UNIVERSITÉ PARIS DESCARTES

ALTRAN

UNIVERSITE PARIS DESCARTES

– UFR DES MATHEMATIQUES ET INFORMATIQUE –

RAPPORT DE STAGE DE MASTER

EN MACHINE LEARNING FOR DATA SCIENCE

Effectué au

Département Recherche - Altran -

Automatisation des Projets de Test

Avec les Méthodes de NLP et Text Mining

Travail de	Encadré par	Responsable académique
Fatiha Kafou	**Dr. Ehab HASSAN** **Dr. Asma HOUISSA**	**Dr. Mohamed Nadif**

Soutenu le 14 septembre 2018

Résumé

Le présent rapport a pour but de présenter le travail au sein du Département Recherche de l'entreprise Altran, dans le cadre du projet GOTTRA++. Le but du projet est d'utiliser les méthodes de Machine Learning et du Traitement Automatique du Langage (TAL) pour automatiser les projets de test. Quant au sujet de stage, celui-ci est focalisé sur l'extraction automatique des scénarii de test à partir des spécifications fonctionnelles, excluant l'étape d'exécution des cas de tests.

Deux approches ont été proposées comme solution à la problématique du sujet de stage. Une première s'appuie sur un nombre de techniques de TAL afin d'extraire les informations pertinentes pour pourvoir générer les scenarii de test à partir de documents de spécifications fonctionnelles. Une deuxième, aboutissant au même résultat, en utilisant les techniques de Machine Learning. Cette deuxième approche s'appuie sur les résultats de la première avec laquelle on a pu annoter automatiquement les données disponibles.

L'originalité du projet GOTTRA++ réside dans le fait que les outils existants qui permettent d'automatiser les projets de test ne se basent pas[1], contrairement à ce qui est fait dans le projet, sur les spécifications fonctionnelles. De plus, ces outils automatisent majoritairement les étapes d'exécution[2], alors que le projet se concentre sur tout le processus entier ; à savoir la planification, conception et exécution.

Abstract

This report aims to present the work achieved in Altran Group's Research Department, as part of the research project GOTTRA++. The project's goal is to automate test projects using Machine Learning and Natural Language Processing (NLP). As for the internship topic, the goal is to automatically extract test scenarios from functional requirements. The execution of those scenarios is out of the scope of the internship.

As a solution to the internship problematic, two approaches have been proposed. The first one is based on a number of NLP techniques used to extract relevant pieces of information from functional requirements in order to generate test scenarios. The second one uses Machine Learning to achieve the same goal, using the annotated data resulting from the first approach.

As far as we know, there exist no tools for test projects automation that are based[1] on functional requirements, unlike what is being done in GOTTRA++. Moreover, those tools generally automate the execution steps[2], while the whole test automation is concerned in GOTTRA++ (planning, conception, and execution). These two main points make the originality of the project.

Remerciements

Je tiens à remercier toutes les personnes qui ont contribué au succès de mon stage et qui m'ont aidé lors de la rédaction de ce rapport.

Je tiens à remercier tout particulièrement mon tuteur de stage, Dr Ehab HASSAN, pour sa disponibilité et son suivi ainsi que Dr Billel GUENI et toute l'équipe au Département Recherche d'Altran pour leur accueil.

J'adresse également tous mes remerciements à :
- Dr Asma HOUISSA qui a su me guider et me conseiller dès son arrivée à l'équipe du projet ;
- Mon tuteur en université Dr Mohamed NADIF pour son écoute et ses conseils ;
- Tout le personnel du Master MLDS de l'université pour l'effort qu'ils ont toujours fait en la faveur des étudiants.

Liste des figures

Table des matières

Chapitre I : Introduction

1. Organisme d'accueil

1.1. Société Altran

Leader mondial du conseil en innovation et ingénierie avancée, le groupe Altran accompagne les entreprises dans leurs processus de création et développement de nouveaux produits et services.

Groupe d'envergure internationale, le groupe Altran est présent dans plus de vingt pays répartis entre l'Europe, l'Asie et les Amériques. En qualité de partenaire stratégique, le groupe Altran propose un accompagnement global des projets de ses clients tout en garantissant un niveau constant de service. Le Groupe a également souhaité conserver une dimension locale afin de permettre un accompagnement spécifique sur des marchés dédiés et de proximité.

Les solutions du Groupe couvrent quatre domaines technologiques principaux, à savoir la gestion du cycle de vie des produits (PLM), les systèmes critiques et embarqués, l'Ingénierie mécanique, et les systèmes informatiques.

Présents dans les principaux secteurs, le groupe Altran est partenaire des acteurs clés du marché, à savoir l'Aéronautique, Défense & Transport ferroviaire ; l'Automobiles, Infrastructure & Transport ; l'énergie, Industrie, et Sciences de la vie ; Télécoms & Média ; et enfin Services financiers & Secteur Public.

En 2017, le groupe Altran a réalisé un C.A. de 2 282 M€ avec un effectif de 33 700 salariés.

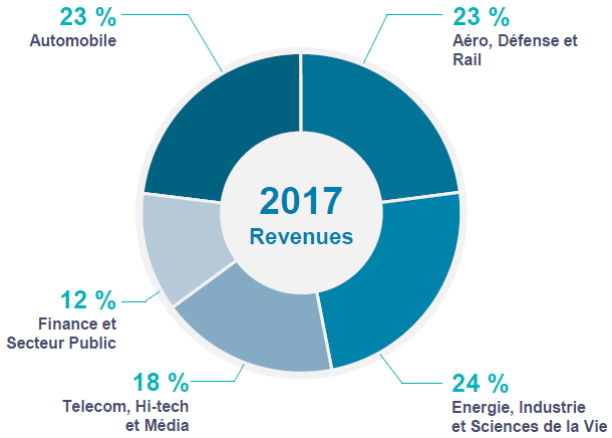


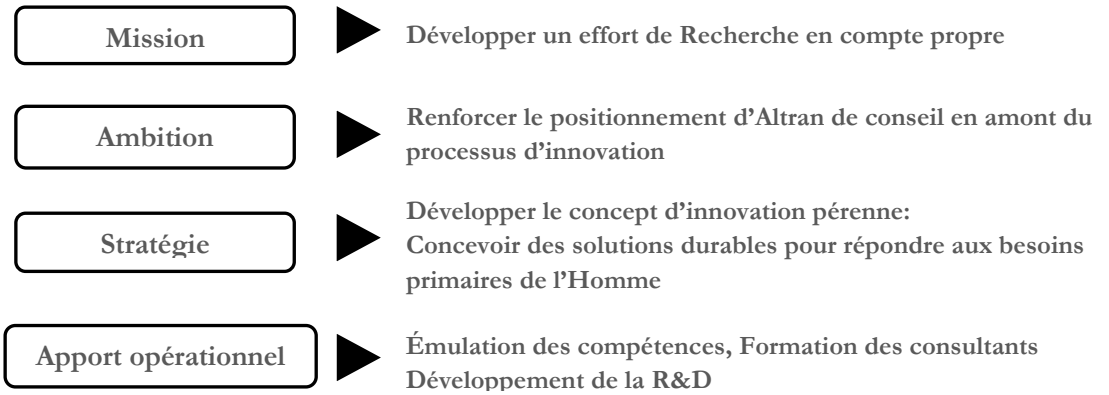
Figure 1: Chiffres-clés du groupe Altran en 2017

1.2. Département recherche - Altran

Le groupe Altran a lancé en janvier 2009 un nouveau département, Altran Research, pour se doter d'une politique de recherche, étoffer son offre de conseil en amont du processus d'innovation et renforcer les compétences de ses consultants en prévision de la sortie de crise. Pilotée par une filiale du groupe, Arendi, spécialiste du conseil en stratégie d'innovation et d'organisation de la R&D,

l'activité de recherche d'Altran mobilise aujourd'hui une centaine de consultants et se focalise sur les « besoins essentiels : boire, manger, se loger, se chauffer, communiquer et se déplacer… ».

Mission, ambition et stratégie :



L'objectif d'Altran Research est de mener des projets de recherche transverses, multidisciplinaires, «frontière», nécessitant l'interaction, la fertilisation de nombreuses connaissances, compétences ou savoir-faire.

2. Projet GOTTRA++

L'objectif principal du projet, lancé en début 2018, est de concevoir et développer un système d'aide à l'automatisation de projets de tests. Initialement, les projets de test à automatiser sont des projets développés en environnement Agile. A cause du manque de ressources, notamment de spécifications fonctionnels de ce type de projets, on s'intéressera alors aux projets développés en cycle en V.

Pour atteindre cet objectif, l'approche à suivre est d'extraire automatiquement les cas d'utilisation et scénarii de test correspondants, à partir de spécifications fonctionnelles. Ensuite, en se basant sur les scénarii de test automatisables, générer des cas de tests et se baser sur ceux-ci pour exécuter automatiquement les tests.

Cependant, la partie de l'exécution des tests ne fait pas partie du périmètre de ce stage. Mes missions pour le stage sont donc les suivantes :

- Proposer un modèle de données standard avec les informations pertinentes présentes dans les documents de spécifications fonctionnelles.
- Faire une analyse lexicale et syntaxique des documents afin d'en extraire les informations pertinentes et remplir le modèle standard.
- Proposer un modèle d'apprentissage permettant d'extraire automatiquement les cas de test à partir de spécifications fonctionnelles.

3. Conclusion

Ce chapitre a mis le point sur les principaux objectifs du sujet de stage. Le chapitre suivant fait le bilan de l'état de l'art en détaillant les outils principaux ayant rapport avec le sujet.

Chapitre II : Etat de l'art

1. Introduction

Un test fonctionnel permet de tester une fonctionnalité (la connexion d'un utilisateur par exemple). Ces fonctionnalités sont testées via des parcours en simulant les actions de l'utilisateur (clics, saisies claviers, mouvement de souris, …).

Les tests manuels sont laborieux et répétitifs. Les automatiser fait gagner du temps aux testeurs qui se chargent de l'exécution des tests principaux. De plus on refait rarement tous les tests à chaque recette. Ce qui signifie que certaines fonctionnalités ne sont pas testées. Le risque de régression est ainsi augmenté. Les automatiser permet de garantir une couverture constante des fonctionnalités.

L'exécution de tests automatisés requiert donc l'utilisation de solutions informatiques dont le but est d'exécuter des actions, soit spécifiquement dans un navigateur web, soit plus généralement au niveau du système d'exploitation.

2. Outils d'automatisation de test

Dans cette section, un ensemble d'outils servant à l'automatisation de test sont présentés.

