

Mémoire de PFE

Formation FIL

IMT Atlantique



Amélioration des performances de détection automatique d'anomalies dans des logs applicatifs

Oscar Gloaguen

Direction Générale des Finances Publiques



Table des matières

| | |
|----------------------------------------------------------|------------|
| Table des matières | ii |
| Remerciements | iii |
| Résumé | iv |
| Introduction | 1 |
| 1 La DGFIP, administration en évolution constante | 2 |
| 2 L'existant : le projet analyse-logs | 4 |
| 2.1 Contexte du projet | 4 |
| 2.2 Que sont les logs ? | 5 |
| 2.3 Le modèle original : DeepLog | 5 |
| 2.4 Evolutions du projet par Léa | 8 |
| 3 Le processus scientifique au cœur du projet | 10 |
| 3.1 État de l'art de l'analyse de logs | 10 |
| 3.2 Analyse des jeux de données disponibles | 10 |
| 3.3 Agilité et recherche : une union difficile | 10 |
| 4 Développement du nouveau projet | 11 |
| 5 Projection de l'impact du projet | 12 |
| 5.1 Gain de temps et de productivité | 12 |
| 5.2 L'humain contre l'algorithme | 12 |
| Conclusion | 13 |
| Bibliographie | |
| Glossaire administratif | |
| Glossaire technique | |
| Annexes | |

Remerciements

Je souhaite remercier la Direction Générale des Finances Publiques et tout particulièrement Olivier Blanc, pour m'avoir permis d'effectuer mon alternance dans cette équipe et pour m'avoir accompagné tout au long jusqu'au PFE.

Je tiens a remercier personnellement Robin Gries, qui a su être un vrai coéquipier tout au long de ce projet malgré sa complexité.

Je voudrais aussi remercier IMT Atlantique ainsi que ses professeurs, et surtout mon tuteur pédagogique Thomas Ledoux qui s'est mis a ma disposition pour le PFE.

Résumé

Les **logs** sont une source importante de données détaillant le fonctionnement interne d'une application, mais ne sont pourtant que rarement utilisés à leur plein potentiel. Dans ce mémoire, je vais détailler le processus d'évolution d'un outil d'**apprentissage automatique** qui utilise les **logs** pour détecter et même tenter de prévoir des anomalies logicielles. Le projet s'apparentant plus à un projet de recherche, la démarche scientifique sera détaillée, ainsi que les caractéristiques techniques et le déroulement de l'implémentation. L'organisation du projet avec un stagiaire et moi-même sera développée, ainsi que ses impacts humains et économiques.

Abstract

Logs are an important source of data when it comes to the internal workings of software, but they are rarely used to their full potential. In this memoir, I will explain the evolution of a machine learning tool which uses **logs** to detect and even attempt to predict software anomalies. The project being similar to a research project, the scientific protocol will be detailed, as well as the technical characteristics and the course of the implementation. Project management with an intern and myself will be developed, as well as the human and economic impacts of the project.

Mots-clés traitement automatique du langage (TAL), apprentissage automatique, apprentissage profond, analyse de logs applicatifs

Introduction

1-2 pages

Le projet analyse-logs a été développé par deux précédents apprentis de la [Direction Générale des Finances Publiques \(DGFIP\)](#), Rémi puis Léa. Cependant, les performances des algorithmes implémentés n'étant pas satisfaisantes, c'est ici que naît le sujet de ce PFE. Ce mémoire détaille le processus d'évolution de ce projet dans l'objectif d'amélioration des performances.

J'ai travaillé sur cette problématique en binôme avec Robin, un stagiaire en dernière année de master. Ce document touchera aussi sur l'organisation du sujet entre nous ainsi que les bénéfices et difficultés à travailler en équipe.

Ce sujet de PFE s'intègre dans la stratégie d'innovation du SI de la [DGFIP](#). L'objectif est de montrer l'efficacité de ces outils, pour mettre en valeur l'innovation et pousser leur utilisation au sein des bureaux. Dans ce cadre, une structure proche de celle d'un projet de recherche a été suivie. Pour cela, nous avons d'abord composé et étudié un état de l'art des algorithmes de détection d'anomalies existants. Ils utilisent pour la plupart des méthodes d'[apprentissage automatique](#), avec des algorithmes classiques ou de l'[apprentissage profond](#).

