

Rapport de stage chez Yooz

# Détection optimisée des incohérences dans les fichiers PDF: une approche basée sur l'appariement de graphes



Cloud P2P Automation. Easy. Powerful. Smart.

Réalisé par :

Tim BROUSSARD

Sous la supervision de :

Saddok KEBAIRI

Année Universitaire 2024 - 2025

## Remerciements

Je tiens à remercier la société Yooz de m'avoir accueilli pour réaliser mon stage de seconde année de BUT Informatique et plus particulièrement, Vincent Poulain d'Andecy, Vice-Président du Département de la Recherche, Saddok Kébairi, Responsable de l'équipe Fraude & Technologies et mon maître de stage et Romain Ravaille Ingénieur en Recherche, pour leur accompagnement tout au long de mon stage.

Je tiens également à remercier l'équipe du Département de la Recherche plus généralement pour leurs conseils, leur bienveillance et pour m'avoir aidé à bien m'intégrer dans l'entreprise.

Enfin, je souhaite également remercier l'équipe pédagogique de l'IUT de Montpellier, de nous permettre de suivre des cours de qualité, nous offrant ainsi l'opportunité de travailler dans ce type d'entreprise.

## Résumé :

Ce document est un rapport de stage effectué au sein de l'entreprise Yooz et rédigé par un étudiant de l'IUT de Montpellier dans le cadre de sa deuxième année d'études. Il présente le travail effectué durant les 12 semaines de stages. Ce rapport parle des deux thématiques qui ont été abordées durant le stage :

1. Le refactoring de l'algorithme existant afin qu'il s'insère correctement dans la chaîne de traitement d'un document. Ce point aborde notamment la manière dont a été modifié l'architecture du code afin de passer d'un mode prenant en entrée un répertoire, à un mode prenant en entrée un fichier.
2. Les améliorations apportées à l'algorithme afin de réduire le nombre d'erreurs de classifications dans une solution visant à détecter des fraudes issues de modification dans des documents PDF. Cette partie liste point par point la liste des modifications qui ont été apportées en détaillant les observations réalisées, puis en décrivant les améliorations effectuées.

**Mots-clés :** Yooz, PDF, Fraude, Classifieur, learning (IA), Graphes, MétaBloc, XML, OCR, JPG, Surcouche Textuelle, Segmentation, Refactor, Matching, Distance.

**Ce rapport est strictement confidentiel et ne doit être ni distribué ni stocké sur un autre support que celui de Yooz**

# Sommaire

<b>1 - Introduction.....</b>	<b>1</b>
<b>2 - Contexte du stage.....</b>	<b>2</b>
2.1 - Présentation de l'entreprise.....	2
2.2 - Dimension commerciale.....	3
2.3 - Enjeux du stage.....	5
<b>3 - Analyse.....</b>	<b>6</b>
3.1 - Analyse de l'environnement technique.....	6
3.2 - Algorithme préexistant.....	7
3.3 - Analyse de la mission.....	12
<b>4 - Rapport technique.....</b>	<b>13</b>
4.1 - Refactoring de l'application.....	13
4.1.1 - Comprendre le projet.....	13
4.1.2 - Isolation du comportement par DataMatch.....	13
4.1.3 - Repenser certains objets.....	14
4.1.4 - Structure finale du projet.....	15
4.2 - Amélioration des résultats de détection de fraude.....	16
4.2.1 - Amélioration de l'assignation des zones fraudées.....	16
4.2.2 - Amélioration du matching entre les MétaBlocs.....	18
4.2.3 - Élargissement des Graphes pas assez dense.....	18
4.2.4 - Erreur sur la segmentation de certaines classes.....	21
4.2.5 - Application de la classification à la page.....	21
4.3 - Perspectives à venir : .....	22
<b>5 - Méthodologie et organisation du projet.....</b>	<b>24</b>
<b>6 - Conclusion.....</b>	<b>26</b>
<b>7 - Références.....</b>	<b>28</b>
<b>8 - Annexes.....</b>	<b>29</b>
Annexe 1 : Exemple fichier XML contenant la couche texte.....	29
Annexe 2 : Exemple fichier XML contenant les fraudes de surcouche.....	29
Annexe 3 : Exemple fichier XML contenant les paramètres d'une classe.....	30
Annexe 4 : Exemple de fichier XML entraînement.....	30
Annexe 5 : Fonctions permettant de vérifier qu'un mot est inclus dans un MétaBloc.....	31
Annexe 6 : Partie du code dans laquelle on cherche le minimum lors du matching par MétaBloc.....	31
Annexe 7 : Partie du code permettant de repérer les voisins d'un MétaBloc.....	32
Annexe 8 : Partie du code permettant de construire un méta-graphe à partir des voisins d'un MétaBloc.....	32
Annexe 9 : Exemple de résultat lors de la récupération des voisins d'un MétaBloc.....	33

# Table des figures

[Figure 1](#) : Workflow YooZ - page 3

[Figure 2](#) : Etapes d'exécution de l'algorithme préexistant - page 7

[Figure 3](#) : Segmentation d'une page - page 8

[Figure 4](#) : Schématisation d'une page traitée - page 8

[Figure 5](#) : Formule de calcul de la distance MétaBloc - page 9

[Figure 6](#) : Exemple de regroupement en DataMatch - page 9

[Figure 7](#) : Exemple de graphe représentant un MétaBloc fraudé - page 10

[Figure 8](#) : Schéma avant/après traitement des pages - page 14

[Figure 9](#) : Représentation du problème d'index des pages - page 14

[Figure 10](#) : Nouveau processus appliqué à une page dans le projet - page 16

[Figure 11](#) : Exemple zone fraudée d'une page - page 17

[Figure 12](#) : Exemple affichage d'un MétaBloc fraudé avant correction - page 17

[Figure 13](#) : Schéma représentatif du problème d'assignation de zones fraudées - page 17

[Figure 14](#) : Représentation problème densité des graphes - page 19

[Figure 15](#) : Schéma représentant l'élargissement d'un graphe - page 20

[Figure 16](#) : Résultat de la classification lors d'une fraude volontaire - page 22

[Figure 17](#) : Tableau Jira du projet de mon stage - page 24

[Figure 18](#) : Schéma représentant l'architecture des branches du projet - page 24

[Figure 19](#) : Table de confusion pour la classe Transgourmet - page 26

[Figure 20](#) : Tableau répertoriant les résultats des tests (en %) - page 26

# Glossaire

## P2P :

Pour « Purchase2Pay » ou « Procure2Pay ». Processus fonctionnel de gestion des achats dans une entreprise, allant de l'expression d'un besoin jusqu'au règlement du fournisseur en passant par la demande d'achat, la commande fournisseur, la réception du bien ou du service, le traitement de la facture associée, sa comptabilisation et son paiement.

## JPG :

Une des normes qui définit le format d'enregistrement et de décodage pour une représentation numérique compressée d'une image.

