formulaire au serveur.

```python
from datetime import datetime
from trinome_panto import db

class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(20), unique=True, nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)
    image_file = db.Column(db.String(20), nullable=False, default='default.jpg'
    password = db.Column(db.String(60), nullable=False)
    posts = db.relationship('Post', backref='author', lazy=True)

    def __repr__(self):
        return f"User('{self.username}', '{self.email}', '{self.image_file}')"

class Post(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(100), nullable=False)
    date_posted = db.Column(db.DateTime, nullable=False, default=datetime.utcno
    content = db.Column(db.Text, nullable=False)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)

    def __repr__(self):
        return f"Post('{self.title}', '{self.date_posted}')"
```

Figure 34 - Échantillon du script Python intégrant Flask

Nous avons intégré Flask à la partie finale du script, celle permettant de générer des prédictions, afin de pouvoir renseigner les labels choisis par l'utilisateur (à travers les fonctions GET et POST). Les labels sélectionnés sont donc envoyés au script, qui va utiliser le modèle de machine learning sauvegardé afin de générer les labels prédits.

```
import email_validator
from trinome_panto.models import User, Post

import numpy as np
import pandas as pd
import joblib

# Import predictions
#df = pd.read_csv('./trinome_panto/ml/predictionresults.csv')

# Homepage
@app.route("/", methods=['GET','POST'])
@app.route("/home", methods=['GET','POST'])
def home():
    form = Category()

    if request.method == "POST":
        #global region_choix, contexte_choix, y_category_choix
        region_choix = request.form['region']
        contexte_choix = request.form['contexte']
        y_category_choix = request.form['y_category']

        ### INSERT MACHINE LEARNING HERE ###
        df1 = pd.read_csv('./trinome_panto/ml/df1.csv')
        df2 = pd.read_csv('./trinome_panto/ml/df2.csv')
        model = joblib.load(open("./trinome_panto/ml/classifier.joblib"

        labels = []
        labels.append(region_choix)
        labels.append(contexte_choix)

        columns = []
        for item in (df1.columns):
          columns.append(item)

        #on lit le nom des colonnes renseignées
        columns = []
```

Figure 35 - Échantillon du script Python permettant d'allier Flask et la prédiction

Nous pouvons lancer l'application en local, depuis le terminal Unix. On y accède ensuite depuis le serveur local, grâce à un navigateur (http://localhost:5000/).

```
yr@yr-TUF-Gaming-FX505DD-TUF505DD:~$ python3 /home/yr/Downloads/flask/trinome_pa
nto/run.py
/home/yr/.local/lib/python3.8/site-packages/flask_sqlalchemy/__init__.py:872: FS
ADeprecationWarning: SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead an
d will be disabled by default in the future.  Set it to True or False to suppres
s this warning.
  warnings.warn(FSADeprecationWarning(
 * Serving Flask app 'trinome_panto' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployme
nt.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with stat
/home/yr/.local/lib/python3.8/site-packages/flask_sqlalchemy/__init__.py:872: FS
ADeprecationWarning: SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead an
d will be disabled by default in the future.  Set it to True or False to suppres
s this warning.
  warnings.warn(FSADeprecationWarning(
 * Debugger is active!
 * Debugger PIN: 142-780-667
```

Figure 36 - Le script “run.py” permettant de démarrer l’application depuis le Terminal Unix

On peut visualiser l’interface et effectuer le choix des catégories à prédire. Ici on choisit la région Auvergne-Rhône-Alpes et le contexte “ Retournement”. On demande à prédire la cause.