2.1. Selenium[3]

Selenium est un framework qui permet d'automatiser les tests pour les applications Web. Parmi les composantes de Selenium, il y a Selenium IDE qui est une extension de Firexfox et qui sert à enregistrer l'exécution de tests. Ces tests enregistrés peuvent ensuite être manuellement modifiés et débuggés. Selenium est aussi appelé automate de tests et est compatible avec tous les langages les plus répandus de la programmation web tels que C#, JAVA, Perl, PHP, Python ou Ruby.

Bien que les testeurs soient flexibles et qu'ils puissent rédiger des scripts de test complexes et avancés pour répondre à différents niveaux de complexité, ils ont besoin de compétences en programmation avancées et d'un effort pour créer des infrastructures et des bibliothèques d'automatisation pour des besoins de test spécifiques.

2.2. HP UFT[6]

UFT (Unified Functional Testing) est un ensemble des outils nécessaires à la conception, à l'exécution et au reporting des tests automatisés pour les applications client-serveur. Cet outil a été conçu pour les applications Web plus lourdes que celles avec lesquelles Selenium est utilisé par exemple. Le langage utilisé par l'outil est le Visual Basic.

Pour concevoir un test automatisé sous UFT, il faut passer par l'enregistrement d'un cas de test manuel, ensuite par le rejeu du cas de test automatisé. Après l'exécution, une analyse peut être menée en analysant les erreurs. De plus, UFT permet également de gérer la collaboration entre développeurs et testeurs.

2.3. Smartesting CertifyIt[4]

Smartesting CertifyIt supporte la modélisation des comportements applicatifs ainsi que des processus et règles métiers, pour piloter la génération et la maintenance des tests. La gestion de la traçabilité est gérée automatiquement et les tests sont publiés et maintenus sous forme de tests manuels ou de scripts automatiques composés de mots-clés à implémenter.

Cet outil se base sur ce qui est communément appelé le MBT (Model Based Testing) et qui est une activité qui permet de concevoir et de dériver (de manière automatique ou non) des cas de tests à partir d'un modèle abstrait et haut niveau du système sous test (SUT).

2.4. Ranorex[7]

Ranorex Studio est un Framework d'automatisation de test par interface graphique. Le Framework est utilisé pour tester les applications de bureau, Web et mobiles. Il prend en charge le développement de modules de test automatisés à l'aide de langages de programmation standard tels que C # et VB.NET.

Les principales fonctionnalités de Ranorex sont le filtrage des éléments d'interface graphique utilisateur de reconnaissance d'objets GUI, et l'enregistrement et la relecture basés sur des objets.

2.5. Panaya[5]

Panaya, un outil de test automatique, est un éditeur cloud spécialisé dans la gestion des mises à jour d'ERP. Son principal objectif est de standardiser les séries de tests.

Panaya inclut du Machine Learning qui crée automatiquement des cas de tests réels, basés sur les activités des utilisateurs métiers, capturés en production. Ce service permet d'améliorer l'efficacité des tests et génère automatiquement une documentation de test, prêt à l'usage. Cela répond aux exigences de conformité.

2.6. Comparaison des caractéristiques des outils d'automatisation de test

Le tableau suivant résume les caractéristiques des outils d'automatisation de tests présentés dans cette section :

Tableau 1: Récapitulatif des outils d'automatisation de tests

	Selenium	**UFT**	**CertifyIt**	**Ranorex**	**Panaya**
Disponible depuis	2004	1998	2015	2010	2006
Application Sous Test	Apps Web	Web (UI et API), Mobile, Desktop	Apps Web	Web (UI et API), Mobile, Desktop	Apps Web
Plateformes supportées	Windows, Linux, OS X	Windows	Windows, Linux	Windows	Windows
Langages de script	Java, C#, Perl, Python, Javascript, PHP, Ruby	VBScript	TOCL	Java, C#, VB.net	NA

Compétences en programmation	Compétences avancées requises	Pas nécessaires, mais recommandées pour des scripts de test avancés	Compétences avancées requises	Pas nécessaires, mais recommandées pour des scripts de test avancés	NA
Facilité d'utilisation	Requiert compétences avancées pour l'installation et l'utilisation	Complexe à installer, requiert une formation pour le faire fonctionner correctement	Facile à installer, requiert une formation pour le faire fonctionner correctement	Facile à installer et à utiliser	Facile à installer et à utiliser
Approche	Enregistrement des activités des testeurs	Enregistrement et relecture des activités des testeurs	Model Based Testing	Enregistrement et relecture des activités des testeurs	Machine Learning
Données en entrée	Tests enregistrés	Tests enregistrés	Modèle abstrait et haut niveau du système sous test	Tests enregistrés	Activités des utilisateurs métiers capturés en production

2.7. Originalité du projet GOTTRA++

Les outils d'automatisation de tests qui existent actuellement interviennent généralement dans l'étape de l'exécution des tests fonctionnels, en se basant sur des scripts exécutables. Ces scripts peuvent être partiellement générés de manière automatique. Le complément des scripts est donc à l'utilisateur de le générer manuellement.

Contrairement à cela, l'outil proposé intervient sur le processus entier ; à savoir la conception, planification, et exécution.

De plus, les deux outils connus à intervenir également à l'étape de la conception et planification, notamment Panaya et Smartesting, sont basés sur le MBT (Model Based Testing). La différence avec GOTTRA++ est que ce dernier se base sur les spécifications fonctionnelles textuelles pour automatiser les projets de test.

3. Conclusion

Ce deuxième chapitre fait le point sur les outils d'automatisation de tests les plus répandus et concrétise la différence entre ceux-ci et le projet GOTTRA++. Le chapitre suivant présente les détails de la première approche proposée pour le projet GOTTRA++, à savoir la standardisation des spécifications fonctionnelles en utilisant NLP.

Chapitre III : Standardisation des spécifications fonctionnelles en utilisant NLP

1. Introduction

Ce chapitre a pour but de présenter et donner les détails de la première approche proposée, pour la standardisation des spécifications fonctionnelles, qui s'appuie sur le Traitement Automatique du Langage.

2. Approche proposée

Dans cette approche, comme illustrée dans la figure ci-dessous, cela consiste à ce qu'un utilisateur charge des documents textuels de spécifications fonctionnelles (format Word, PDF, ou Txt), qui utilise une application web. Cette application web a été développée dans le cadre du projet GOTTRA++, et est démontrée dans l'annexe 2.

Un ensemble de techniques, détaillées dans la section 4 de ce chapitre, sont appliquées afin de standardiser les spécifications fonctionnelles récupérées. L'intérêt de cette étape est expliqué dans la section 3 de ce chapitre.

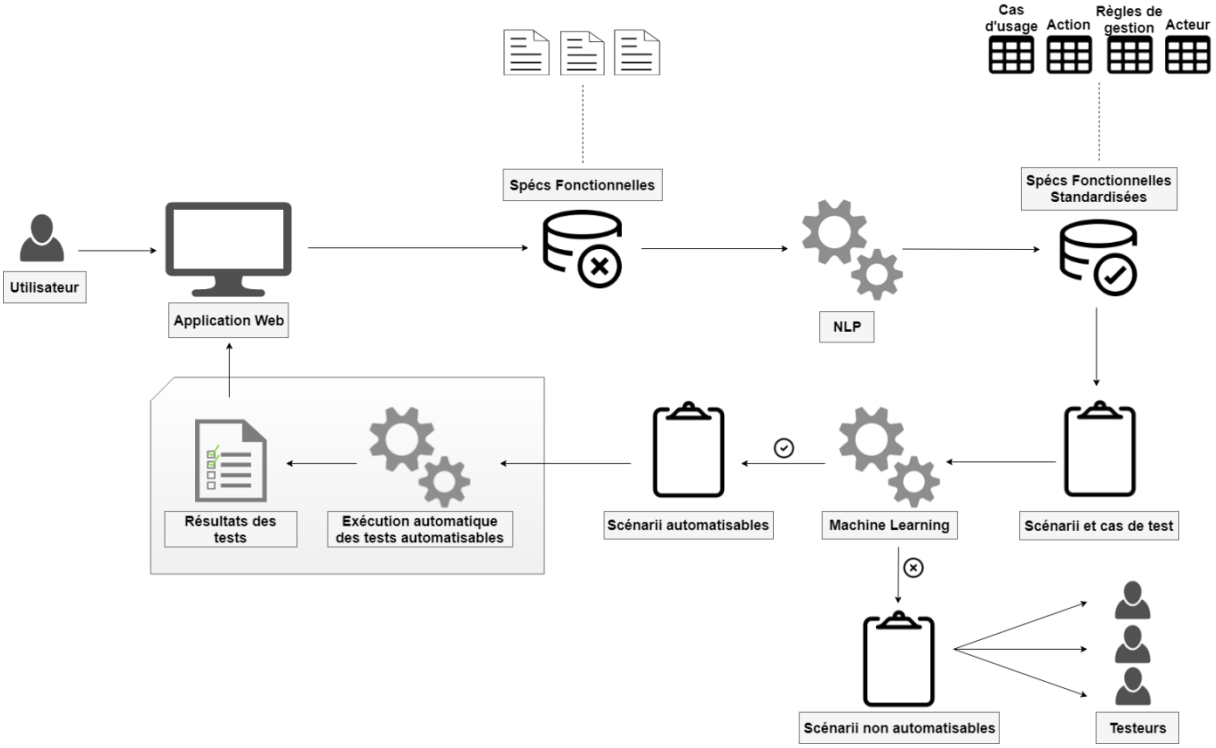


Figure 2: schéma de la première approche (en utilisant NLP)

Ensuite, à partir de ces spécifications fonctionnelles standardisées, les scénarii et cas de test sont extraits. Un exemple de cela est donné dans ce qui suit. En appliquant du Machine Learning, on arrive à distinguer dans l'étape suivante les scénarii automatisables et ceux non automatisables.

Les scénarii non automatisables doivent être testés manuellement par les testeurs. En revanche, ceux qui sont automatisables passent à l'étape de l'exécution automatique et le résultat de ces exécutions est affiché à l'utilisateur. Cependant, cette étape d'exécution et affichage des résultats ne fait pas partie du cadre du sujet de stage.

3. Description des données

Dans le projet, les données en entrée sont les spécifications fonctionnelles textuelles d'un projet quelconque. Ce sont des données sous forme de documents généralement rédigés par des experts où ils décrivent en détail les fonctionnalités du projet.