## XML :

Pour « eXtensible Markup Language » (langage de balisage extensible), un langage de balisage généraliste, servant à structurer des données sous forme d'arbre, utilisé ici pour structurer les PDF d'origine.

## Faux positifs :

Fraude ayant mal été classifiée, dans notre contexte, une fraude dite comme connue de la base d'entraînement de l'application, alors qu'elle ne l'est pas.

## OCR :

Pour « Optical Character recognition » (OCR) désigne les procédés informatiques pour la traduction d'images de textes imprimés/dactylographiés en fichiers de texte.

## MétaBloc :

Un ensemble de mots découpé par un procédé de segmentation, afin de former une unité logique.

**Graphe de MétaBloc :**

Un graphe représentant un MétaBloc, chaque nœud représente un mot, chaque arête représente la distance physique entre ces mots et les nœuds rouges représentent les mots qui ont été modifiés, c'est-à-dire la zone qui a été fraudée.

**DataMatch :**

Groupe de MétaBloc de pages différentes dont la distance est suffisamment basse pour être considéré similaire.

**Classe de document :**

Un ensemble de documents qui présente de forte similitude sur leur structure et sur le type de zone fraudée

**YoozProtect :**

Solution développée et vendue par Yooz visant à détecter les fraudes, protéger les documents et garantir leur conformité

# 1 - Introduction

Ce rapport de stage présente le travail que j'ai réalisé au sein de l'entreprise Yooz du 13 Janvier au 4 Avril 2025 dans le cadre de ma formation à l'IUT Informatique de Montpellier.

L'utilisation d'Internet accélère aujourd'hui l'utilisation des transactions numériques et des échanges de documents électroniques toujours plus nombreux dans les entreprises. En 2021, dans ce contexte, 95% des entreprises disent avoir subi une tentative de fraude [1]. Les escroqueries ayant évolué, les entreprises doivent alors assurer une gestion rigoureuse de leurs factures. Afin de limiter les risques, il devient vital d'identifier les fraudes sur ces factures. Toutefois, au vu du nombre titanesque de transactions effectuées par jour et les caractéristiques toujours changeantes de ces factures, il semble impératif de trouver une solution capable d'automatiser le processus d'évaluation de l'information des factures [2].

C'est dans ce contexte que Yooz, qui est une entreprise proposant divers services autour de la facturation électronique, a eu besoin de mettre en place un outil de reconnaissance des incohérences sur les factures de ses clients dans son outil d'automatisation du traitement des factures, partie du projet [YoozProtect](#). Le projet que l'on m'a confié est une application ayant été développée par d'anciens stagiaires et M. Kébairi, visant à améliorer la reconnaissance de ces incohérences. Cela permettra ainsi de réduire le coût humain de l'entreprise, en effet, cela évitera à l'équipe de recherche d'avoir à effectuer un travail manuel de classification tous les matins, pouvant parfois durer jusqu'à plus d'une heure.

Dans ce rapport, nous allons d'abord présenter le contexte du stage avec une présentation de l'entreprise, son fonctionnement ainsi que l'organisation de ses équipes. Nous verrons également sa dimension commerciale, sa place dans le marché ainsi que ses enjeux. Par la suite, nous verrons l'ensemble de l'environnement technique en lien avec le stage avant d'analyser l'algorithme préexistant afin de mieux comprendre les modifications apportées, après quoi nous analyserons la mission du stage. Dans un second temps, nous observerons d'abord l'ensemble du refactoring du projet qui a été effectué afin de l'intégrer au mieux à la solution proposée par Yooz, puis nous analyserons chacune des améliorations apportées afin de réduire le nombre de [faux positifs](#) détectés par l'algorithme. Nous conclurons enfin par la méthode et l'organisation du projet tout au long de mon stage et enfin les résultats que j'ai pu apporter à l'entreprise.

## 2 - Contexte du stage

### 2.1 - Présentation de l'entreprise

Yooz est une compagnie mondiale opérant dans le secteur de la digitalisation des achats et des factures. Elle propose une solution de digitalisation des factures à ses 5 000 entreprises clientes, ce qui la positionne comme leader dans son domaine. Implantée en France à Aimargues et Montpellier, ainsi qu'à Dallas aux États-Unis et récemment présente sur le marché espagnol. Yooz emploie actuellement plus de 450 personnes, dont environ 350 en France et 100 aux États-Unis.

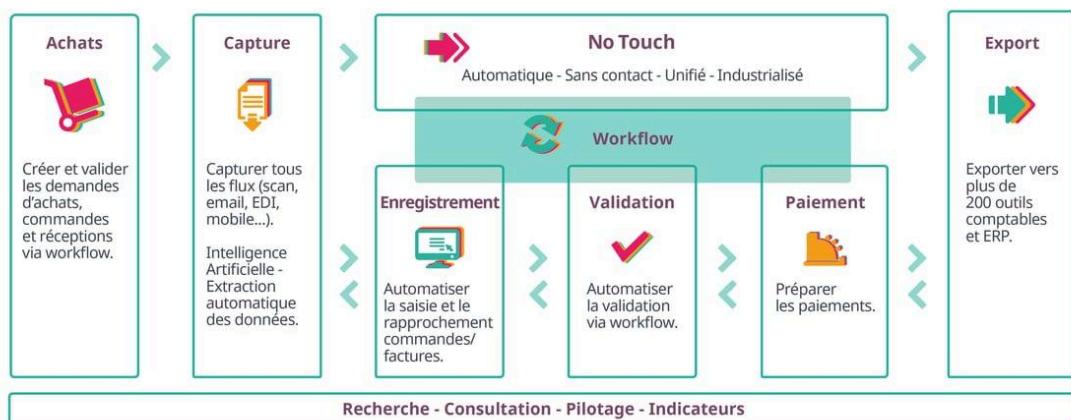
Pour ma part, je suis affecté au département Recherche et Technologie qui est un département dont le but est de concevoir de nouvelles solutions innovantes adaptées aux besoins des clients, tout en maintenant une avance sur la concurrence. Ce département comporte une vingtaine d'employés répartis en trois branches toutes sous la direction de Vincent POULAIN D'ANDECY : Yoozing, Innovation et enfin Fraude et Technologies. Lors de mon stage, j'ai eu l'opportunité d'intégrer l'équipe de la branche Fraude et Technologies sous la supervision de M. Saddok Kébairi, celle-ci visant à détecter les fraudes dans les documents PDF et à les traiter.

Yooz met à disposition de ses employés un open space dans lequel nous avons la possibilité de réserver une place sur l'application LUCCA ou bien d'indiquer que l'on souhaite faire du télétravail. Chaque poste mis à disposition dans l'open space est équipé de deux écrans, un clavier, une souris, une connexion stable, une alimentation passant tous dans le même câble permettant donc de s'installer rapidement où l'on souhaite et offrant une grande flexibilité. En plus de cette open space l'entreprise nous fournit des PC récents pour travailler ainsi qu'un micro casque.