Chapitre 1

La DGFIP, administration en évolution constante

La DGFIP est une administration française née de la fusion de la Direction Générale des Impôts (DGI) et de la Direction Générale des Comptes Publics (DGCP) en 2008. Elle hérite alors des missions des deux entités, en faisant un service public très étendu, en charge notamment de la collecte des impôts et taxes et de la législation fiscale.

Cette administration possède une hiérarchie forte séparée en 8 services, qui définit leur domaine de travail, ainsi que différentes directions (voir organigramme 1 en annexes). Une majorité des services sont des services métiers, directement en lien avec les missions de la DGFIP, mais 3 de ces services sont des services support, ou *transverses*. Ces derniers sont le service des Ressources Humaines, le service de Stratégie, Pilotage et Budget, et le Service des systèmes d'Information (SI). Malgré une interaction indirecte avec le domaine métier des finances publiques, ils répondent aux besoins de la DGFIP en permettant le bon fonctionnement des autres services, ou même en améliorant leur performance.

Comme toute grande structure aujourd'hui, la DGFIP possède un besoin très fort en technologies de l'information, qui est rempli par le SI. Ce service est indispensable, car de nombreuses missions de la DGFIP reposent sur des programmes (e.g., calcul et déclarations d'impôts et des taxes) permettant de traiter de larges quantités de données en un temps restreint. Les premières versions de ces applications datent des années 80, et ont pour la plupart évolué et sont restées utilisées jusqu'à aujourd'hui. L'informatique est donc au centre de cette administration, autant pour les agents en interne que pour les utilisateurs externes.

Le SI est séparé en de nombreux bureaux (voir annexe 2). Il comprend lui-même des bureaux *transverses* qui facilitent le bon fonctionnement des autres bureaux. Le reste des bureaux est regroupé sous la Direction des projets numériques (DPN), et sont chargés du pilotage, du développement et de la maintenance d'applications d'un domaine précis.

Cette nouvelle hiérarchie date de 2021, où une réorganisation a eu lieu. En effet, la

CHAPITRE 1. LA DGFIP, ADMINISTRATION EN ÉVOLUTION CONSTANTE

DPN n'existait pas avant cela, et les bureaux étaient regroupés sous deux directions, "étude et développement" et "production". Le SI comporte aujourd'hui plus de bureaux, qui sont donc plus spécialisés, avec par exemple des bureaux en charge d'une unique mission importante.

Le Bureau du SI des professionnels (BSI-3) est un bureau de la DPN chargé de la fiscalité des professionnels, dirigé par Alain Kerdoncuff. Il a à sa charge une dizaine d'applications qu'il spécifie, développe et maintient. L'une d'entre elles est MEcanisation Des Opérations Comptables (MEDOC) qui est une application d'encaissement d'impôts et de gestion de comptabilité de l'État. Le BSI-3 possède une mission particulière de modernisation de cette application, tâche très complexe étant donné son échelle et sa complexité.

Ce bureau était auparavant nommé SI-1C, et ses missions n'ont pas changé avec la réorganisation. Cependant, chacun des bureaux sont encore spécialisés en plusieurs divisions, chacune en charge de projets spécifiques. Ces divisions ont-elles changé avec la réorganisation, notamment une en particulier qui a été supprimée, la Division technique transverse (DTT). Cette dernière, chapeautée par Olivier Blanc (aussi mon tuteur) était à la fois une aide technique sur le domaine du logiciel, ainsi qu'une division détachée des projets principaux, permettant de mettre en avant des technologies innovantes et des projets expérimentaux.

Parler de la cellule innovation ?

Détailler avec des chiffres

Chapitre 2

L'existant : le projet analyse-logs

2.1 Contexte du projet

Si le premier chapitre peut mettre quoi que ce soit en avant, c'est bien le besoin très fort en informatique de la DGFIP, rempli par le SI et ses très nombreuses applications. Mais un parc applicatif si étendu pose un problème majeur : celui de la maintenance. En effet, la maintenance logicielle est un processus très coûteux, surtout en termes d'heures de travail de personnes qualifiées. Ce sont les développeurs des applications à maintenir qui doivent effectuer cette maintenance, car c'est eux qui connaissent le fonctionnement interne de l'application.

Une solution serait de former des équipes de maintenance au différents logiciels. Malheureusement, cela demanderait encore plus de moyens, à la fois prenant encore du temps aux développeurs, mais aussi nécessitant d'embaucher des personnes qui travailleraient à temps plein sur la maintenance, ce qui n'est pas envisageable. Il n'est même pas donné que cela libère vraiment du temps aux développeurs, étant donné l'évolution constante des logiciels et les formations supplémentaires qu'il faudrait donner pour tenir une équipe de maintenance à jour.