Les données dont on dispose actuellement, et dont on fait référence dans le reste de ce rapport, sont sous forme de document de spécifications fonctionnelles dont la description est la suivante :

- **Type** : document de spécifications fonctionnelles
- **Objectif** : décrit le produit attendu à l'issue du projet
- **Projet** : application web d'un musée, avec gestion des événements et expositions, galerie des médias, et billetterie
- **Statistiques** : ~300 pages, +80 000 mots

CU103-011 Afficher le bloc d'images

Acteur : Internaute
Cas d'utilisation décrivant la consultation du bloc Evènement en images d'un évènement.

Actions :
Au clic sur l'image en cours de visualisation, l'entité image X est affichée
Un clic sur le lien renvoie l'internaute vers la page d'accueil

Règles de gestion :
Si une valeur de l'entité n'est pas renseignée, la valeur ne s'affiche pas
Le bloc s'affiche si des images ont été contribuées en back-office dans

Figure 3: extrait du document

Le document semble avoir une structure unifié pour tout le texte. Pour chaque cas d'utilisation, il y a des champs spécifiques qui sont remplis selon le besoin. Un exemple où tous les champs sont remplis est donné dans la figure ci-contre.

On distingue notamment quatre éléments pertinents pour chaque cas d'utilisation : **nom du cas d'utilisation** précédé d'un **ID** ; **l'acteur** qui est censé avoir accès à la fonctionnalité décrite par ce cas d'utilisation ; les **actions** que celui-ci peut exécuter pour atteindre l'objectif de la fonctionnalité, et **le résultat attendu** de chacune de ces actions. De plus, des règles, appelées **règles de gestion/règles métier**, permettant de contrôler ou contraindre les fonctionnalités peuvent être décrites.

Il est à noter que ces éléments ne sont pas nécessairement tous présents pour chaque cas d'utilisation.

4. Modèle standard proposé

Afin de pouvoir récupérer les informations pertinentes des documents de spécifications fonctionnelles, on a intérêt à les standardiser en se basant sur un modèle standard. Ceci est dû au fait que tous les documents de spécifications fonctionnelles n'ont pas forcément exactement la même structure, mais contiennent généralement des éléments communs. A l'issue de cette standardisation, on a alors des spécifications fonctionnelles dites structurées.

Pour ce faire, deux modèles standards ont été proposés. Ceux-ci retiennent les éléments les plus pertinents pour le reste des étapes du processus de l'automatisation. L'un des deux modèles est illustré dans le tableau de l'annexe 1 et correspond au modèle standard pour les spécifications fonctionnelles des projets dans l'environnement Agile. Les projets sont développés dans cet environnement de manière itérative et incrémentale, et sont généralement plus « formels » que l'environnement traditionnel.

Comme on s'intéresse plutôt aux projets en cycle en V (méthode traditionnel), on a proposé un modèle standard pour ce type de projets. Celui-ci est illustré dans la figure suivante sous forme de modèle conceptuel de données :

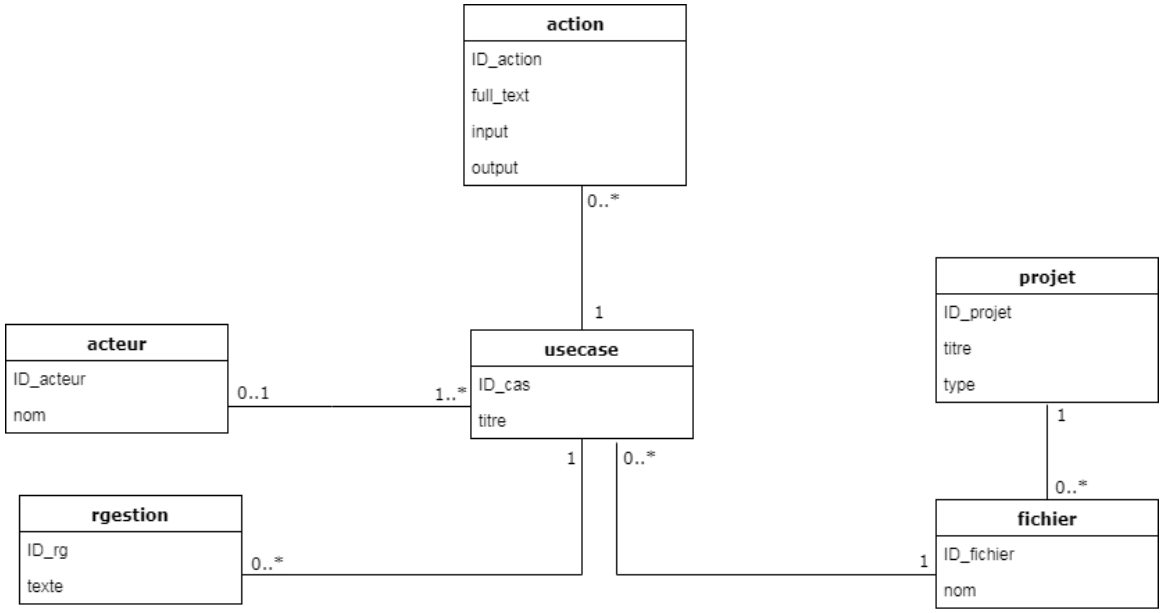


Figure 4: modèle conceptuel de données

Le modèle contient quatre tables principales :

- **Action**: est décrit par un ID, et contient un texte qui lui-même est divisé en un input et un output. L'input signifie l'action que l'utilisateur effectue, et l'output est le résultat attendu de cette action.
- **Usecase**: contient le titre du cas d'utilisation et un ID pour chaque usecase. Chaque cas d'utilisation peut avoir un acteur, plusieurs règles gestion, et plusieurs actions. Il appartient à un unique fichier.
- **Rgestion**: contient un ID et du texte décrivant la règle de gestion.
- **Acteur**: contient un ID et un nom.

Il existe également deux autres tables ajoutées pour structurer la base de données :

- **Fichier**: contient un ID et un nom. Chaque fichier appartient à un unique projet. Pour stocker les documents des spécifications fonctionnelles.
- **Projet**: contient un ID, un titre et un type : Agile ou standard/cycle en V. Un projet peut contenir plusieurs fichiers.

Ce modèle a été implémenté comme base de données MySQL.

5. Techniques de NLP utilisées

Afin de pouvoir extraire les informations pertinentes du texte et remplir le modèle standard proposé, des traitements, ont été appliqués en utilisant le langage Python. Ceux-ci sont détaillés dans cette section.

5.1. Expressions régulières

Les expressions régulières, ou plus communément regex (contraction de regular expression), permettent de représenter des modèles de chaînes de caractère. Elles ont été utilisées dans notre cas pour représenter les cas d'utilisation pour pouvoir les extraire par la suite.

Dans l'exemple de spécification fonctionnelle donnée dans la section 2, la phrase qui représente le cas d'utilisation est : « CU103-011 Afficher le bloc d'images ». Après avoir analysé le document, il s'est avéré que tous les cas d'utilisation respectent la même structure syntaxique de cette phrase.

L'expression régulière qui représente alors les cas d'utilisation est la suivante :

[a-zA-Z]+\d+-\d+

Cette expression régulière représente les phrase commençant par au moins une lettre minuscule ou majuscule, suivie d'au moins un chiffre, suivi d'un tiret « - », suivi d'au moins un chiffre. Ceci permettra d'extraire dans notre exemple la chaîne de caractère : « CU103-011 ». Elle sera alors affectée à l'ID du cas d'utilisation. Le reste de la phrase, à savoir la chaîne de caractère « Afficher le bloc d'images », sera affectée au nom du cas d'utilisation.

5.2. Tokenisation

La tokenisation est l'opération de segmenter un texte en unités "atomiques" : les tokens. Les tokenisations les plus courantes sont le découpage en mots ou bien en phrases. Ainsi la tokenisation en mots de la phrase « Cas décrivant la consultation du bloc. » nous donnerait les tokens suivants : « Cas », « décrivant », « la », « consultation », « du », « bloc », « . ».

L'intérêt de cette technique dans notre cas réside dans le fait que, quand on est en train d'extraire les actions se trouvant dans le texte des spécifications fonctionnelles, on doit chercher les verbes afin de déterminer les inputs et les outputs.

Dans notre exemple, les actions sont :

- Au clic sur l'image en cours de visualisation, l'entité image X est affichée.
- Un clic sur le lien renvoie l'internaute vers la page d'accueil.

Dans le premier exemple, un simple *split* basé sur la présence d'une virgule « , » est capable de séparer l'input, à savoir « Au clic sur l'image en cours de visualisation », et l'output, à savoir « l'entité image X est affichée ». Or, dans la deuxième action, ce qui permettrait de séparer l'input et l'output est la présence du verbe « renvoie ».

Il est alors nécessaire de segmenter la phrase en mots afin de pouvoir reconnaître le verbe parmi les mots. Ceci sera alors fait dans l'étape décrite dans la section suivante.

5.3. Part Of Speech (POS) Tagging

Après avoir effectué la tokenisation des phrases qui suivent le mot clé « Actions : », et ne contenant pas une virgule, on peut appliquer le POS Tagging pour reconnaître les verbes.

En effet, le POS Tagging permet d'affecter à chaque token les informations grammaticales correspondantes. Si on reprend l'exemple de la phrase « Cas décrivant la consultation du bloc. », le résultat du POS Tagging serait le suivant :

'Un DET:ART,
'clic NOM,
'sur PRP,
'le DET:ART,
'lien NOM,
'renvoieVER:pres,
…

Ce résultat montre un mot (token) par ligne suivi de son étiquette grammatical (POS). Dans notre exemple, on a eu comme résultat un article suivi d'un nom, suivie d'une préposition… jusqu'à ce qu'on trouve le mot « renvoie » dont l'étiquette grammatical est un verbe.

Ainsi, on récupère le verbe de la phrase « Un clic sur le lien renvoie l'internaute vers la page d'accueil. ». On peut alors distinguer l'input, à savoir « Un clic sur le lien » de l'output, à savoir « renvoie l'internaute vers la page d'accueil. ».

5.4. Résultat

Les traitements appliqués, décrits dans les sections précédentes nous ont permis d'extraire les informations pertinentes et ainsi remplir les tables de notre modèle standard, à partir du texte de spécifications fonctionnelles.