## 2.2 - Dimension commerciale

L'objectif de Yooz est de fournir à tous ses clients des solutions logicielles innovantes de dématérialisation et de traitement de documents dans le Cloud. L'entreprise propose des outils performants permettant d'automatiser et d'optimiser la gestion des documents comptables et financiers. Ses solutions sont particulièrement adaptées aux entreprises souhaitant réduire leur dépendance au papier, améliorer leur efficacité opérationnelle et renforcer leur conformité réglementaire.

L'un des principaux atouts de Yooz est sa capacité à prendre en charge l'ensemble du processus [P2P \(Purchase-to-Pay\)](#), incluant la réception, l'extraction, la validation et le paiement des factures. En automatisant ces étapes, Yooz permet à ses clients de diminuer considérablement le temps consacré au traitement manuel et de limiter les risques d'erreurs. L'optimisation de ce flux de travail favorise également une meilleure visibilité sur les finances des entreprises.



**Figure 1 : Workflow Yooz**

L'offre de Yooz repose sur un modèle **SaaS (Software as a Service)**, cela signifie qu'elle propose des logiciels dans lesquels ceux-ci sont installés sur des serveurs distants plutôt que sur la machine de l'utilisateur. Ce modèle inclut également un support client, des mises à jour régulières et une évolutivité adaptée aux besoins des entreprises. Les clients peuvent choisir différentes options en fonction du volume de documents traités et des fonctionnalités souhaitées, garantissant ainsi une flexibilité et un ajustement aux exigences spécifiques de chaque organisation.

D'après Yooz, son application permet de réduire jusqu'à 80 % des coûts liés à la gestion des documents et des factures. Cette réduction des coûts, couplée à un gain de temps significatif, améliore directement la rentabilité des clients et contribue à renforcer leur compétitivité sur le marché.

Sur le plan financier, Yooz a connu une forte croissance ces dernières années. À la fin de l'année 2021, l'entreprise affichait un revenu annuel récurrent (ARR) de 37 millions de dollars, en progression de 37 %.

Yooz a notamment rencontré un vif succès aux États-Unis, où elle réalise désormais 40 % de son chiffre d'affaires. Son implantation sur ce marché stratégique lui permet de toucher une clientèle variée, allant des PME aux grandes entreprises, et de consolider sa présence à l'international.

En s'appuyant sur une stratégie commerciale axée sur l'innovation, la satisfaction client et l'expansion internationale, Yooz continue de renforcer sa position en tant que leader des solutions de dématérialisation et d'automatisation. Son ambition est de poursuivre cette croissance en développant de nouvelles fonctionnalités basées sur l'intelligence artificielle et en consolidant son réseau de partenaires commerciaux.

## 2.3 - Enjeux du stage

Dans le cadre de mon stage, mon sujet est : Détection des incohérences dans les « images/documents PDF » : Finalisation et industrialisation des travaux en cours. Au sein de ce sujet, la mission qui m'a été confiée pour le moment est d'améliorer la pertinence des alertes émises par la technologie de détection des fraudes dans les documents PDF. Concrètement, la détection de fraudes proposée par Yooz permet de détecter les fraudes de surcouches, c'est-à-dire une zone de texte qui recouvre une autre zone de texte. L'importance de ses modifications peut varier, en effet, il peut s'agir d'une modification du bénéficiaire, des dates et des montants, tout comme il peut s'agir d'un message rajouté par un fournisseur afin de transmettre une information à son client. Actuellement, la solution ne distingue pas ses deux types de fraudes et elles sont donc triées à la main par un membre de l'équipe recherche ce qui représente une perte de temps et d'argent. Cependant, il y a un grand nombre de ces fraudes de faible importance qui reviennent très régulièrement, celles-ci pouvant avoir été modifiées en ajoutant une valeur par-dessus une autre, la détection actuelle des fraudes permet de détecter ces modifications et d'émettre une alerte afin d'en informer l'entreprise. Mon objectif lors de mon stage sera donc d'améliorer un projet d'exploration entamé par 2 stagiaires avant moi visant à identifier ces fraudes de surcouches récurrentes et ainsi de ne pas émettre d'alertes pour celles-ci.

## 3 - Analyse

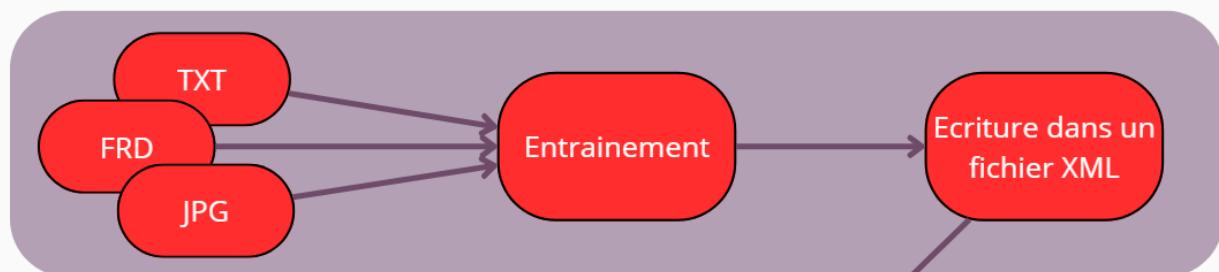
### 3.1 - Analyse de l'environnement technique

Comme décrit dans la partie 1.2 le logiciel mis en place par l'entreprise est vaste et propose de nombreuses fonctionnalités, en effet, il dispose d'une numérisation, de la reconnaissance automatique du type de document, de l'enregistrement et des imputations, du traitement des factures, du circuit de validation électronique, du contrôle de conformité, de l'exportation des écritures vers les ERP/logiciels comptables et de l'archivage électronique. Pour ma part, celle que je vais être amenée à modifier est l'étape du contrôle de conformité. Les entrées qui sont fournies à la solution sur laquelle je travaille, est le résultat d'un autre algorithme exécuté avant celle-ci qui permet de séparer un document PDF en trois parties : L'image [JPG](#) représentant la partie visuelle du PDF. Un fichier [XML](#) représentant la couche textuelle du document, celui-ci étant découpé en blocs, composé de lignes elles-mêmes composées de mots composés de caractères. Ce fichier regroupe également d'autres types d'informations comme par exemple : la position des bordures sur l'image. Dans la section "word" représentant un mot de la page, on retrouve ses coordonnées à gauche, à droite, en haut et en bas ([Cf. Annexe 1](#)). Un autre fichier XML contient les fraudes détectées par la solution précédente, il est découpé par page puis par "zone fraudée". Chacune des "zones fraudées" est constituée de 5 éléments importants, : "FRD\_WORD\_VAL" et "FRD\_WORD\_BB" constituent respectivement le mot fraudé (c'est-à-dire le mot recouvrant une zone) et ses coordonnées, "ORG\_WORD\_VAL" et "ORG\_WORD\_BB" constituent respectivement le mot qui a été recouvert par le mot fraudé et ses coordonnées. Enfin, "CONFIDENCE" représente l'indice de confiance dans cette fraude, autrement dit, un pourcentage représentant à quel point l'algorithme est sûr que cette fraude existe bel et bien dans le document ([Cf. Annexe 2](#)). Enfin, parmi les spécificités techniques du projet, le langage de programmation utilisé est le C++, l'[IDE](#) utilisé est Visual Studio 2022. Pour ce qui est des bibliothèques utilisées, parmi les plus importantes et celle que j'ai été amené à utiliser : [openCV](#), [iostream](#) et [fstream](#). Nous avons également utilisé XnView qui est une visionneuse d'images afin d'observer les documents et de pouvoir voir les surcouche de fraude et Notepad++, qui est un éditeur de texte libre, pour pouvoir lire les fichiers XML. De plus, j'ai également utilisé le format DOT qui est un langage de description de [graphe](#) dans un format texte et le format XLS (Excel) afin de répertorier les résultats de nos tests et de faire des statistiques de ceux-ci.