Il y a 4 ans, mon tuteur Olivier Blanc s'est penché sur la question. Le vrai problème était bien de faire gagner du temps aux développeurs en facilitant et accélérant la maintenance d'une application. Dans le cas d'une erreur dans l'application qui stopperait partiellement ou complètement son fonctionnement, sa correction est obligatoire pour les équipes. Le problème de retrouver la source de cette erreur et de la corriger est alors aussi important, et c'est ici que rentrent en jeu les **logs** applicatifs (voir partie suivante). Olivier mit en avant le papier de recherche DeepLog [2], dont le but était d'utiliser les logs pour détecter et prédire les erreurs, ainsi que remonter vers leur cause initiale. Le projet analyse-logs est alors né, commençant comme sujet de PFE de Rémi Grison, apprenti à la DGFIP en 2019. Léa Lebert a ensuite succédé à Rémi, puis un an après le départ de Léa j'ai repris le projet.

2.2 Que sont les logs ?

Le terme “log” est tiré de l’anglais et signifie à l’origine “journal”, tel un journal de voyage. Il décrit donc un document contenant une liste chronologique d’évènements datés, qui peuvent être utilisés pour reformer l’histoire de ce qui s’est passé. Ils sont utiles dans le cas d’un accident, comme par exemple les journaux décrivant les voyages d’explorateurs ou aujourd’hui les boîtes noires dans l’aviation.

Les logs informatiques découlent de cette définition, décrivant le chemin d’exécution d’une application, d’un site, ou même d’un système d’exploitation. Ces fichiers de logs sont séparés en lignes de logs (souvent appelée un “log”) qui contiennent des informations communes à chaque ligne, ainsi que le message de log qui est la partie variable. Les informations communes vont en général être au minimum la date, l’heure et le niveau du log. Le niveau représente la gravité de l’information présentée, en général décomposée de cette façon :

0. DEBUG : Log utile pour déboguer l’application, qui sera en général utilisé par le développeur. Ce log ne devrait pas être trouvé dans des fichiers de logs d’un logiciel en production.
1. INFO : Information sur le statut fonctionnel d’une application, qui peuvent être utiles et compréhensibles d’un point de vue métier.
2. WARNING : Anomalie logicielle non bloquante pour l’application, peut être fonctionnelle ou logicielle mais ne nécessite pas d’action immédiate.
3. ERROR : Anomalie logicielle bloquante, souvent un bloc logiciel qui cesse de fonctionner. Nécessite en général une intervention assez rapide, pour éviter d’autres erreurs au sein du système.

Grâce à ces informations, un développeur (ou quelqu’un chargé de la maintenance) sera capable de retrouver les anomalies qui ont eu lieu ainsi que de remonter les causes de ces anomalies. C’est un processus en général assez manuel, qui nécessite souvent de comparer les logs et le code source, et où la recherche par mot clé peut s’avérer très utile.

Déplacer dans un chapitre ‘notions’ ?

Ajouter une section sur les réseaux de neurones ?

2.3 Le modèle original : DeepLog

Le travail de Rémi qui a démarré ce projet a principalement été l’implémentation de l’algorithme DeepLog à partir du papier de Du et al. [2]. Je vais ici détailler le fonctionnement de DeepLog ainsi que l’implémentation qui a été faite par Rémi.

La structure complète du modèle est détaillée dans la figure 2.1, tirée directement du papier. Une étape qui n’est pas détaillée ici est celle de la récupération et du regroupement des logs, qui n’est pas aussi triviale qu’elle pourrait en avoir l’air. Cependant, ce