Le résultat de ces traitements pour l'extrait donné dans la section 2 est donné dans les tableaux suivants :

Cas d'usage	
ID	**Titre**
'CU103-011'	'Afficher le bloc d'images'

Acteur
'Internaute'

Action	
Input	**Output**
'Au clic sur l'image en cours de visualisation'	'l'entité image X est affichée'
'Un clic sur le lien'	'renvoie l'internaute vers la page d'accueil'

Règle de gestion
'Si une valeur de l'entité n'est pas renseignée, la valeur ne s'affiche pas'
'Le bloc s'affiche si des images ont été contribuées en back-office dans l'onglet Images'

Figure 5: modèle standard rempli

6. Scénarii et cas de test

Les scénarii et cas de tests sont extraits directement à partir des spécifications fonctionnelles standardisées et explicitent les étapes par lesquelles le testeur, ou l'outil d'exécution automatique, doit passer afin de tester un cas d'utilisation donné.

Un exemple pour les cas de test d'un cas d'utilisation est donné dans la figure ci-dessous :

Cas d'usage couvert :	CU001-004	
Libellé du cas d'usage :	Afficher le bouton de partage	
STEP	Action	Résultat attendu
1	Au clic sur le bouton de partage	un menu supplémentaire s'affiche avec 2 boutons :
2	Au clic sur le bouton Facebook	l'événement est partagé sur le réseau social de l'internaute
3	Au clic sur le bouton Twitter	l'événement est partagé sur le réseau social de l'internaute

Figure 6: cas de test pour un cas d'utilisation

Ce cas de test est défini par un ID : « CU001-004 » et concerne le cas d'utilisant intitulé « Afficher le bouton de partage ». Les scénarii à tester sont ensuite décrits en suivant un ordre chronologique qui correspond à leur ordre d'apparition dans le document de spécifications fonctionnelles. Chaque scénario est décrit par une action et un résultat attendu.

7. Conclusion

Les traitements faits dans cette section ont été propres à la structure particulière du document dont on dispose. Comme la structure des documents de spécifications fonctionnelles diffère d'un document à un autre, il est obligatoire de réécrire ou adapter les règles d'extraction pour chaque nouveau document.

D'où l'intérêt de l'étape suivante où l'on va appliquer de l'Apprentissage Automatique en se servant des données annotés dans cette première étape à l'aide de NLP. Ce qui fait l'objet du chapitre suivant détaillant cette deuxième approche.

Chapitre IV : Hybridation NLP et ML pour l'extraction des informations pertinentes

1. Introduction

Dans le chapitre précédent, on a proposé un modèle standard de documents de spécifications fonctionnelles, et on a utilisé les méthodes de NLP afin d'extraire les informations pertinentes à partir de ces documents.

Dans ce chapitre, on se concentre sur la deuxième approche qui a pour but d'automatiser l'extraction des informations pertinentes à partir d'un document de spécifications fonctionnelles, pour le transformer en un document de spécifications fonctionnelles structurées.

2. Approche proposée

La deuxième approche proposée consiste à utiliser les données précédemment annotées en utilisant les techniques de NLP, afin de construire un modèle d'apprentissage pour automatiser l'extraction des informations pertinentes à partir d'un document de spécifications fonctionnelles.

Le schéma de cette deuxième approche d'extraction d'informations pertinentes est illustré dans la figure suivante :

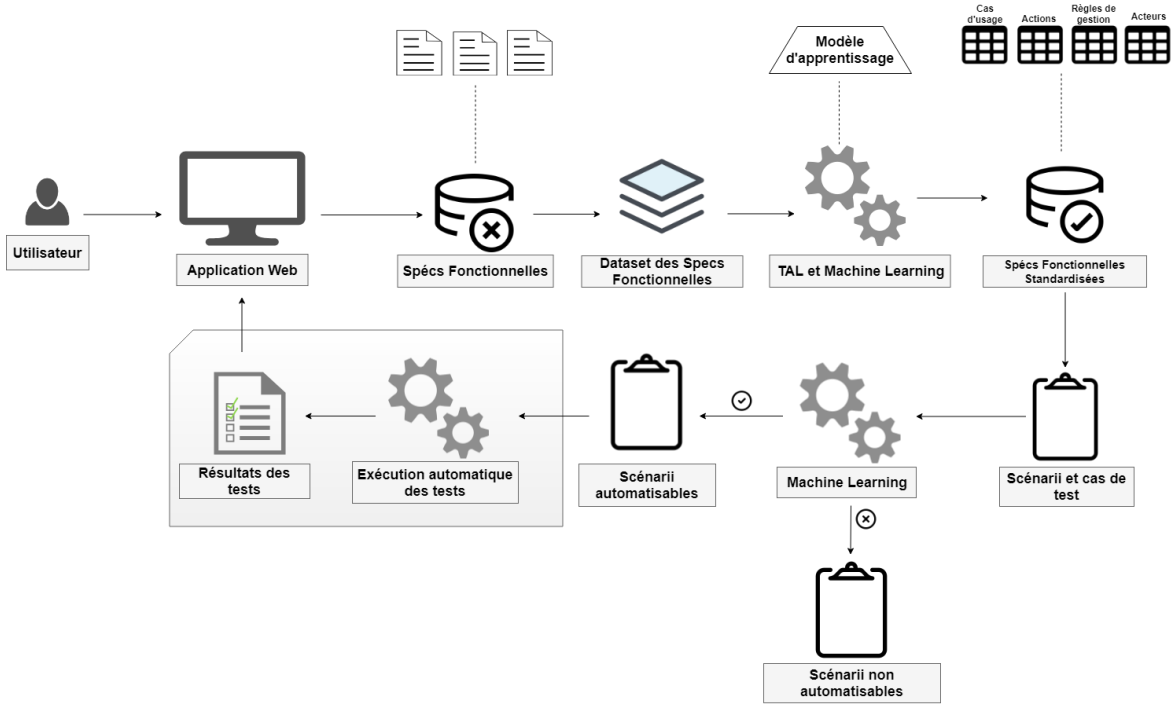


Figure 7: schéma de la deuxième approche

Le nouveau schéma diffère du celui de la première approche (présenté dans le chapitre précédent à la section 2) uniquement dans ce qui est entre les spécifications fonctionnelles, et les spécifications fonctionnelles standardisées. En effet, le document de spécifications fonctionnelles récupéré de l'utilisateur est transformé en un dataset où les paragraphes (séparés par des lignes vides) représentent les observations à classifier.

Ensuite, en passant par un modèle d'apprentissage, le dataset est classifié, et par conséquent standardisé. Ceci résulte alors en un document de spécifications fonctionnelles respectant le modèle standard précédemment proposé.

Dans ce qui suit, on présente les détails de la construction et de la sélection du modèle d'apprentissage le plus optimal utilisé dans cette approche.

3. Modèle d'apprentissage

Pour notre problème de classification, on a analysé trois méthodes d'apprentissage, à savoir le SVM (Support Vector Machine), les réseaux de neurones, et le Naïve Bayes. Au final, il faut sélectionner un seul modèle optimal.

Pour cela, des prétraitements des données ont été appliqués, ensuite, on a évalué les résultats donnés par chaque méthode d'apprentissage sur la base de données annotée. Ce processus est illustré dans le schéma suivant :

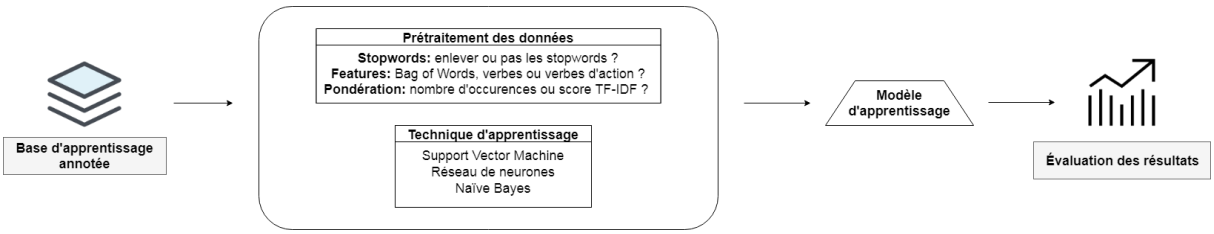


Figure 8: processus d'évaluation des modèles d'apprentissage

On détaille dans ce qui suit l'ensemble des éléments intervenant dans ce processus d'évaluation.

3.1. Base d'apprentissage annotée

Pour la base d'apprentissage, les spécifications précédemment standardisées ont été reprises et transformées en une matrice. Celle-ci contient deux colonnes : **texte**, et **tag**. Dans la colonne texte, on a mis un paragraphe du document par ligne. Le tag (étiquette) correspondant est mis dans la colonne tag. Les valeurs possibles pour la colonne tag sont : (usecase, acteur, action, rgestion, texte).

Un extrait du dataset est illustré dans la figure suivante :

	A	B
1	texte	tag
2	4.Version Desktop - liste des cas d'utilisations	texte
3	4.1.1CU001-001 Afficher le header	usecase
4	Acteur : Internaute	acteur
5	Cas d'utilisation permettant à un internaute de consulter le header du site. L'utilisateur visualise le header	texte
6	•Notes :	texte
7	•Actions :	action
8	4.1.2CU001-002 Afficher le bouton de langue	usecase
9	Acteur : Internaute	acteur
10	Cas d'utilisation permettant à un internaute de consulter le bouton de langue.	texte
11	•Notes :	texte
12	•Actions :	action
13	•Règles de gestion :	gestion
14	4.1.3CU001-0021 Afficher la page Langue	usecase

Figure 9: extrait de la base d'apprentissage

3.2. Prétraitement des données

Dans cette section, on décrit les différentes étapes par lesquelles on passe pour préparer nos données avant d'appliquer les techniques d'apprentissage. Il est à noter que ces prétraitements sont faits indépendamment des méthodes d'apprentissage utilisés.

3.2.1. Modèle Bag Of Words et One Hot Vector

Pour pouvoir appliquer des techniques d'apprentissage sur nos données, on a construit un modèle « Bag of Words ». Pour la construction de ce modèle, on utilisera la matrice où chaque paragraphe du document est étiqueté par le tag correspondant. Ce fichier est notre dataset pour le modèle ; il compte plus de 2000 lignes.

Ensuite, un dictionnaire est construit en utilisant l'ensemble des mots du document. Le dictionnaire construit sera utile pour représenter le texte sous forme de vecteurs numériques. Cependant, en utilisant ce modèle, les vecteurs numériques vont uniquement représenter la présence (ou non présence) de chaque mot du dictionnaire. Ni la signification ni la grammaire ni la séquence des mots ne sont prises en considération.

Les observations de nos données sont donc les paragraphes du document des spécifications fonctionnelles (un paragraphe par ligne), et les caractéristiques sont les mots du document.