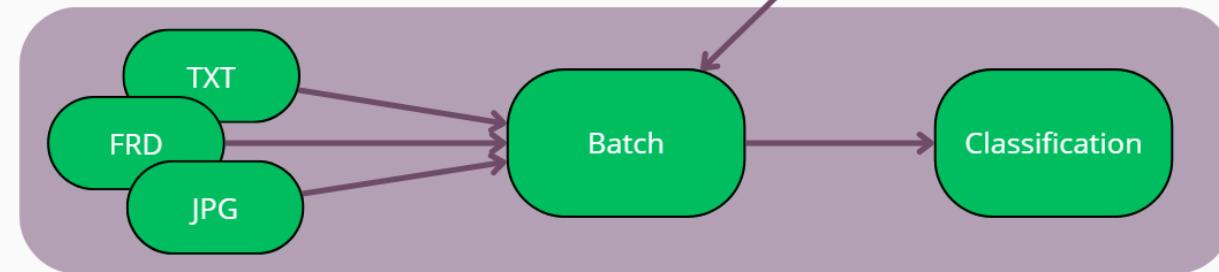
### 3.2 - Algorithme préexistant

L'algorithme préexistant s'effectue en deux étapes. La première étape consistait à analyser les documents faussement fraudés et récurrents afin de constituer une base d'entraînement. Ainsi il apprend à reconnaître les fausses fraudes au travers des exemples. La seconde utilise cet apprentissage pour déterminer les incohérences dans les nouveaux fichiers PDF qui lui sont passés.

#### Etape 1



#### Etape 2

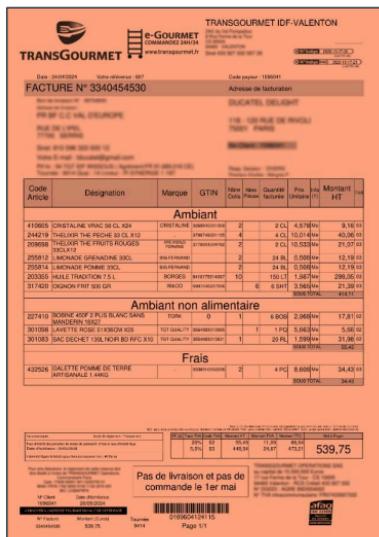


**Figure 2 : Etapes d'exécution de l'algorithme préexistant**

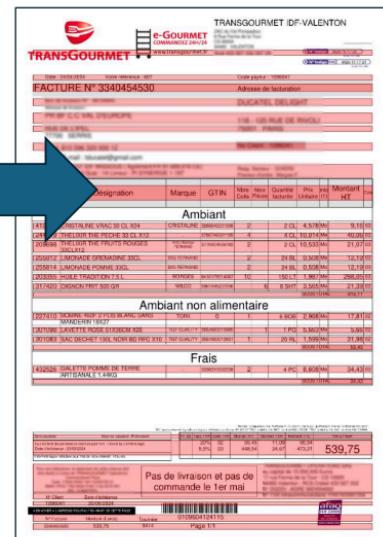
Concrètement, l'entrée de l'algorithme récupérait la sortie d'une autre solution. Cette sortie se compose de 3 parties, JPG, XML de texte, et XML de fraudes comme dit précédemment. L'algorithme était composé de deux parties principales, une partie entraînement et une partie analyse qui ne pouvait se faire que par dossier. Tout d'abord, la partie entraînement se déroule de la manière suivante, premièrement l'initialisation d'une page, la construction de ses [DataMatch](#), le repérage des fraudes dans la page et enfin la sauvegarde de l'information dans un fichier XML. La partie initialisation consiste à récupérer dans un premier temps les paramètres d'une [classe](#) ([Cf. Annexe 3](#)) qui permettent notamment la segmentation de l'image extraite du PDF qui est un procédé visant à découper verticalement puis

horizontalement celle-ci afin d'obtenir ses lignes puis ses MétaBlocs composés de mots.

#### Niveau 0 – Page entière



#### Niveau 1 – Division Verticale



#### Niveau 2 – Division Horizontale

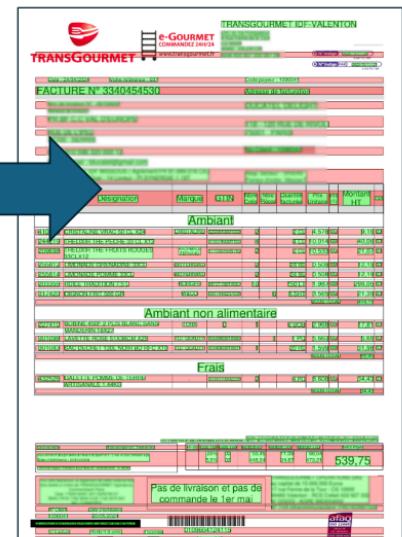


Figure 3 : Segmentation d'une page

A des fins de schématisation nous pouvons représenter cette segmentation de la manière suivante ([Cf. Figure 4](#)) où les rectangles bleus représentent les MétaBlocs.