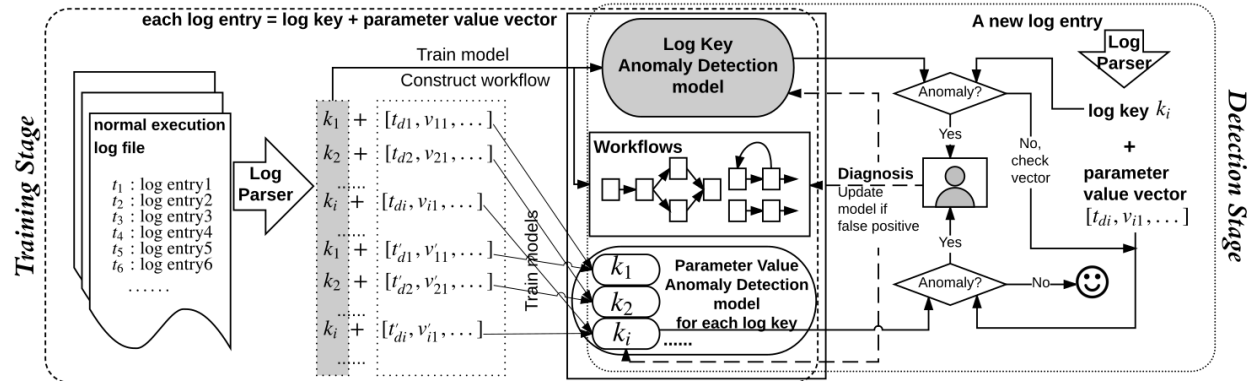


FIGURE 2.1 – Structure de DeepLog

point ne sera pas plus approfondi dans ce rapport étant donné que nous ne l'avons pas vraiment traité.

Parsing

La première étape est celle du “parsing” (que l’on pourrait approximativement traduire par analyse syntaxique en français). Elle est appliquée à un fichier de **logs** pour le traduire en une liste d’ids et de vecteurs de paramètres. Par exemple, avec les **logs** suivant :

```
Session ouverte id 123
Session ouverte id 456
Session fermée id 123
Session fermée id 456
```

On s’attend à obtenir un tel résultat parsé :

```
1, [123]
1, [456]
2, [123]
2, [456]
```

La méthode utilisée par DeepLog pour produire ces résultats est Spell [1], développée l’année précédente par une partie des chercheurs qui ont travaillé sur DeepLog. Seulement, entre la publication du papier et l’implémentation par Rémi, 2 années sont passées, et l’état de l’art avait changé en faveur de l’algorithme Drain [3]. Cet algorithme avait été publié la même année que DeepLog, et est encore aujourd’hui une solution de choix pour le parsing de **logs**. Rémi a donc produit sa propre implémentation de l’algorithme Drain pour le projet. Il existe aujourd’hui une très bonne implémentation open source de drain en Python, qui est donc plus robuste et performante que l’implémentation de Rémi, et nous allons donc la réutiliser plus tard.

Analyse des suites

L'étape d'analyse de DeepLog est séparée en deux parties. Les chercheurs de DeepLog avaient pour but de créer un modèle plus puissant que ce qui a pu exister auparavant. Un des problèmes de ces anciens modèles est l'absence de prise en compte de la temporalité, les **logs** étant seulement analysés un par un sans prendre en compte leur ordre d'apparition. L'analyse de l'enchaînement des **logs** est donc très importante pour mieux comprendre certaines erreurs moins évidentes à premier abord.

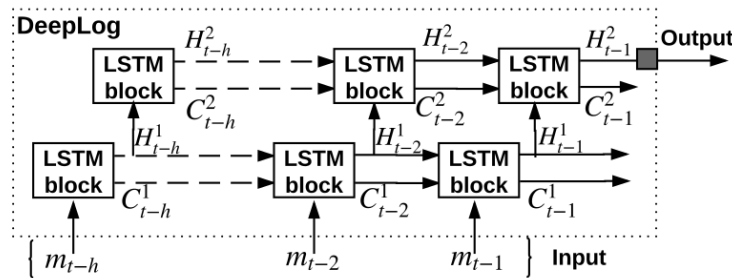


FIGURE 2.2 – Structure d'un réseau LSTM empilé

L'analyse temporelle est faite par DeepLog à l'aide de réseaux LSTM empilés (voir figure 2.2). Un LSTM est un type de réseau de neurones en apprentissage profond, qui possède des connexions de rétroaction qui lui permettent de capturer des comportements sur une durée relativement longue.

Les ids des **logs** sont utilisés, et on entraîne le modèle à prédire les g **logs** les plus probables qui apparaîtraient après une suite de h **logs**, à l'aide d'une fenêtre glissante. Par exemple, pour cet enchaînement d'ids : 1, 2, 1, 3, 4, 1, 2, avec une taille de fenêtre $h = 5$, on va envoyer 1, 2, 1, 3, 4 au réseau, et on s'attend à voir 1 au sein des g **logs** prédits. Ensuite on enverrait 2, 1, 3, 4, 1 en attendant 2, etc.