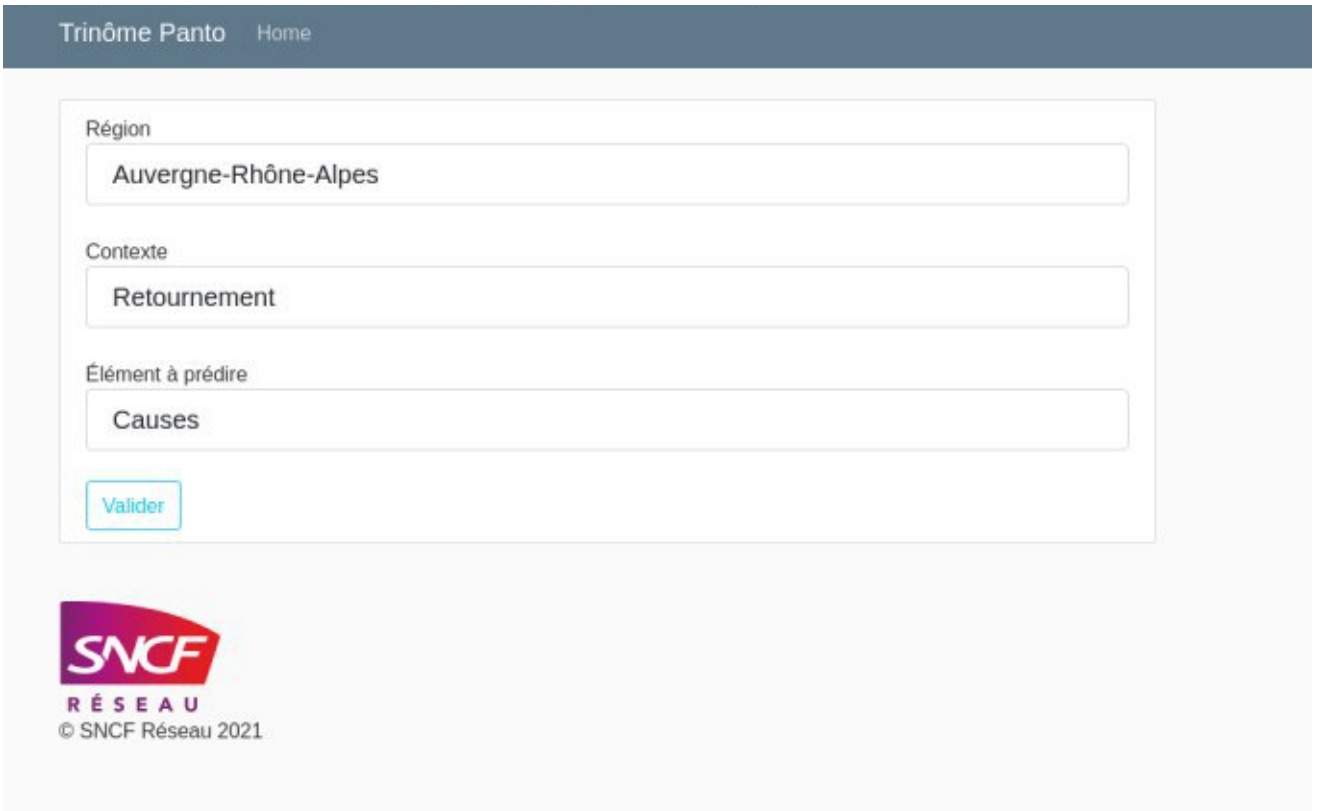


Figure 37- La page d’accueil de l’application

La cause la plus probable est la végétation, avec une certitude de plus de 97%. Aucun autre label ne possède un taux de certitude suffisamment élevé.

Figure 38 - La page de résultats de l'application

C-) Gestion de Projet:

Afin de mener à bien ces activités et nous familiariser avec le processus de réalisation d'un projet dans le cadre d'une entreprise, nous avons dû effectuer plusieurs tâches de gestion de projet. Nous avons dû apprendre à communiquer avec le client, à cerner de façon précise les objectifs du projet, ainsi que les contraintes de réalisation comme les délais.

1-) La méthode agile:

Les pratiques agiles désignent la collaboration entre une ou des équipes pluridisciplinaires et le client, elles mettent en avant la communication. Ces méthodes proposent donc un travail adaptatif et continu, centré sur la flexibilité.
Elles impliquent le client et permettent une grande réactivité à ses besoins.