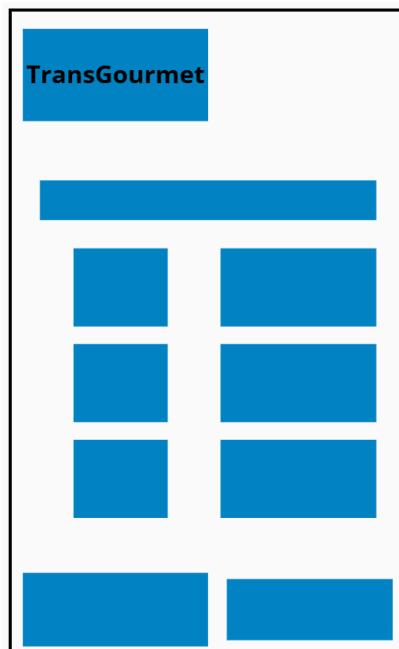


Figure 4 : Schématisation d'une page traitée

Ensuite, pour regrouper ces MétaBlocs en groupe en fonction des similitudes présentes entre eux, pour cela, l'algorithme dispose d'une fonction de matching entre deux MétaBlocs. Cette fonction va calculer la distance entre deux MétaBlocs,

cette distance est pondérée par plusieurs facteurs, dont le nombre de caractères, la taille moyenne des caractères, leur distance physique (ci-dessous la formule complète).

$$Distance_{ij} = \sqrt{\sum \left( \frac{(x_i - x_j)^2}{xIndexDiffCoef} + \frac{(y_i - y_j)^2}{yIndexDiffCoef} + \frac{(h_i - h_j)^2}{fontSizeDiffCoef} + \frac{(w_i - w_j)^2}{nbCharDiffCoef} \right)}$$

**i** : rang du Métabloc référent

**j** : rang du Métabloc candidat

**x** : valeur en pixel du centre de gravité horizontale du Métabloc dans l'espace.

**y** : valeur en pixel du centre de gravité verticale du Métabloc dans l'espace.

**h** : valeur en pixel de la hauteur du Métabloc

**w** : valeur en pixel de la largeur du Métabloc

$$xIndexDiffCoef = \frac{1}{(\|xIndex_i - xIndex_j\| + 1)^2} \quad \mid \quad yIndexDiffCoef = \frac{1}{(\|yIndex_i - yIndex_j\| + 1)^2}$$

**xIndex** : l'index du rang horizontale du Métabloc dans le tableau

**yIndex** : l'index du rang verticale du Métabloc dans le tableau

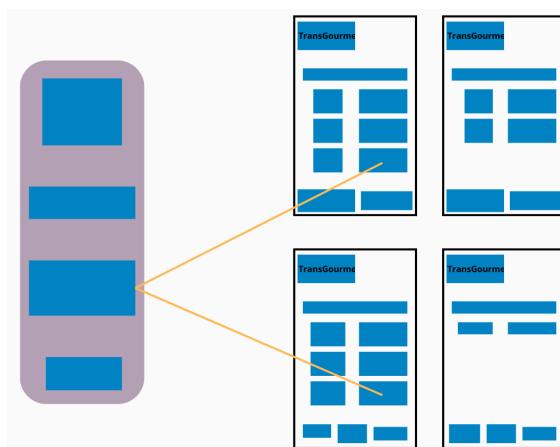
$$nbCharDiffCoef = \frac{\min(nbChar_i, nbChar_j)}{\max(nbChar_i, nbChar_j)} \quad \mid \quad fontSizeDiffCoef = \frac{\min(fontSize_i, fontSize_j)}{\max(fontSize_i, fontSize_j)}$$

**nbChar** : le nombre total de caractère du Métabloc

**fontSize** : la taille moyenne de la police du Métabloc en pixel

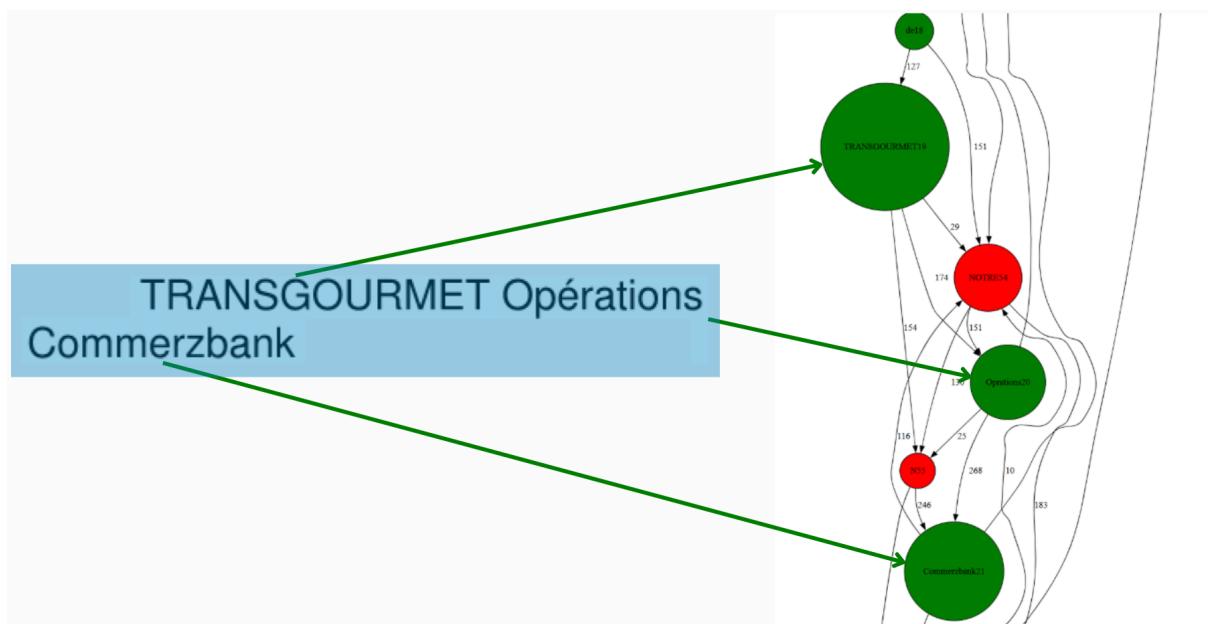
**Figure 5 : Formule de calcul de la distance MétaBloc**

Ci-dessous, un exemple de regroupement en DataMatch en utilisant la formule de calcul de la distance MétaBloc



**Figure 6 : Exemple de regroupement en DataMatch**

Ensuite, l'algorithme va lire les fraudes contenues dans le fichier .xml de fraudes et va chercher le MétaBloc ayant la plus petite distance physique entre son barycentre et celui du mot fraudé. Chaque MétaBloc ayant match avec un mot fraudé devient un MétaBloc lui-même fraudé. Des graphes orientés sont par la suite créés à partir de ces MétaBlocs fraudés, leur construction est faite de la manière suivante : un mot correspond un nœud et une arête représente la distance physique entre ces deux mots. Certains nœuds sont représentés en rouge pour indiquer que ce sont des mots qui ont été recouvert par d'autres, les mots recouvrant quant à eux ont leurs arêtes qui sont dirigés vers le(s) noeud(s) rouge(s) sinon les arêtes sont dirigés selon le sens de lecture du document. (Cf. Figure 7) Ci-dessous, le MétaBloc sur la gauche a donné le graphe sur la droite, nous voyons donc que le MétaBloc recouvrail deux mots : "NOTRE N°".