Exemple :
Supposons que notre dictionnaire contient 4 mots (4 = vocab_size) : ['étant', 'alors', 'quand', 'utilisateur']. La première phrase du dataset est « *quand un utilisateur clique sur le bouton, la fenêtre s'ouvre.* » Cette dernière est alors représentée sous forme : [0 0 1 1], cela correspond à l'absence des mots « étant » et « alors » dans le texte, et la présence des mots « quand » et « utilisateur ». Cette représentation est appelée « Bag of Words ».

Les tags, eux, sont représentés sous forme de « one-hot vector ». Les tags possibles sont : usecase, acteur, action, rgestion, texte. Les « one-hot vector » correspondants sont : [1 0 0 0 0], [0 1 0 0 0], [0 0 1 0 0], [0 0 0 1 0], [0 0 0 0 1]. Pour notre phrase de l'exemple précédent est associé le tag « action » alors le « one-hot vector » [0 0 1 0 0].

3.2.2. Pondération

Pour chaque observation, les valeurs pour les caractéristiques dans la matrice documents termes peuvent être binaires : 0 ou 1, selon la présence absence du mot dans le paragraphe. Elles peuvent aussi être sous forme du nombre d'occurrences des mots, ou sous forme des scores tf-idf comme est le cas dans l'exemple illustré dans la figure suivante :

	YG	YH	YI	YJ	YK	YL	YM	YN	YO	YP	YQ	YR
1	dossier	double	download	drag	dramaturgie	droit	droite	droiteavec	droits	drop	du	duplicata
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.132944727(	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.160433882!	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.072582282:	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.070912532!	0.0
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.055633517!	0.0
13	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
16	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
17	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
18	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
19	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
20	0.288147268:	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
21	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.097414844(	0.0

Figure 10: extrait de la matrice document-terme avec scores tf-idf

Le score tf-idf est calculé en multipliant la mesure $tf_{i,j}$ par la mesure idf_i. Où $tf_{i,j}$ est la fréquence brute du mot i dans le document j : nombre d'occurrences de i dans j, et idf_i est la fréquence inverse de document (inverse document frequency). Cette mesure est calculée comme suit :

$$idf_i = \log \frac{|D|}{\left|\{d_j : t_i \in d_j\}\right|}$$

où :

$|D|$ est le nombre total de le document dans le corpus

$\left|\{d_j : t_i \in d_j\}\right|$ est le nombre de documents où le terme t_i apparaît

L'utilisation de la fréquence tf-idf permet d'avoir une idée sur l'importance d'un mot dans un paragraphe, et si ce mot caractérise le paragraphe ou pas. Il donne moins d'importance aux mots les plus communs dans tous les paragraphes.

3.2.3. Stopwords

Quelques mots (la, le, avec…) sont communs entre la plupart des phrases dans le langage naturel. Tel est le cas pour le mot « du » dans l'extrait dans la figure précédente. Il se peut que même avec les scores tf-idf, la présence de ces mots dans le dataset fausse les résultats. Il convient alors parfois de les enlever des caractéristiques du dataset.

3.2.4. Sélection de variables

Vu qu'on compte près de 2000 mots uniques, cela revient à tourner les classifieurs sur des données de grande dimension (nombre de mots = nombre de caractéristiques). En réduisant le nombre de variables, on optimise l'algorithme de l'apprentissage, améliorant ainsi les performances de classification.

Pour réduire le nombre des variables, il existe un nombre de méthodes tel que LDA, ANOVA, ou encore Chi-square. Cependant, dans notre cas, on s'est servi de notre connaissance préalable de la nature des mots afin de sélectionner ceux susceptibles d'améliorer les résultats de la classification.

Ainsi, la solution proposée est de considérer uniquement les verbes du document, puisque ceux-ci ont généralement le plus de signification dans notre cas : spécifications fonctionnelles. On peut également considérer uniquement les verbes d'action existant dans le domaine d'informatique (comme cliquer, afficher, sélectionner …), afin de réduire davantage le nombre de variables.

Le schéma suivant montre le nombre de variables après avoir passé par une étape de sélection de variables.

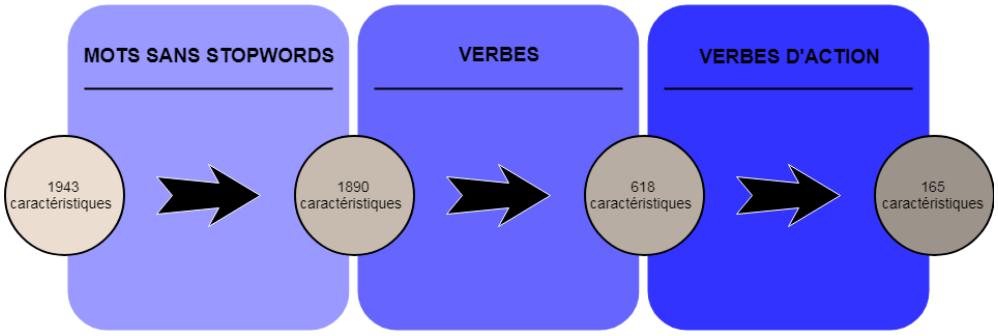


Figure 11: sélection de variables

Le nombre de variables est réduit de 1943 à 1890 en enlevant les stopwords comme décrit dans la sous-section précédente. Ensuite, de 1890 à 618 en gardant que les mots représentant des verbes. Enfin, en ne considérant que les mots représentant les verbes, on réduit le nombre de variables jusqu'à 165.

3.3. Méthodes d'apprentissage

Dans cette section, on décrit les méthodes d'apprentissage analysés pour notre problème, ainsi que leurs hyper paramètres associés.

3.3.1. Réseau de neurones

Une fois nos données sont toutes représentées sous forme numérique, elles sont alors prêtes à être injectées au réseau de neurones. Ce dernier contient une première couche qui correspond aux données d'entrée contenant n neurones, où n est le nombre de mots du vocabulaire. La figure suivante illustre ce réseau de neurones.

La deuxième couche du réseau de neurones est la couche dite « cachée ». Elle contient généralement un nombre puissance de 2 de neurones, chacun d'eux est connecté avec tous les autres neurones de la couche précédente (couche dite Dense). On fixe n = 512 neurones dans notre cas.

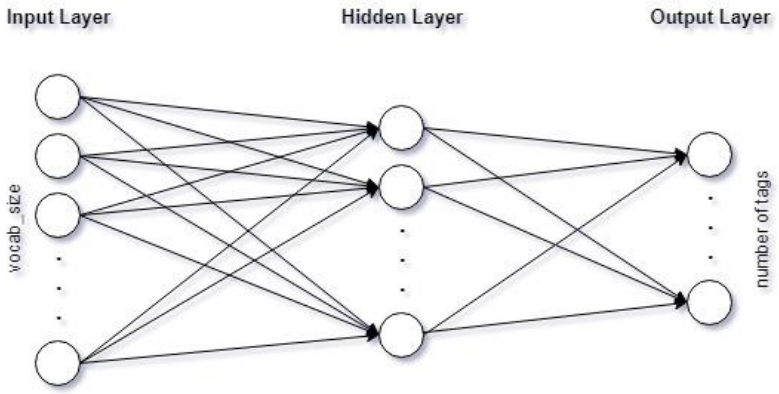


Figure 12: réseau de neurones

La fonction de combinaison est un simple produit scalaire entre le vecteur des entrées et le vecteur des poids synaptique, en plus d'un biais initialisé à 0. La fonction de combinaison est définie par :

$$f(x) = \mathrm{b} + \sum_{i=1}^{n} (W_i x_i)$$

La fonction d'activation, utilisée pour le passage de la première couche à la deuxième, est la fonction tangente hyperbolique. Le choix de cette fonction est fait suite à une comparaison des résultats donnés par 5 fonctions d'activation sur la base d'apprentissage initiale. Cette comparaison est basée sur le score F-mesure et est résumée dans le tableau suivant :

Tableau 2: comparaison de fonctions d'activation

Fonction d'activation	**linéaire**	**relu**	**softsign**	**sigmoid**	**tanh**
F-mesure	94.53%	94.93%	94.96%	84.26%	95.03%

La troisième couche est la couche de sortie et contient autant de neurones que de tags (5 dans notre cas). La fonction d'activation « Softmax » est utilisé afin d'avoir une probabilité que le paragraphe aie chacun des 5 tags, normalisée entre 0 et 1. La somme de toutes les 5 probabilités est égale à 1. Cette fonction est définie par :

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{pour tout j} \in \{1..K\}$$

Où K est la dimension du vecteur z.

Inconvénients de cette classification

Les inconvénients suivants sont ceux propres aux réseaux de neurones, indépendamment de la nature de nos données :

- Les résultats donnés par le réseau de neurones, quoi qu'ils soient bons, ne peuvent pas être expliqués concrètement.
- Détermination de l'architecture du réseau est complexe.

3.3.2. Support Vector Machine

Pour entraîner le classifieur SVM, on a choisi l'algorithme du gradient stochastique, puisqu'il est adapté aux problèmes où les données sont assez sparses, comme est le cas dans le traitement du langage naturel et Text Mining.

Cet algorithme met à jour, à chaque itération, un nombre de paramètres pour minimiser une fonction de perte donnée. Il est appelé « stochastique » parce qu'il utilise uniquement une partie du jeu de données d'entraînement, au lieu de tout le jeu de données, pour la mise à jour des paramètres. Il faut alors ajuster les paramètres suivants de la fonction SGDClassifier sous scikit-learn. Ces paramètres ainsi que les valeurs qu'on leur a choisies sont les suivants :

- La fonction de perte à minimiser (loss) : « hinge » pour dire que c'est pour le modèle SVM linéaire.
- Le terme de régularisation (penalty) : « L2 » pour dire que le terme est sous forme d'une norme L2 pour les coefficients des caractéristiques du jeu de données. Ce paramètre a pour but d'éviter le problème d' « overfitting ».
- Alpha : la constante par laquelle est multiplié le terme de régularisation.
- Nombre d'itérations de l'algorithme (n_iter) : 2 dans notre cas semble être suffisant.

Inconvénients de cette classification

Les inconvénients suivants sont ceux propres à l'algorithme SVM, indépendamment de la nature de nos données :

- Un grand nombre d'observations en entrée implique un calcul matriciel important.
- Plusieurs paramètres clés à définir correctement pour obtenir les meilleurs résultats de classification.

3.3.3. Naïve Bayes

Parmi les trois modèles du classifieur NB : Multinomial, Gaussien, et Bernoulli, on a opté pour le modèle Multinomial, vu que la variable à prédire est une variable à N classes. On entraîne le modèle en utilisant 70% des données.