Cet entraînement pourra ensuite être utilisé durant l'analyse d'un fichier de **logs**, ou même durant son exécution. On prévoit les g ids les plus probables, puis on compare avec le **log** qui est vraiment à la suite. Si son id n'est pas dans les prédictions de l'algorithme, on considère ce **log** comme anormal.

Analyse des paramètres

Le choix des créateurs de DeepLog a été de considérer chaque partie variable des **logs** (soit chaque paramètre) comme sa propre série temporelle à travers les **logs** de même id. En faisant cela, il est possible de réutiliser la même architecture de LSTM empilés détaillés à la figure 2.2. Cette fois, on attend une unique prédiction de valeur, et les lignes sont considérées comme anormales si la valeur observée est trop différente de la valeur réelle.

2.4 Evolutions du projet par Léa

Suite à l'implémentation de DeepLog par Rémi, le projet était certes fonctionnel mais très peu utilisable par un utilisateur lambda. En effet, l'utilisation se faisait à travers une ligne de commande comportant de nombreux paramètres, celle-ci étant la seule interface pour les utilisateurs.

Le travail de Léa a donc été d'améliorer ce projet pour le rendre plus simple d'utilisation, mais aussi d'agrandir l'éventail de possibilités d'analyse proposé. Il a été choisi de créer une nouvelle façon de faire l'interface avec le modèle via la bibliothèque Blockly créée par Google, détaillée à la suite. Je ferai aussi une explication des autres algorithmes mis en place, sur quoi j'avais aidé Léa au moment de son PFE.

Blockly

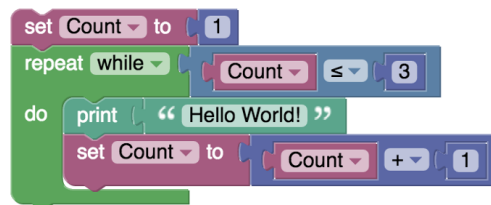


FIGURE 2.3 – Exemple de programme avec Blockly


Blockly est une librairie publiée par Google, permettant de créer des éditeurs de code visuels. L'utilisateur relie des blocs qui représentent des instructions, des variables, des boucles (voir figure 2.3) et cette représentation est traduite en code, de façon entièrement configurable par un développeur. Si vous avez déjà eu affaire au moteur de jeu en ligne Scratch, son éditeur visuel est basé sur Blockly.

Olivier et Léa ont fait le choix d'utiliser cette librairie pour représenter les différentes structures de modèles, ou "pipelines". En effet, si on considère chaque bloc du modèle, il est possible de les remplacer par d'autres blocs équivalents, par exemple un autre analyseur de modèle. Blockly permet très bien cette représentation, avec des couleurs et des formes qui correspondent aux types des éléments. Chaque étape du modèle pourrait alors avoir son type de bloc, et l'interface serait visuellement très intuitive.

L'interface Blockly a donc été développée par dessus le travail de Rémi, mais aussi avec les autres algorithmes détaillés dans la partie suivante. Elle n'aurait eu que peu d'intérêt si l'idée d'algorithmes/modèles alternatifs n'avait pas été émise, mais cette idée permet de créer facilement et intuitivement des modèles adaptés aux besoins de différentes applications de la DGFiP.



Algorithmes alternatifs

Il existe de nombreux algorithmes qui, à partir de suites d'ids ou de paramètres, seraient capables de prévoir la validité d'une occurrence suivante. En effet, une partie de la force de DeepLog repose dans son architecture très modulaire, et il serait donc envisageable de remplacer une étape par une autre similaire.

Le premier algorithme qui a été choisi est les "forêts d'isolation" [4], qui se base sur une valeur d'isolation d'un point de données : plus il est facile de le séparer du reste des données, plus il a de chances d'être une anomalie (voir [article wikipedia](#)  pour une explication simple). Un module d'analyse de paramètre a été développé avec une implémentation des forêts d'isolation, capable de remplacer celui de DeepLog dans l'implémentation de Rémi.

C'est ici que j'ai eu mon premier contact avec le projet durant ma première année d'alternance. Il m'a fallu assister Léa sur la fin de son PFE, pour effectuer des tâches qu'elle n'avait pas le temps de faire. J'ai aussi pu apporter une aide technique sur la partie [apprentissage automatique](#) du projet, étant suite à avoir suivi un cours à ce sujet.