**Figure 7 :** Exemple de graphe représentant un MétaBloc fraudé

Enfin, pour finir la phase d'entraînement, chaque MétaBloc ainsi que les graphes qui leur sont associés sont écrits dans un fichier XML “d'entraînement” afin de pouvoir les utiliser plus tard dans la partie analyse. Ce fichier XML d'entraînement est donc composé de deux parties, une partie MétaBloc avec ses caractéristiques (position, taille de la police, nombre de caractères ainsi que son contenu) et une autre partie Graphe avec son ordre, sa taille, la liste de ses nœuds (pour chacun d'entre eux la position dans le document du mot représenté et sa valeur) et la liste de ses arêtes. ([Cf. Annexe 4](#))

La deuxième partie de l'algorithme consiste à tester notre entraînement sur un répertoire de documents. Celle-ci comporte des éléments en commun avec le mode entraînement : l'algorithme récupère les pages qui doivent être traitées, récupère les paramètres de la classe afin d'aider à la segmentation et de récupérer les threshold (seuil) de graphe et de MétaBloc qui sont des limites qui permettront de décider si une fraude est connue ou non. Les pages sont ensuite segmentées pour obtenir des MétaBlocs et les fraudes récupérées dans le fichier XML de fraudes sont assignées à des MétaBlocs eux-mêmes devenus fraudés.

Ensuite, le fichier XML dans lequel les MétaBloc et les graphes d'entraînement ont été stockés est lu et parcouru par l'algorithme. Par la suite, les objets du même nom sont initialisés afin de les utiliser plus tard. L'algorithme parcourt par la suite l'ensemble des MétaBlocs fraudés des pages et chacun d'entre eux va passer par 2 étapes pour décider de s'il s'agit d'une fraude connue (et donc l'ignorer) ou bien s'il s'agit d'une fraude inconnue (et donc émettre une alerte). Premièrement, l'algorithme va chercher un MétaBloc dont la distance (calculée de la même manière que pour l'entraînement) respecte la condition d'être inférieure au threshold de la distance MétaBloc qui a été récupéré dans les paramètres de la classe, si ce n'est pas le cas alors la fraude est considérée comme inconnue.

Par la suite, l'algorithme va passer par une deuxième étape qui consiste à créer un graphe à partir du MétaBloc fraudé parcouru et de le comparer aux graphes de l'entraînement. Pour cela, il va calculer un second coût cette fois-ci basé sur la distance entre deux graphes qui sera pondérée en fonction de deux critères : le coût des nœuds multiplié par le coût des arêtes, le coût des nœuds correspond à la différence entre l'ordre de chacuns des deux graphes + 2 fois le nombre de noeuds fraudés (donc les nœuds rouges) et pour ce qui est du coût des arêtes il correspond à la différence entre la taille des deux graphes. Ainsi, il récupère la distance la plus basse avec le graphe fraudé et si cette distance est inférieure au threshold graphe alors cette fraude est considérée comme connue, sinon elle est inconnue.

### 3.3 - Analyse de la mission

Ma mission s'inscrit dans la continuité de ce qui a été commencé par les précédents stagiaires, mon objectif sera d'améliorer cette application en utilisant une solution de Graph Matching et en améliorant l'algorithme existant.

Tout d'abord, le Graph Matching est un procédé en Mathématiques qui consiste à comparer plusieurs graphes entre eux et de trouver des similitudes. Actuellement, l'algorithme préexistant permet de modéliser le MétaBloc d'une image potentiellement fraudé sous forme d'un graphe en représentant un mot comme un nœud, la distance physique en pixel entre deux mots est inscrite sur les arêtes, enfin certains nœud sont représentés en rouge représentant donc la partie qui est désormais recouvert par d'autres mots. Ce graphe, ainsi formé est appelé un graphe de MétaBloc fraudé. Parmi les solutions proposées, l'une d'elles serait de procéder à cela dans l'entièreté d'une page de PDF formant donc un méta graphe et d'utiliser le Graph Matching sur l'ensemble des documents désormais représentés par des méta graphes afin de mettre en évidence les incohérences. Cela nécessite d'effectuer un travail de recherche dans des domaines de pointes, mais surtout une maîtrise de l'algorithme existant d'autant plus que le langage utilisé dans le projet est le C++ qui est un langage que je n'ai jamais appris, heureusement mes connaissances dans d'autres langages de programmation m'aideront à apprendre rapidement celui-ci. De plus, nous avons étudié en première année la théorie des graphes.

## 4 - Rapport technique

### 4.1 - Refactoring de l'application

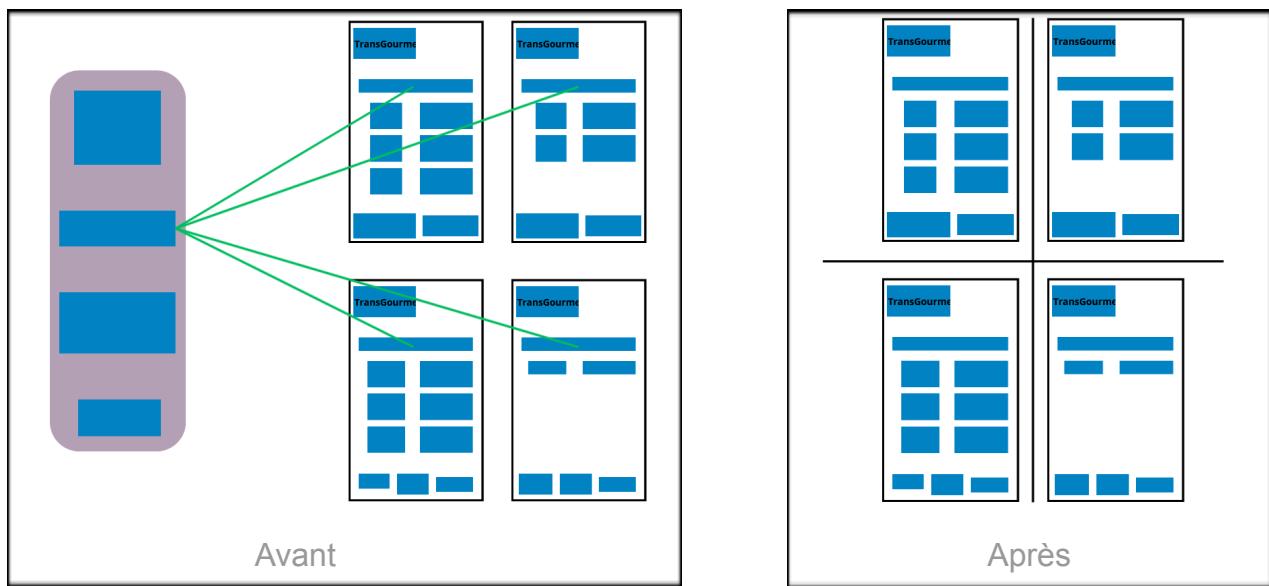
La solution actuellement proposée par Yooz à ses clients suit un processus dans lequel un document va recevoir une chaîne de traitements. Parmi ces traitements, la détection de fraudes. L'entrée de mon algorithme doit donc correspondre avec les données qui lui seront passées. C'est en ce sens que le fonctionnement par répertoire de dossier est incompatible.