On a utilisé la fonction GridSearchCV sous scikit-learn pour régler de manière optimale les paramètres suivants:

- alpha : la constante par laquelle est multiplié le terme de régularisation, à choisir entre 0.01 et 0.001
- ngram_range : si le vocabulaire contiendra que des unigrams ou aussi des bigrams et trigrams
- use_idf : si la matrice des entrées va inclure les fréquences tf ou tf-idf

Inconvénients de cette classification

Les inconvénients suivants sont ceux propres à l'algorithme SVM, indépendamment de la nature de nos données :

- Besoin d'un grand ensemble de données afin de pouvoir estimer de manière fiable la probabilité de chaque classe. L'utilisation Naïve Bayes avec un petit ensemble de données, entraîne des scores de précision et de rappel très faibles.

3.4. Méthodes d'entraînement

En effet, l'entraînement des classifieurs s'est fait en utilisant deux méthodes dans le but de comparaison :

- La méthode du holdout (HO) : avec comme entrée 70% des données pour l'entraînement et 30% des données pour le test.
- La méthode du cross validation (CV): en utilisant 5 partitions (5-fold cross validation), avec à chaque fois 20% des données pour le test et 80% des données pour l'entraînement.

Ainsi, pour comparer correctement les modèles, on a intérêt à utiliser les mêmes données (et par conséquent les mêmes caractéristiques) pour les entraîner. Cependant, la méthode d'entraînement du holdout utilisée partitionne les données en apprentissage/testing aléatoirement. La solution proposée est alors de partitionner manuellement une fois pour toutes.

On a alors partitionné de manière aléatoire les données en deux sous-ensembles : 70% des données pour l'entraînement et 30% des données pour le test. Cette même partition est ensuite utilisée pour entraîner toutes les méthodes.

Enfin, on utilise la méthode du cross validation pour faire le choix définitif des paramètres du modèle, ainsi que pour sélectionner le modèle optimal. La méthode du holdout nous sert, quant à elle, à avoir une idée sur la variation de performance entre les partitions. Si cette variance est très grande, alors le modèle sera peut-être peu stable.

3.5. Résultats

Cette section récapitule les résultats des classifications faites pour l'extraction automatique des cas de test à partir des documents de spécifications fonctionnelles. On utilisera ces résultats afin de sélectionner le modèle d'apprentissage le plus optimal à utiliser dans l'application.

Pour qualifier chaque modèle, en combinant une sélection donnée de variables avec une méthode d'apprentissage et une méthode d'entraînement, on utilise les scores de rappel, précision, et « F-mesure » calculés comme suit :

$$Precision = \frac{VP}{VP + FP} \qquad F - mesure = 2 * \frac{precision * recall}{precision + recall}$$

$$Recall = \frac{VP}{VP + FN}$$

N.B : Notre choix du modèle optimal se base sur le score F-mesure, puisqu'il combine les deux autres scores.

Les tableaux suivants montrent les résultats des trois techniques d'apprentissage. Les variables utilisées sont : tous les mots uniques du document, tous les mots uniques du document sans stopword, les verbes du document, et les verbes d'action du document.

A chaque fois qu'on sélectionne des variables, on y associe soit le nombre d'occurrences dans le paragraphe, soit le score TF-IDF.

3.5.1. Analyse des résultats de SVM

Le tableau des résultats du SVM est le suivant :

Tableau 3: résultats du SVM

Support Vector Machine							
		Cross validation			**Holdout**		
Variables	**Nombre de variables**	Precision	Recall	F-mesure	Precision	Recall	Fmesure
BOW+ TF*IDF	1943	0,90	0,87	0,89	0,89	0,91	0,89
BOW+ Occurrence		0,89	0,87	0,88	0,88	0,93	0,90
BOW_sans_stop_words+ Occurrence	1890	0,92	0,93	**0,92**	0,89	0,92	0,91
BOW_sans_stop_words+ TF*IDF		0,92	0,93	**0,92**	0,89	0,9	0,9
Verbe+ TF*IDF	618	0,78	0,77	0,78	0,75	0,77	0,76
Verbe+ Occurrence		0,79	0,83	0,81	0,76	0,85	0,80
Verbe_action+ TF*IDF	165	0,75	0,49	0,59	0,72	0,49	0,58
Verbe_action+ Occurrence		0,70	0,48	0,57	0,71	0,49	0,58

Le score F-mesure est optimal avec la méthode SVM quand le dictionnaire contient tous les mots du document mais sans stopwords, avec un score de 92%. Quelle que soient les valeurs dans la matrice document terme (tf-idf ou nombre d'occurrence), le score est le même.

Cela peut être expliqué par le fait que dans notre cas, quand les stopwords sont enlevés, alors les mots restants sont suffisants pour pouvoir caractériser chaque catégorie de paragraphe. Chaque catégorie est peut-être caractérisée par un ensemble de mots, et ces ensembles de mots ne sont pas très chevauchants.

On remarque que la classification du texte est toujours aussi précise avec les verbes qu'avec les verbes d'action (un peu moins avec les verbes d'action), mais le score de rappel s'est beaucoup dégradé avec les verbes d'action. Cela signifie que le SVM n'arrive passe à côté de beaucoup plus de paragraphes non classifiés, en se basant sur les verbes d'action.

On remarque également que la performance du classifieur s'est dégradée en utilisant les verbes par rapport à tous les mots du document. Cependant, on peut quand même assumer d'opter pour cette sélection de variables (les verbes), et gagner en temps d'apprentissage.

Dans tous les cas, la variance entre les scores obtenus avec cross validation et avec le holdout n'est pas assez grande.

Analyse des résultats de Naïve Bayes

Le tableau des résultats du Naïve Bayes est le suivant :

Tableau 4: résultats de Naïve Bayes

Naïve Bayes							
		Cross validation			Holdout		
Variables	**Nombre de variables**	Precision	Recall	F-mesure	Precision	Recall	Fmesure
BOW+ TF*IDF	1943	0,85	0,54	0,66	0,89	0,51	0,64
BOW+ Occurrence		0,81	0,89	**0,85**	0,80	0,89	0,84
BOW_sans_stop_words+ Occurrence	1890	0,85	0,86	**0,85**	0,81	0,88	0,84
BOW_sans_stop_words+ TF*IDF		0,75	0,49	0,59	0,77	0,52	0,62
Verbe+ TF*IDF	618	0,67	0,33	0,44	0,60	0,33	0,43
Verbe+ Occurrence		0,78	0,55	0,65	0,79	0,55	0,65
Verbe_action+ TF*IDF	165	0,58	0,32	0,41	0,68	0,32	0,44
Verbe_action+ Occurrence		0,68	0,41	0,51	0,67	0,39	0,49

Pour Naïve Bayes, le meilleur score F-mesure est de 85% avec les variables étant soit avec tous les mots soit avec tous les mots sauf stopwords, sans fréquence TF-IDF à chaque fois.

La suppression des stopwords dans ce cas n'a aucun effet sur la mesure F-mesure, mais la classification est un peu plus précise (score precision augmente de 4%) quand on enlève stopwords.

On remarque que la mesure F-mesure est meilleur en utilisant le nombre d'occurrences, et non pas le score TF-IDF. En effet, ceci peut être expliqué par le fait que dans notre cas, il existe certains mots qui, plus ils sont fréquents, aident à caractériser nos catégories de paragraphes. Le score TF-IDF, en leur donnant moins une valeur plus petite, les rend moins indicatifs.

Comme pour SVM, la variance entre les scores obtenus avec cross validation et avec le holdout n'est pas assez grande.

Analyse des résultats de réseau de neurones

Le tableau des résultats du réseau de neurones est le suivant :

Tableau 5: résultats du réseau de neurones

Réseau de Neurones							
		Cross validation			Holdout		
Variables	**Nombre de variables**	Precision	Recall	F-mesure	Precision	Recall	Fmesure
BOW+ TF*IDF	1943	0,91	0,88	0,88	0,91	0,90	0,90

BOW+ Occurrence		0,91	0,88	**0,89**	0,90	0,92	0,91
BOW_sans_stop_words+ Occurrence	1890	0,91	0,88	**0,89**	0,91	0,91	0,91
BOW_sans_stop_words+ TF*IDF		0,86	0,9	0,88	0,9	0,9	0,9
Verbe+ TF*IDF	618	0,8	0,84	0,8	0,76	0,85	0,8
Verbe+ Occurrence		0,8	0,84	0,8	0,78	0,86	0,82
Verbe_action+ TF*IDF	165	0,76	0,60	0,7	0,78	0,61	0,68
Verbe_action+ Occurrence		0,77	0,61	0,7	0,79	0,61	0,69

Le meilleur score F-mesure pour le réseau de neurones est de 89%. Comme pour SVM et Naïve Bayes, l'entraînement en utilisant le dataset avec tous les mots sans stopwords et avec les nombres d'occurrences donne les meilleurs résultats.

Ce qui fait la différence avec les deux autres techniques d'apprentissage, c'est que le réseau de neurones est long durant l'entraînement, mais donne de bons résultats presque pour toutes les sélections de variables. On peut ici se permettre de choisir le dataset avec verbes, avec moins de 1% de moins en score F-mesure, mais avec uniquement 618 variables.

Choix du modèle optimal

Pour choisir le modèle optimal pour la classification des paragraphes des spécifications fonctionnelles à utiliser dans le projet, on compare les meilleurs scores F-mesure obtenus lors de l'entraînement. Le tableau suivant récapitule ces scores :

Tableau 6: meilleur score par méthode d'apprentissage

	SVM	**Naïve Bayes**	**Réseau de neurones**
F-mesure optimal	**0.92**	0.85	0.89

Le modèle choisi en se basant sur le tableau ci-dessus est donc SVM, étant entraîné sur le dataset avec tous les mots sans stopwords.

N.B : Pour que le modèle soit prêt à être utilisé dans le projet, on l'a ré entraîné avec le dataset tout entier.

Chapitre V : Conclusion

1. Bilan du travail effectué

1.1. Forces de l'étude

A l'issue de ce stage, les missions confiées ont été accomplies avec succès, à savoir :

- Proposition d'un modèle standard de spécifications fonctionnelles
- Utilisation de techniques de NLP pour le remplissage du modèle standard
- Hybridation NLP et apprentissage automatique pour extraire les cas de test à partir des spécifications fonctionnelles
- Test de plusieurs méthodes d'apprentissage, et choix du modèle optimal
- Participation au développement de l'application Web pour l'extraction des cas de tests à partir de spécifications fonctionnelles

L'utilisation des méthodes d'apprentissage dans le cadre de ce projet est pertinente, puisqu'elle a permis d'extraire les informations pertinentes des documents, sans tenir compte de leur spécificité syntaxique.