Lors des projets effectués dans le cadre de ce stage, nous avons utilisé la méthode agile. Ainsi, nous avons communiqué de façon continue et régulière avec les clients, ainsi qu'avec le métier. Nous avons donc effectué le suivi de notre progression, pour s'assurer que cette dernière correspondait aux objectifs. Les verbatims ou labels identifiés ont également bénéficié des suggestions du clients ou du métier, afin de correspondre le plus aux attentes.
Par exemple, nous avons dressé l'ontologie des verbatims du projet DGNUM en parallèle avec le client. Afin de s'assurer que la labellisation du projet Trinôme Panto correspondait à la réalité du métier et du terrain, nous avons défini les catégories de l'ontologie avec le métier. Nous avons également trié les mots-clés grâce au savoir-faire technique du métier, pour les attribuer à des catégories de l'ontologie.

```
Causes:

isolateur x engagement de gabarit/désolidarisation/arrachement/


séparation x arrachement/arc électrique

courant x arc électrique/givre/défaut de captation du courant


amorçage x arc électrique

tension x séparation/ arc électrique avec dégradation


différentiel ??

destruction ?? ( heurt ?)
```

Figure 39 - Échantillon de la liste des mots-clés, annotée selon les catégories désignées par le métier

2-) Définition des objectifs:

Nous avons dû définir des objectifs clairs et précis, parfois en parallèle avec les clients ou le métier. Nous avons également dû apprendre à les communiquer, en adaptant parfois notre support et notre méthode de communication à l'interlocuteur. Il s'agit d'identifier la problématique à résoudre, pourquoi on la résout, ainsi que le planning et les dates du projet.

Figure 40 - Slide d'une présentation, listant les objectifs de notre démarche

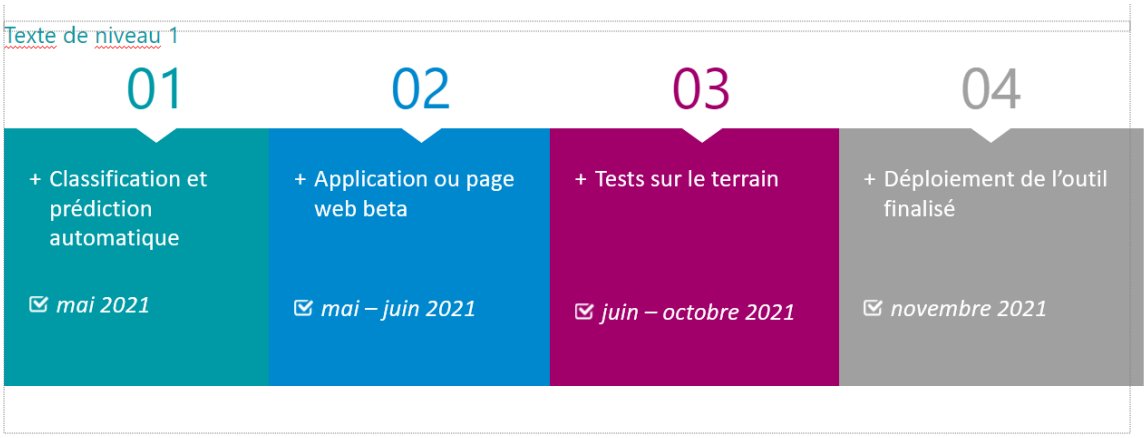


Figure 41 - Slide d'une présentation, avec un mindmap présentant un planning

3-) Planification du projet:

Afin d'identifier les différentes tâches à réaliser et estimer leurs charges, ainsi que les organiser, nous avons réalisé des feuilles de route. Parallèlement, nous avons utilisé la fonctionnalité Planner disponible sur Proxem afin d'identifier le progrès des différentes tâches. Le fait de travailler en binôme sur les mêmes projets demande également de se coordonner, d'où la nécessité de tels outils.