#### 4.1.1 - Comprendre le projet

L'idée de cette première mission était de proposer un mode "mono", c'est-à-dire un mode prenant en entrée les 3 fichiers composant un document PDF lors de la chaîne de traitement (l'image JPG, le fichier XML couche texte et le fichier XML de fraude) au lieu d'un répertoire. La première étape dans cette mission a été d'assimiler le code existant, en effet, pour être capable de faire des modifications dans l'entièreté de l'algorithme sans altérer son bon fonctionnement, il me fallait une très bonne connaissance de celui-ci. Plusieurs difficultés me sont alors apparues, premièrement le langage de programmation utilisé était le C++ et le code reposait sur la programmation orientée objet, bien que nous avions déjà étudié la programmation orientée objet durant les deux premières années de BUT Informatique, il m'a fallu me documenter en ligne et suivre des formations rapides en C++ afin d'assimiler les bases du langages et ses subtilités, en ce sens, mes connaissances en Java m'ont grandement aidées. Secondelement, le projet comportait plus de treize mille lignes de code avec des concepts qui m'étaient inconnus, cela a rendu ma compréhension du projet plus longue et difficile que prévu.

#### 4.1.2 - Isolation du comportement par DataMatch

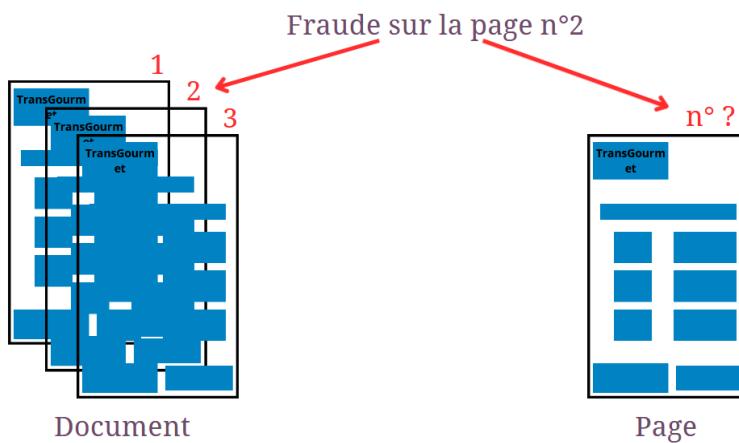
Lorsque j'avais bien assimilé le code du projet, il m'a fallu débuter le refactoring du projet. J'ai dû dissocier le fonctionnement par groupe de documents qui récupérait l'entièreté des documents qui lui étaient passés en entrée pour former ce qui est appelé des DataMatch dans le projet, c'est-à-dire un ensemble de MétaBloc présentant des similitudes suffisamment fortes pour être considérés comme similaires. Ainsi, l'entièreté du processus du projet était appliquée à ces groupes directement, il fallait donc modifier le comportement de chacune des fonctions afin d'isoler leur comportement et de l'appliquer cette fois-ci sur un bloc au lieu d'un match.



**Figure 8 : Schéma avant/après traitement des pages**

#### 4.1.3 - Repenser certains objets

Il a également fallu repenser certains objets, en effet, une des différences entre la version présente et la version souhaitée était que l'ancienne version pouvait traiter un document complet, qui pouvait donc comporter plusieurs pages or dans la nouvelle version, on souhaitait traiter les pages une à une désormais. Un des problèmes majeurs était que jusqu'ici pour reconnaître une fraude sur une page d'un document celle-ci avait un index correspondant au numéro de la page dans le document et lorsque l'objet document était initialisé, les pages de celui-ci étaient parcouru et numéroté dans l'ordre de lecture. Cependant, comme cela se faisait au moment de l'initialisation de l'objet dans le projet, cette information va désormais se perdre puisqu'il va recevoir en entrée des pages sans savoir de quelle page du document il s'agit.



**Figure 9 : Représentation du problème d'index des pages**

Il fallait donc trouver un nouveau moyen de repérer une fraude sur une page. J'ai d'abord essayé de modifier le moyen dont est identifié une fraude dans le fichier XML contenant les fraudes, notamment en essayant d'identifier la zone fraudée sur la page parcourue cependant cela n'offrait aucune garantie puisque deux pages pouvaient avoir la même zone sans pour autant que les deux soient fraudées. La solution que j'ai alors mise en place consiste à se servir du nom du fichier, en effet, j'ai pu constater que l'algorithme permettant de découper les PDF (donc des documents) en pages avec les 3 types de fichiers XML et JPG utilisait une convention de nommage telle que : nomFichier\_indexPage.extension ainsi il me suffisait d'isoler la partie index de la page, pour cela j'ai utilisé la fonction "find\_last\_of" qui, appliquée à une chaîne de caractère trouve l'indice de la dernière occurrence d'une chaîne de caractère qui lui est passé en paramètre ainsi en retrouvant l'emplacement du dernier underscore et du dernier point je peux récupérer tout ce qui se situe dans cet intervalle.

#### 4.1.4 - Structure finale du projet

Ainsi, après avoir réglé les problèmes d'index sur les pages, j'ai repris les différentes fonctions du projet comme dis précédemment et modifié de sorte à ce que chacune des fonctions s'appliquent à une page unique. Voici donc une figure ([Cf. Figure 10](#)) représentant donc le nouveau chemin suivi par une page, les modes entraînement et mono vont chacun commencer par initialiser la page d'entrée, les découper, récupérer les MétaBloc, ensuite matcher les fraudes correspondant à la page en cours et enfin créer un graphe pour chaque MétaBloc fraudé. Enfin, le mode batch va parcourir les pages et appeler le mode mono pour chacune d'elles. Le mode entraînement va écrire dans un fichier XML d'entraînement les MétaBloc et les graphes des zones fraudées (cette étape ne diffère pas de celle présente à l'origine). Enfin, le mode mono va lui lire le fichier XML d'entraînement afin de récupérer les MétaBloc et les graphes, puis les deux étapes de matching et de calcul de distance précédemment décrites seront utilisés pour décider de la classification de la page.