1.2. Limites de l'étude

Durant le stage, il y a eu un manque de données puisqu'uniquement un seul document de spécifications fonctionnelles nous a été fourni. Par conséquent, le modèle d'apprentissage construit en se basant sur ce document est performant quand il s'agit de la classification des paragraphes d'un autre document similaire. Mais la classification sera certainement beaucoup moins performante pour des documents ayant une structure différente ou décrivant une application d'un domaine différent.

Ceci est dû au fait que moins le modèle aurait vu différents mots de différents domaines durant apprentissage, moins il sera capable de caractériser les paragraphes de chaque classe. Avec plus de données, le modèle aurait alors été capable de classifier des documents de domaines différents et avec plus de précision.

1.3. Perspectives

Si l'étude poursuivait dans le cadre de ce projet, on aurait fait de sorte que le modèle d'apprentissage construit puisse apprendre et s'enrichir de manière autonome au fur et à mesure de voir des documents. Ceci peut être fait avec l'intervention des utilisateurs de l'application, qui indiqueront lorsque le modèle aurait mal classifié les paragraphes des documents.

Parmi les perspectives, on peut également penser à construire différents modèles pour différents domaines d'applications. En utilisant le « topic modeling », on peut ensuite reconnaître le domaine

du document de spécifications fonctionnelles, et appliquer le modèle correspondant. Ceci peut être pertinent pour élargir l'utilité de l'application automatisant le processus des projets de test.

2. Bilan personnel

2.1. Difficultés

impossibilité d'automatisation entière, intervention humaine nécessaire ?

2.2. Apports

En plus de l'application concrète des connaissances acquises en Apprentissage Automatique, ce stage nous a également permis de s'ouvrir sur le domaine du Traitement du Langage naturel ; ce qui peut faire une éventuelle orientation de carrière.

Références

[1] Douglas Homan. Test automation architectures : Planning for test automation. Software Quality Methods, LLC., 1999.

[2] Kai P. Dudekula M. R., Katam R. K. M. Benets and limitations of automated software testing : Systematic literature review and practitioner survey. IEEE Press Piscataway, NJ, USA, 2012.

[3] Selenium website. https://www.seleniumhq.org/.

[4] CertifyIt website. http://www.smartesting.com/fr/certifyit/.

[5] Panaya website. https://www.panaya.com/.

[6] UFT website. https://software.microfocus.com/fr-fr/products/unified-functional-automated-testing/features

[7] Ranorex website. https://www.ranorex.com/.

[8] Helmut Schmid. Treetagger - a language independent part-of-speech tagger. 1995.

Annexe 1 : modèle standard pour les projets en mode Agile

Tableau : modèle standard pour les projets en mode Agile

ID US	Situation	Périmètre	RG ID	Scénario ID	Zone ID	Input	action	Valeur associée à l'action	Output
CF001	Page informations personnelles- sous les informations générales et au-dessus du changement de mot de passe	Site web desktop	CART 01-01	CF001-SC1	CART 01, CART 03	saisit dans le champ « CART 01 » un nombre de caractères égal à « 15 »	Clique sur le bouton valider « CART 03 »		le message « votre modification est prise en compte » s'affiche sous le bouton « CART03 »
			CART 01-01	CF001-SC2	CART 01, CART 03	saisit dans le champ « CART 01 » un nombre de caractères différent de « 15 »	Clique sur le bouton valider « CART 03 »		le message d'erreur « veuillez vérifier vos informations » s'affiche sous le bouton « CART03 »

Annexe 2 : démonstration de l'application web

L'application web développée contient deux modules principaux : la gestion des utilisateurs, et la gestion des projets. On démontre dans cette annexe la partie de gestion des projets, puisque c'est la plus pertinente pour le stage.

La première interface est celle de connexion. (Figure 13)

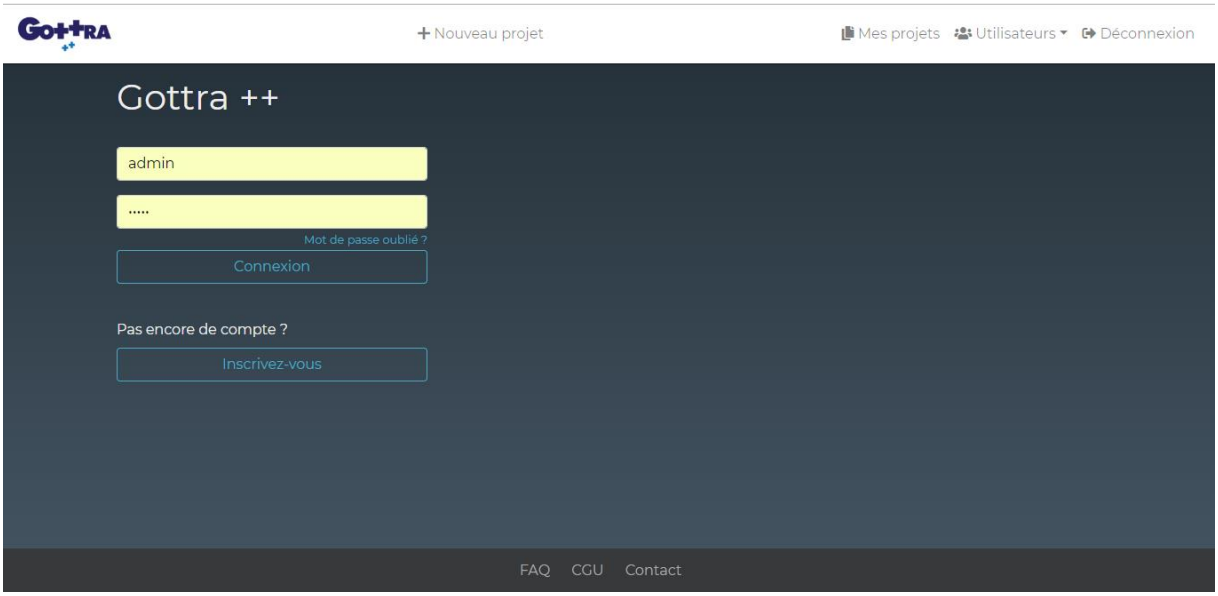


Figure 13: Interface de connexion

Une fois connecté, l'utilisateur peut visualiser les projets existants ou, selon son rôle, gérer les utilisateurs. (Figure 14)

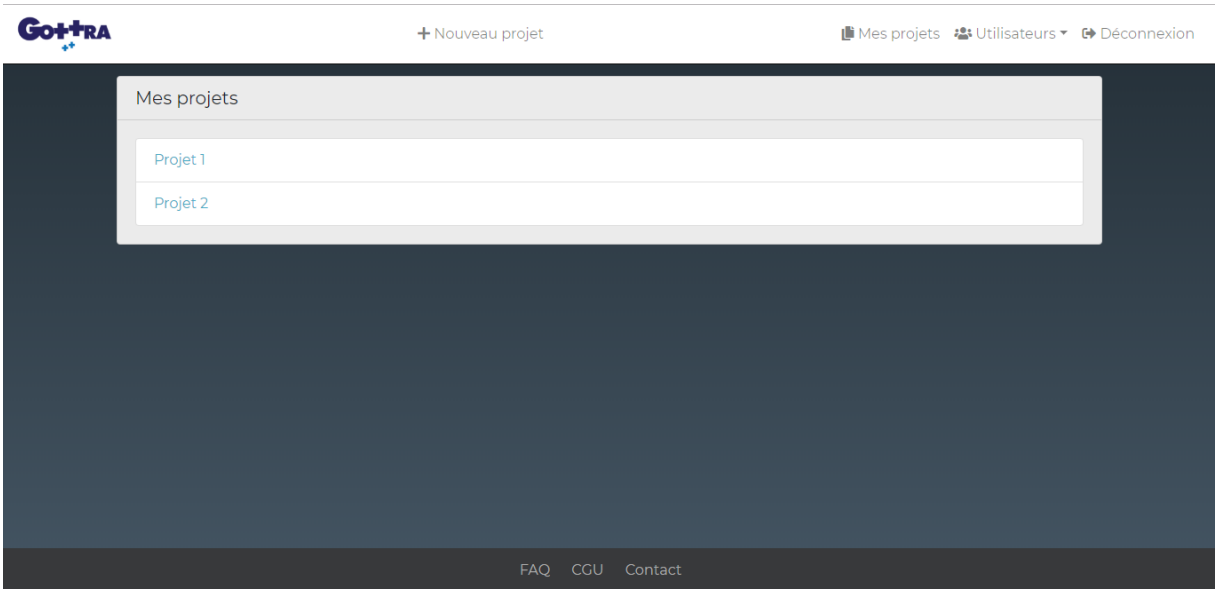


Figure 14: Interface d'accueil

En cliquant sur "Nouveau Projet", l'utilisateur peut créer un nouveau projet en entrant un nom et en choisissant le type de projet : Agile ou Standard (Cycle en V). (Figure 15)

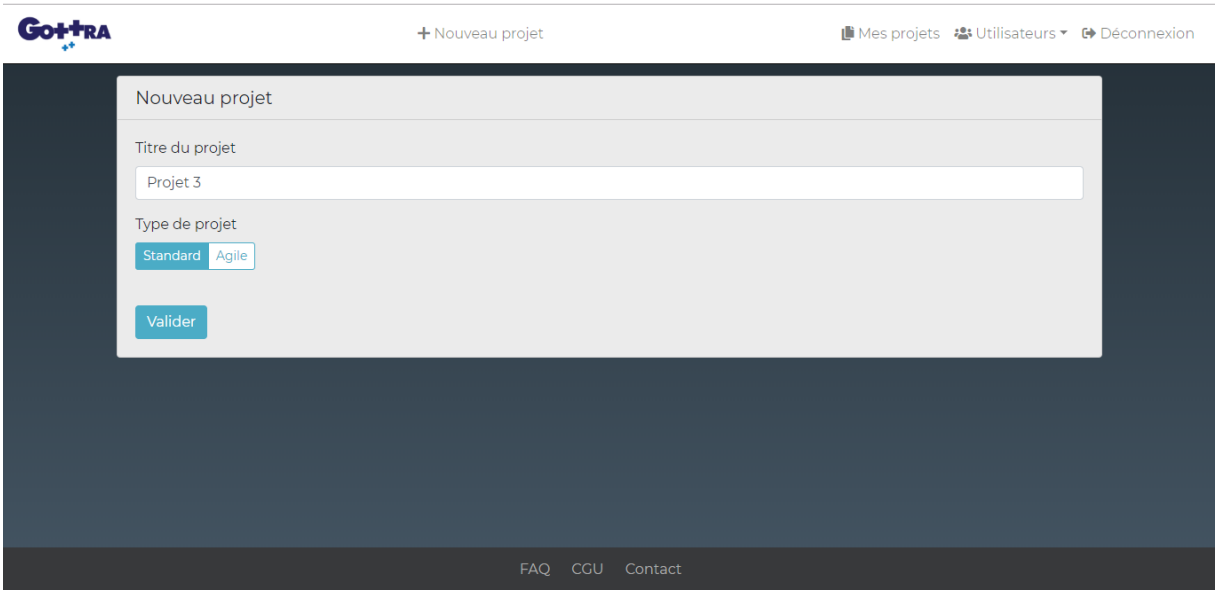


Figure 15: Interface de création d'un nouveau projet

Lorsque le projet est créé, les fichiers de spécifications fonctionnelles lui appartenant peuvent être chargés. (Figure 16) C'est à partir de ces fichiers que les scénarii et cas de tests seront extraits.