En m'inspirant de ce qui a été fait sur les forêts d'isolation, j'ai fait le choix de créer un module d'analyse de suites et de paramètres générique basé sur la librairie d'[apprentissage automatique](#) scikit learn. C'est une bibliothèque libre présentant de nombreux algorithmes communs d'[apprentissage automatique](#), tels que les forêts d'isolation. Utiliser une librairie telle que scikit me permet tout d'abord une meilleure robustesse et performance des algorithmes, mais aussi une certaine généricité. En effet, les algorithmes du même type ont tous la même structure de fonctions, et cela permet de très facilement les interchanger.

Les modules qui ont été développés permettent donc de remplacer l'analyse de paramètres de DeepLog par n'importe quel algorithme de la bibliothèque scikit, s'intégrant parfaitement dans la structure par blocs proposée par Blockly. Trois algorithmes de détections d'anomalies ont été retenus : les forêts d'isolation (version scikit), les SVM (Support Vector Model, ou [Séparateur à Vaste Marge](#) ) à une classe, et les LOF ([Local Outlier Factor](#) ). Ce sont trois méthodes de l'état de l'art en apprentissage non profond pour la détection d'anomalie, qui pourraient s'avérer utiles pour l'analyse de paramètres en particulier.

Chapitre 3

Le processus scientifique au cœur du projet

Nous arrivons maintenant au cœur de ce mémoire, qui démarre au début de mon PFE, et plus précisément l'arrivée de Robin à la [DGFIP](#). Robin était étudiant en Master ALMA à l'université de Nantes, et a travaillé avec moi sur le projet d'analyse de [logs](#) en tant que sujet de stage de fin de master. Il s'est chargé de beaucoup de travail de documentation sur le sujet ainsi que des sujets plus axés recherche.

Je vais détailler au sein de ce chapitre cette grande partie du projet et les différents points important que nous avons rencontrés.

3.1 État de l'art de l'analyse de logs

3.2 Analyse des jeux de données disponibles

3.3 Agilité et recherche : une union difficile

Chapitre 4

Développement du nouveau projet

Chapitre 5

Projection de l'impact du projet

5.1 Gain de temps et de productivité

5.2 L'humain contre l'algorithme

Conclusion

Bibliographie

- [1] Min Du and Feifei Li. Spell : Online streaming parsing of large unstructured system logs. *IEEE Transactions on Knowledge and Data Engineering*, 31(11) :2213–2227, 2019.
- [2] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. Deeplog : Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS '17*, page 1285–1298, New York, NY, USA, 2017. Association for Computing Machinery.
- [3] Pinjia He, Jieming Zhu, Zibin Zheng, and Michael R. Lyu. Drain : An online log parsing approach with fixed depth tree. In *2017 IEEE International Conference on Web Services (ICWS)*, pages 33–40, 2017.
- [4] Fei Tony Liu, Kai Ting, and Zhi-Hua Zhou. Isolation forest. In *Eighth IEEE International Conference on Data Mining*, pages 413 – 422, 01 2009.

Glossaire administratif

BSI-3 Bureau du SI des professionnels: Auparavant nommé SI-1C, bureau dépendant de la **DPN**, chargé du développement et de la maintenance des applications du domaine de la fiscalité des professionnels.

DGCP Direction Générale des Comptes Publics (DGCP): Aussi appelée «trésor public», ancienne administration chargée de la gestion des comptes de l'État et du recouvrement des impôts.

DGFIP Direction Générale des Finances Publiques: Service public de l'État rattaché au ministère des finances, chargé des missions de gestion publique et de fiscalité.

DGI Direction Générale des Impôts: Ancienne administration chargée de la liquidation des impôts.

DPN Direction des projets numériques: Direction au sein du **SI** regroupant les différents bureaux en charge de la direction et de la réalisation de projets dans le numérique.

DTT Division technique transverse: Ancienne division du SI-1C (**BSI-3** aujourd'hui) apportant un soutien technique aux autres équipes et proposant des projets d'innovation.

MEDOC MEcanisation Des Opérations Comptables: Application traitant l'encaissement des taxes (telles que la TVA) ainsi que la génération d'écritures comptables pour l'État.

SI Service des systèmes d'Information: Service support de la **DGFIP** chargé de la gestion informatique et du développement d'applications.

transverse Utilisé fréquemment à la **DGFIP** comme synonyme de transversal, dans le sens "recoupant plusieurs disciplines ou secteurs".