	SEMAINE 23 24-28 MAI	SEMAINE 24 31 MAI – 04 JUIN	SEMAINE 25 07-11 JUIN	SEMAINE 26 14-18 JUIN
INTÉGRATION DU MODÈLE PYTHON À L'INTERFACE				
Séparation du script en plusieurs blocs		lun. 31 mai		
Exécution de ML depuis l'interface		mer. 02 juin		
LECTURE DES PRÉDICTIONS ET AFFICHAGE				
Lecture directe du dataframe à partir du choix utilisateur		ven. 04 juin		
CRÉATION ET AFFINAGE DES CONCEPTS				
Extraction des mots		ven. 04 juin		
Sélection et ajout des mots-clés à l'ontologie			mer. 09 juin	
EXPLORATION ET ÉVALUATION DU MODÈLE				
Recherche de nouvelles mesures d'évaluation				mar. 15 juin
Amélioration du modèle				ven. 18 juin
AMÉLIORATION DU SCRIPT ENTIER				

Figure 42 - Feuille de route

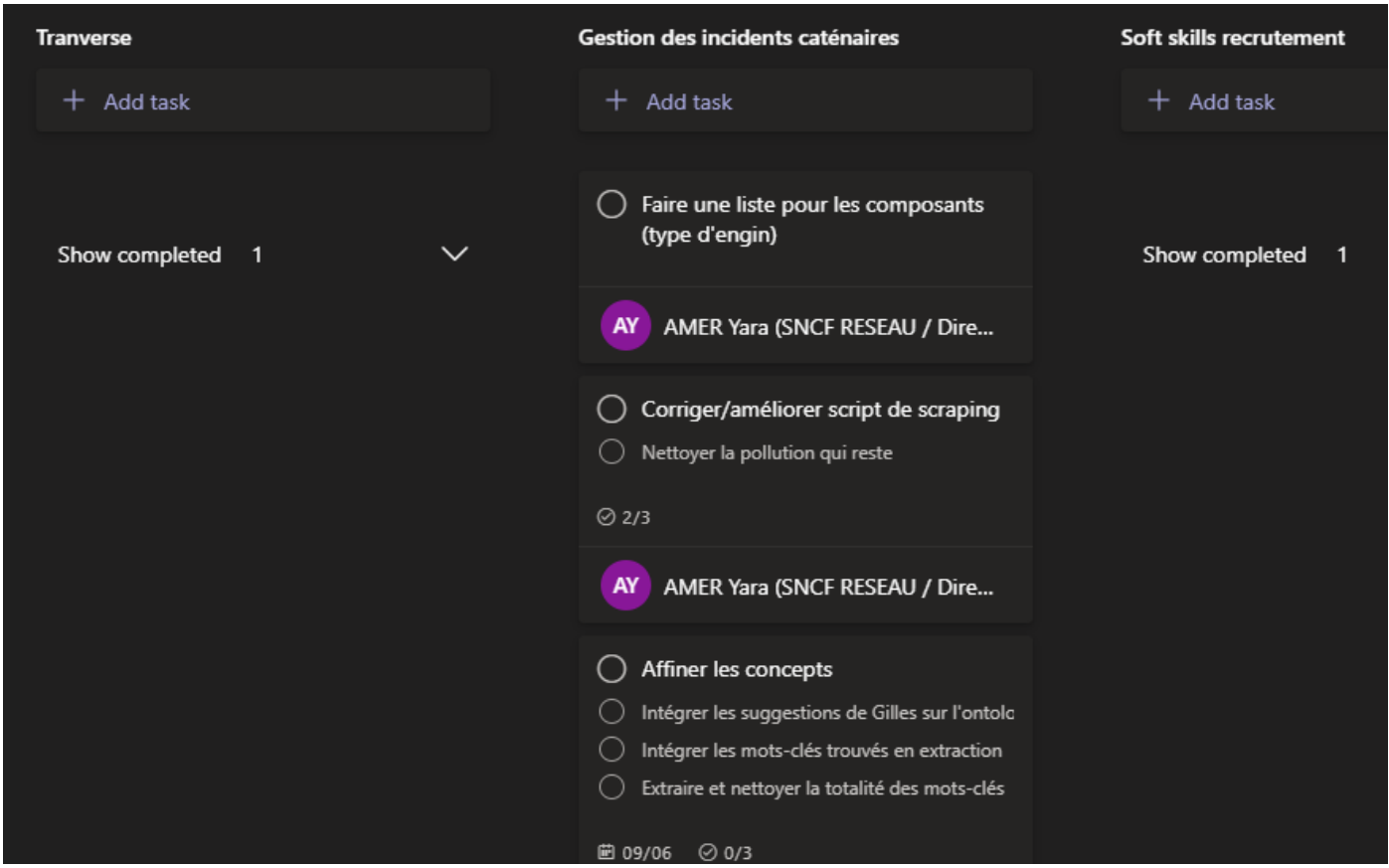


Figure 43 - Tâches du planner sur Teams

Nous avons également réalisé un diagramme inspiré de la structure de WBS (work breakdown structure). Cette structure permet de faire le lien entre les livrables (les résultats) et les tâches à accomplir. Les tâches sont également annotées selon leur priorité (grâce aux émoticones).

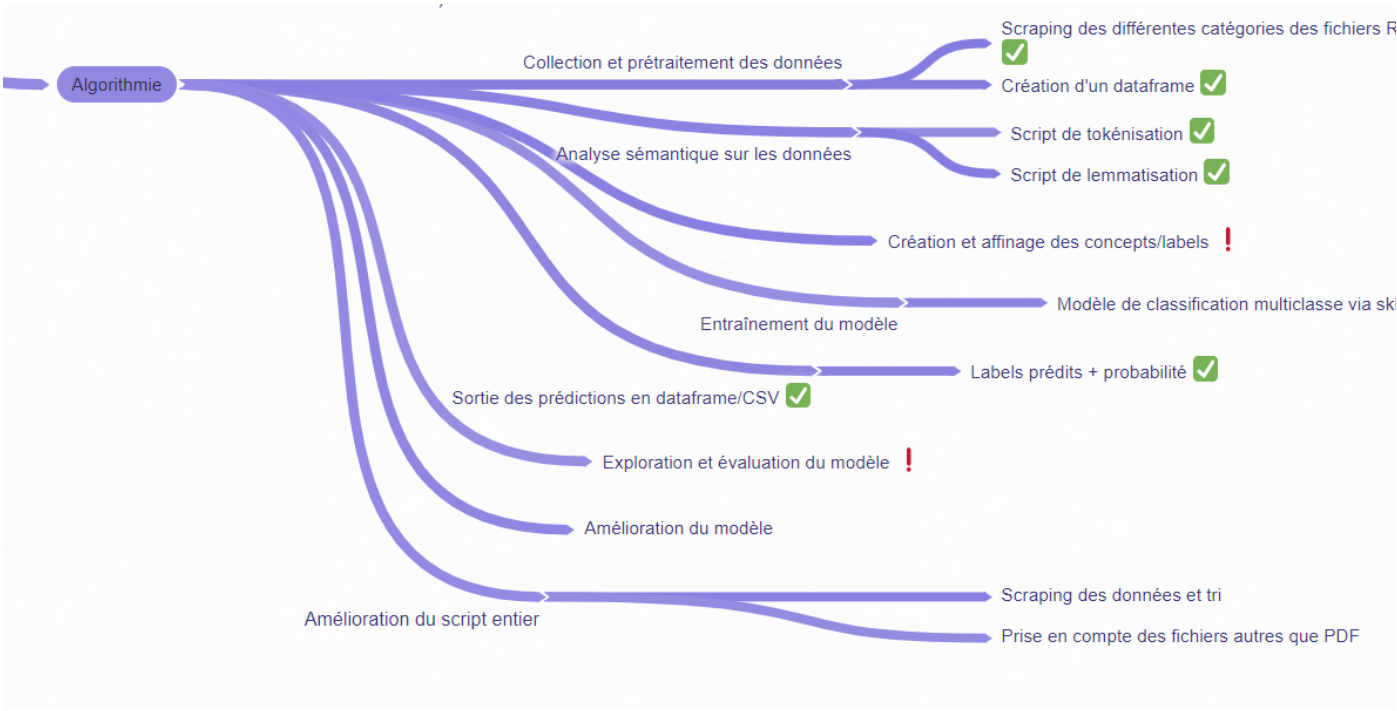


Figure 44 - Mindmap servant de planner pour nos tâches, dénotant leur importance

4-) Les livrables:

Un projet débouche sur un produit ou bien un service. Cette finalité est appelée le livrable, il s'agit d'un résultat concret. Ici, notre livrable est par exemple l'application prédictive réalisée lors du projet Trinôme Panto. Nous avons réalisé des mindmaps sous forme de diagramme, afin de formaliser l'outil. Le livrable doit être précisément défini afin de correspondre aux attentes.

Figure 45 - Slide d'une présentation, avec un mindmap permettant de formaliser l'outil

Conclusion:

Au cours de cette alternance nous avons pu réaliser des projets représentatifs de l'analyse sémantique dans le cadre d'une entreprise, nécessitant d'analyser un grand volume de données afin d'en faire sens. Nous avons ainsi réalisé une analyse d'opinions à travers l'identification de verbatims mais aussi une analyse prédictive à l'aide d'une classification automatique. Nous avons donc pu explorer plusieurs approches utilisées dans l'analyse sémantique, l'analyse semi-automatisée à travers Proxem Studio ainsi que le machine learning.

Pour répondre aux spécificités des problématiques de la SNCF, donc celles d'une entreprise ferroviaire, nous avons dû consulter à la fois la documentation et le métier afin d'acquérir les connaissances techniques nécessaires à la compréhension et au traitement des corpus analysés.

Afin de mener à bien ces projets, nous avons dû nous confronter à la gestion de projet, facette importante du travail en entreprise. Nous avons donc appris à communiquer avec le client de façon régulière selon la méthode agile, ainsi qu'à définir des objectifs et acquérir des capacités de planification.

Certains de ces projets étant encore d'actualité, comme le projet Trinôme Panto, le travail présenté n'est pas encore finalisé et est susceptible d'évoluer. L'application n'a pas encore été déployée, par exemple. Nous allons également travailler sur de nouveaux projets

Ce stage m'a permis d'allier les enseignements théoriques de ma formation à leurs applications concrètes dans le cadre d'une entreprise. J'ai notamment été intéressée par l'aspect Machine Learning, que j'espère explorer davantage à l'avenir.

Sitographie:

Analyse Sémantique, disponible sur
https://fr.wikipedia.org/wiki/Analyse_s%C3%A9mantique

Coggle, disponible sur
https://en.wikipedia.org/wiki/Coggle

Documentation Proxem Studio, disponible sur
https://proxem.helpdocs.io/l/fr

Flask, disponible sur
https://flask.palletsprojects.com/en/2.0.x/

Méthode Agile, disponible
https://fr.wikipedia.org/wiki/M%C3%A9thode_agile

Microsoft Teams, disponible sur
https://fr.wikipedia.org/wiki/Microsoft_Teams

Multiclass and Multialgorithm Labels, disponible sur
https://scikit-learn.org/0.15/modules/multiclass.html

Planification de Projet, disponible sur
https://blog-gestion-de-projet.com/gestion-de-projet/planification-de-projet/

Smart Studio, disponible sur
SMART STUDIO (sharepoint.com)

SNCF Réseau, disponible sur
https://fr.wikipedia.org/wiki/SNCF_R%C3%A9seau

Société Nationale des Chemins de Fer Français, disponible sur
https://fr.wikipedia.org/wiki/Soci%C3%A9t%C3%A9_nationale_des_chemins_de_fer_fran%C3%A7ais

Stanza, disponible sur
https://stanfordnlp.github.io/stanza/

WBS projet, disponible sur
WBS projet : Définition et exemple de Work Breakdown Structure (blog-gestion-de-projet.com)