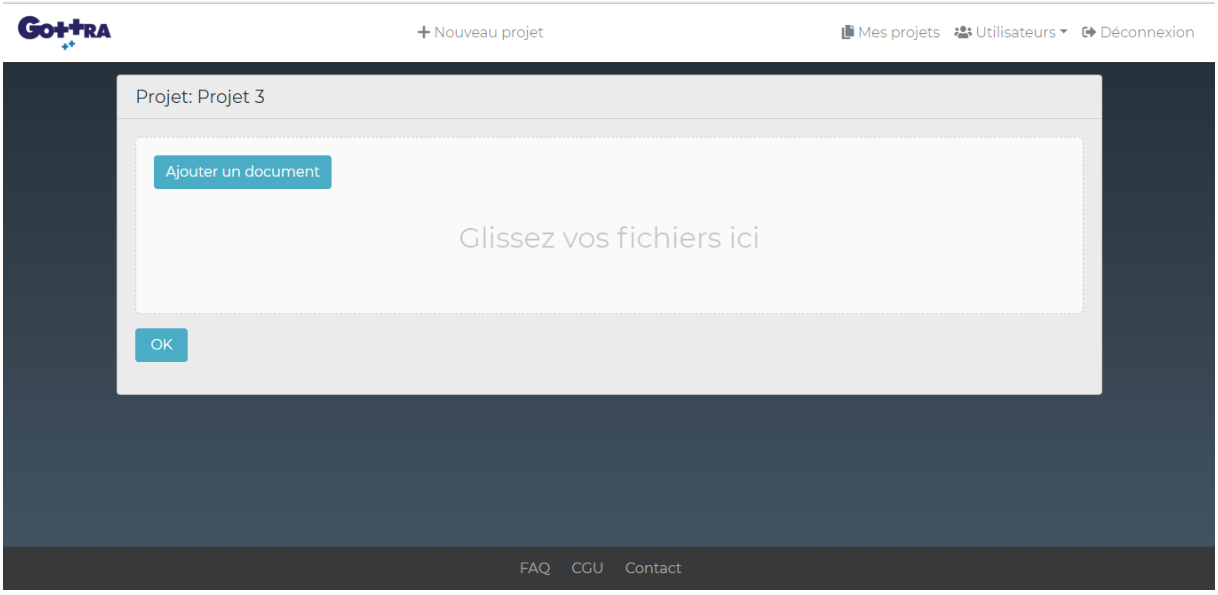


Figure 16: Interface des documents d'un projet

Prenons comme exemple un fichier test (Figure 17), contenant un seul cas d'utilisation. On va charger ce fichier dans le projet, et attendre qu'il soit traité. (Figure 18)

```
4.1.2   CU001-002 Afficher le bouton de langue

Acteur : Internaute
Cas d'utilisation permettant à un internaute  de consulter le bouton de langue.

•   Notes :
    Le système affiche :
    la langue sélectionnée
    La langue sélectionnée par défaut est la langue française (FR)

•   Actions :
    Au clic sur la langue sélectionnée, le système affiche les autres langues
    Si la langue sélectionnée est français, le système affiche
•   En deuxième langue,  le bouton Anglais (EN) à la place du bouton réseau social Twitter
•   En troisième langue, le bouton Espagnol (ES) à la place bouton réseau social Facebook
•   Le bouton +  (voir cas d'utilisation Afficher la page Langue)
    Si la langue sélectionnée est anglais, le système affiche
•   En deuxième langue,  le bouton Francais (FR) à la place du bouton réseau social Twitter
•   En troisième langue, le bouton Espagnol (ES) à la place bouton réseau social Facebook
•   Le bouton +  (voir cas d'utilisation Afficher la page Langue)
    Si la langue sélectionnée est espagnol, le système affiche
•   En deuxième langue,  le bouton Francais (FR) à la place du bouton réseau social Twitter
•   En troisième langue, le bouton Anglais (EN) à la place bouton réseau social Facebook
•   Le bouton +  (voir cas d'utilisation Afficher la page Langue)
    Au clic sur la deuxième ou troisième langue, le système réaffiche  les boutons réseaux sociaux et le bouton + est caché


•   Règles de gestion :
    Si l'utilisateur clique sur Anglais / Espagnol depuis une page française :
    Si la page est traduite, celle-ci est affichée.
    Si la page n'est pas traduite
•   le système redirige vers la page parente traduite de niveau N-1
•   Si la page parente de niveau N-1 n'est pas traduite, le système redirige vers la page parente traduite de niveau N-2
•   Si la page parente de niveau N-2 n'est pas traduite, le système redirige vers la page d'accueil
```

Figure 17: Exemple de fichier à classifier

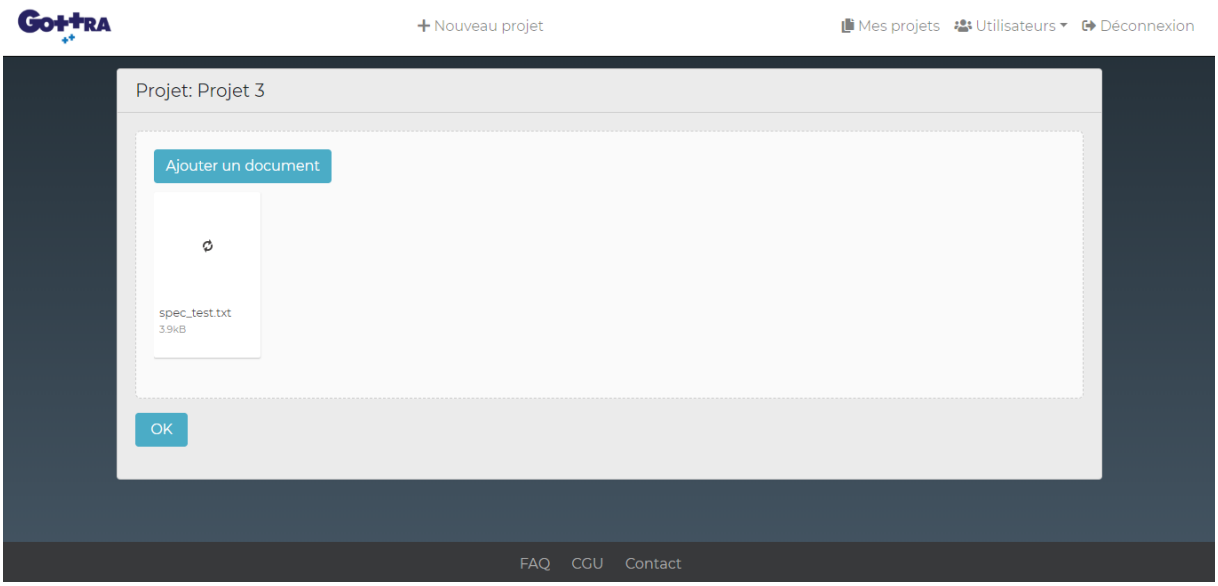


Figure 18: Interface de chargement d'un nouveau document

Une fois le traitement fini, on clique sur "Ok", et on peut visualiser les cas d'utilisation extraits à partir du document (Figure 19). Le développement de l'application n'étant pas encore fini, il reste encore à visualiser les scénarii et cas de tests correspondants. Cependant, ceux-ci sont bien extraits et insérés dans la base de données.

Figure B.7

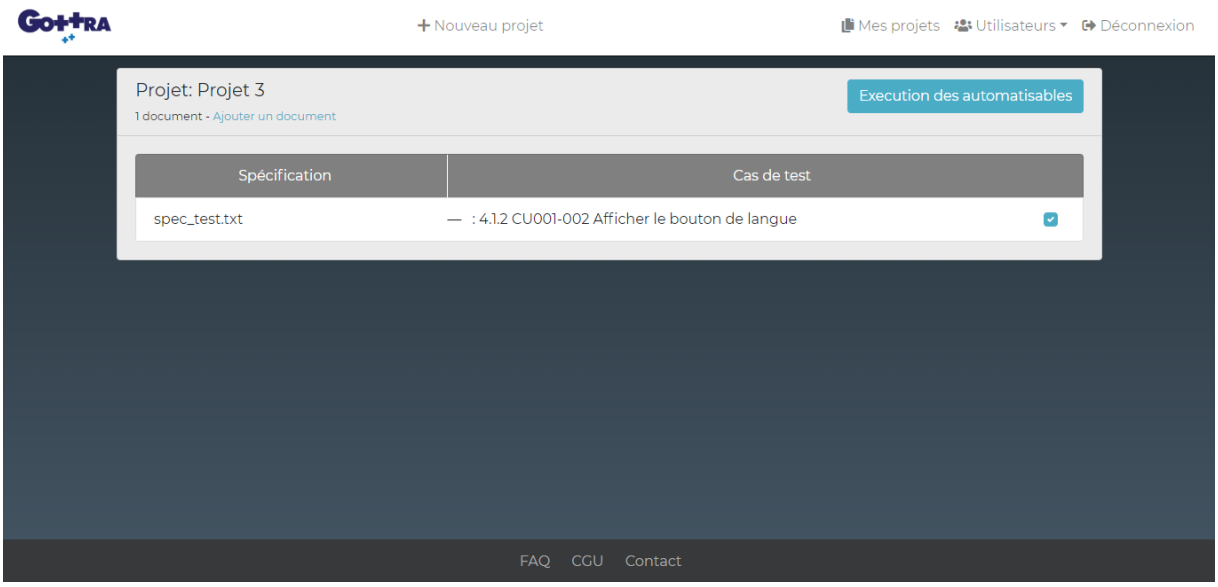


Figure 19: Interface du résultat de la classification