Glossaire technique

apprentissage automatique Ou *machine learning* en anglais, ensemble de méthodes visant à développer des algorithmes généraux basés sur l'apprentissage de données, s'opposant à un algorithme explicite classique.

apprentissage profond Branche de l'[apprentissage automatique](#) se basant sur des réseaux de neurones artificiels, comportant plusieurs couches de traitement de données permettant d'extraire des caractéristiques complexes.

log De l'anglais log (journal), sortie d'une application (souvent un fichier) représentant le chemin d'exécution d'une application (voir section [2.2](#)).

LSTM Long Short Term Memory (Longue mémoire à court terme): Type de réseau de neurones en [apprentissage profond](#) capable de retenir de l'information sur une longue durée.

TAL traitement automatique du langage: Ensemble de méthodes visant à analyser et traiter le langage naturel.

Annexes

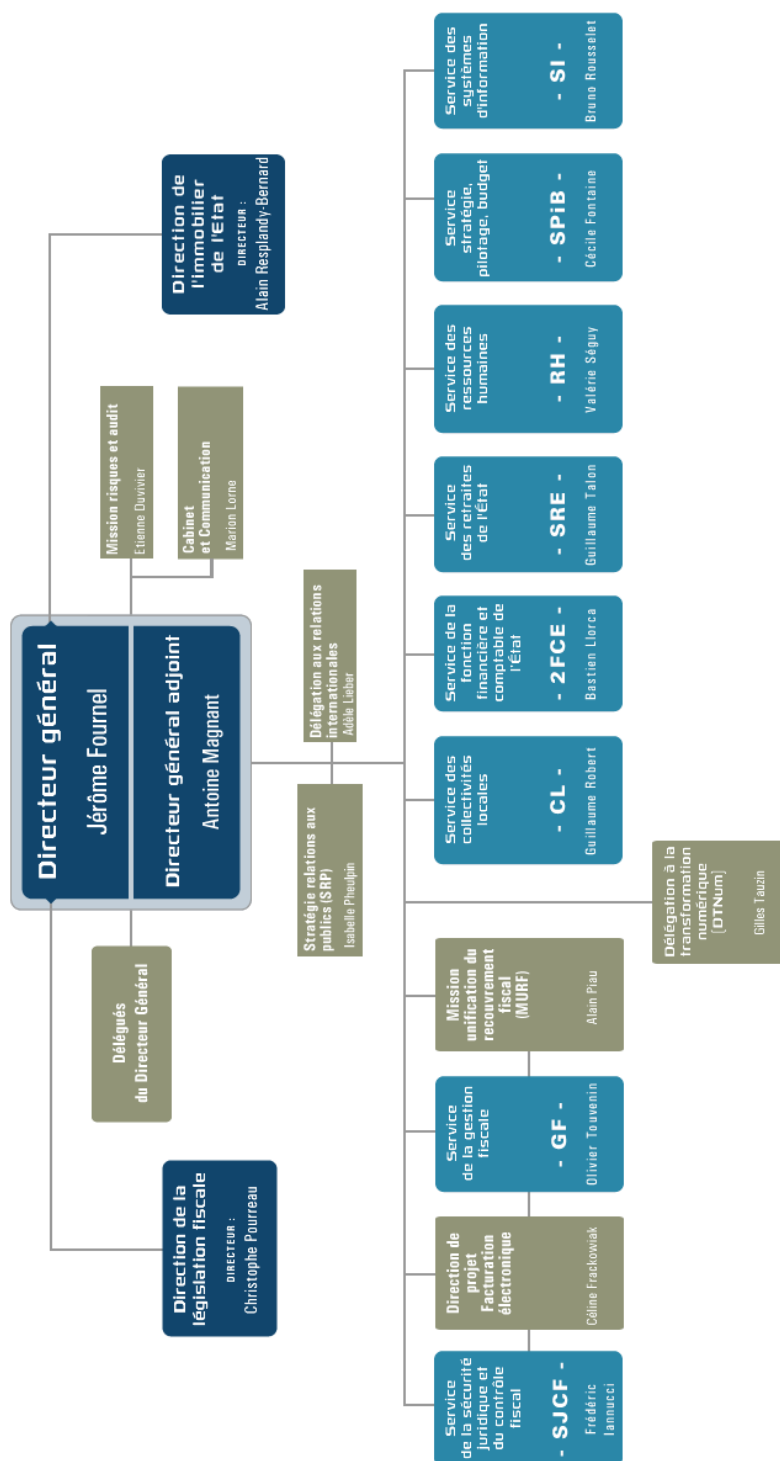


FIGURE 1 – Organigramme des services de la DGFIP

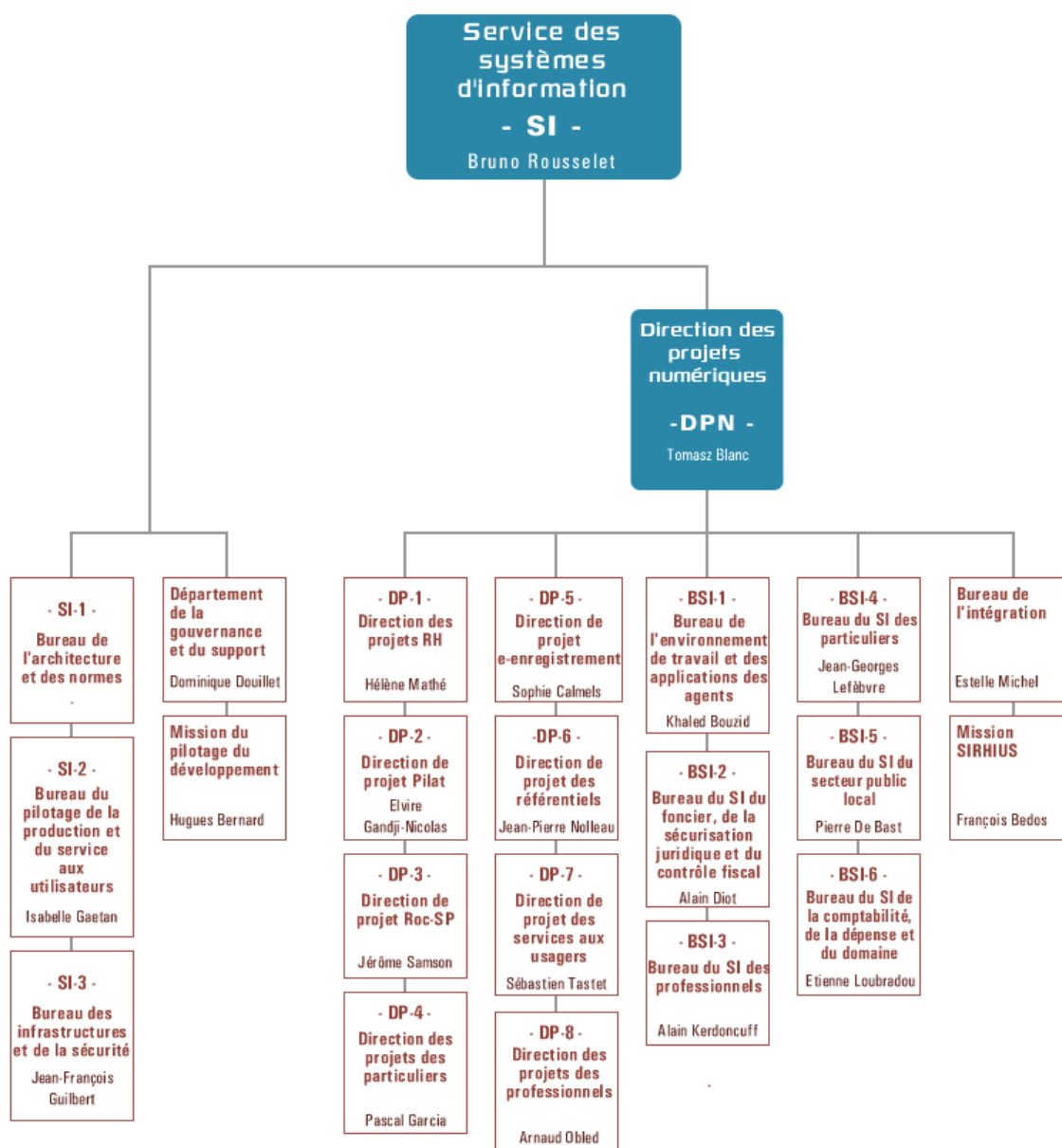


FIGURE 2 – Organigramme des bureaux du SI

Todo list

| | |
|--------------------------------------------------------------------------------------|---|
| <input type="checkbox"/> 1-2 pages | 1 |
| <input type="checkbox"/> Parler de la cellule innovation ? | 3 |
| <input type="checkbox"/> Détailler avec des chiffres | 3 |
| <input type="checkbox"/> Déplacer dans un chapitre 'notions' ? | 5 |
| <input type="checkbox"/> Ajouter une section sur les réseaux de neurones ? | 5 |