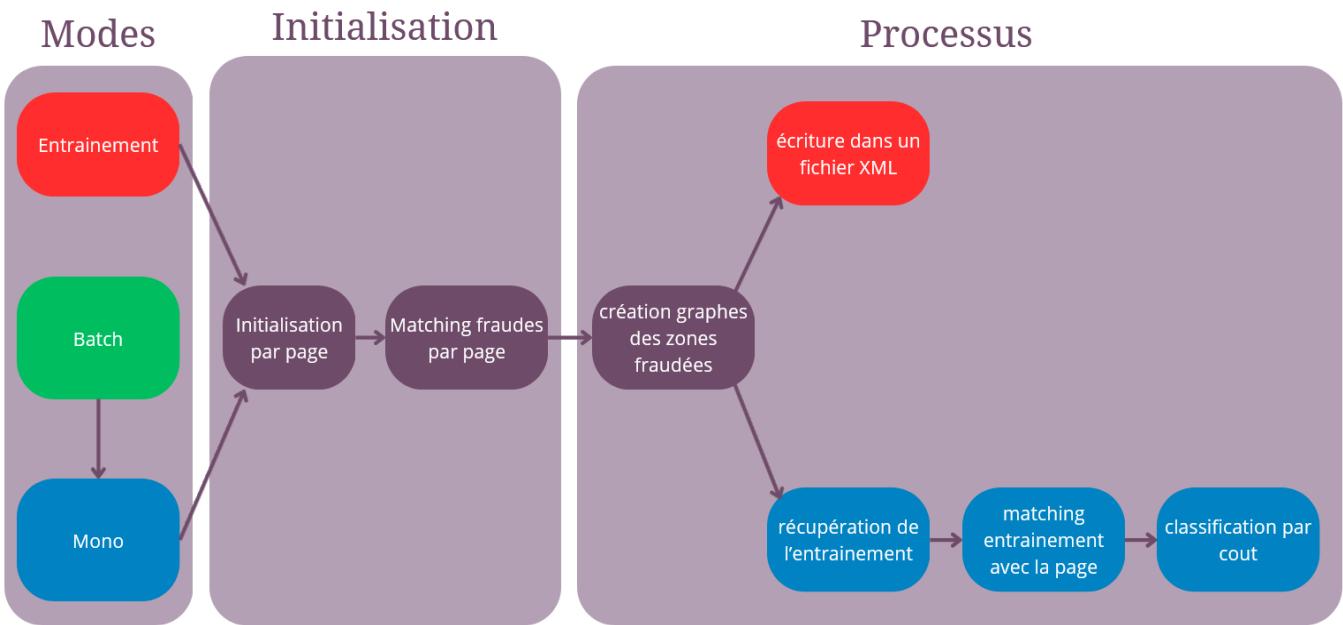


Figure 10 : Nouveau processus appliqué à une page dans le projet

## 4.2 - Amélioration des résultats de détection de fraude

Après avoir refactoré le code du projet, j'ai pu observer les résultats et il est apparu que sur une première classe les résultats étaient similaires à avant le refactoring et j'ai pu m'assurer que les fonctionnalités principales étaient toujours fonctionnelles. Ainsi débuta la partie de mon stage qui consistait à réduire les faux positifs présents dans les documents. J'ai alors adopté une approche qui consistait à prendre un cas qui ne fonctionnait pas, qui ne donnait pas les résultats escomptés et d'utiliser le mode débogage de Visual Studio 2022 afin de repérer les zones du code qui provoquaient cette décision et de remonter jusqu'à la source de celle-ci.

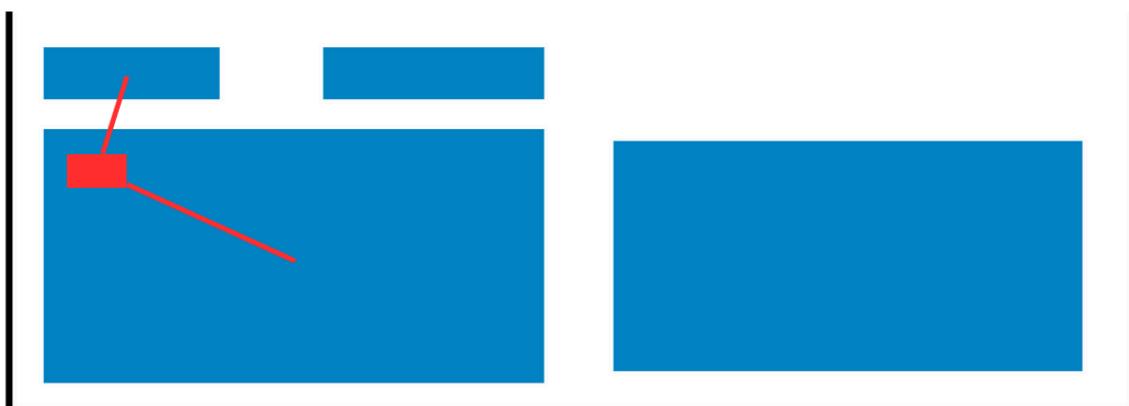
### 4.2.1 - Amélioration de l'assignation des zones fraudées

Lors du débogage du projet j'ai pu observer un problème majeur, en effet, il m'est apparu en affichant les MétaBloc qui ont été désignés comme fraudés et en les comparant au fichier XML de fraudes, qu'un certain nombre d'entre eux ne contenaient aucune surcouche de fraudes. Il a fallu, dans un premier temps, s'assurer de ce résultat, pour cela, j'ai utilisé la sortie graphique proposée par l'algorithme permettant de repérer les fraudes qui dessine sur une image des rectangles autour des zones où les surcouches ont été détectées et je les comparer aux MétaBlocs que j'avais en affichant simplement leur contenu.

Sans escompte		Mode de règlement : Virement Bancaire						
Taux d'intérêt des pénalités de retard de paiement : 3 fois le taux d'intérêt légal		Renvois : (1) Me=Mercuriale, Pr=Promotion, Oj=Opportunité du jour, @=Promotion Internet   (2) Frais de facturation						
Date d'échéance : 21/02/2025		FF (2)	Taux TVA	Code TVA	Montant HT	Montant TVA	Montant TTC	Net à Payer
Indemnité légale forfaitaire pour frais de recouvrement : 40 Euros			5,5%	03	721,78	39,70	761,48	<b>761,48</b>
Pour être libérateur, le règlement de cette créance doit être libellé à l'ordre de TRANSGOURMET Opérations Commerzbank Paris		ATTENTION NOTRE N° A CHANGE IL EST DESORMAIS NON SURTAXE MERCI DE VOUS RAPPROCHER DE VOTRE EQUIPE COMMERCIALE					TRANSGOURMET OPERATIONS SAS au capital de 15.000.000 Euros 17 rue Ferme de la Tour - CS 10005 94460 Valenton - RCS Créteil 433 927 332 N° DGDDI : AGRE 99DI054002 N° TVA intracommunautaire: FR07433927332	
Cpte: IBAN: [REDACTED]								
BIC: N° Client Date d'échéance 21/02/2025								
A ENVOYER A L'ADRESSE FIGURANT EN HAUT DE CETTE PAGE								
N° Facture	Montant (Euros)	Tournée	0017175725021					
[REDACTED]	761,48	7093	Page 1/1					

**Figure 11 : Exemple zone fraudée d'une page****Contenu du MétaBloc fraude : sans escompte****Figure 12 : Exemple affichage d'un MétaBloc fraudé avant correction**

On peut voir que la zone fraudée ([Cf. Figure 11](#)) ne contient pas le contenu du MétaBloc fraudé ([Cf. Figure 12](#)), cela est dû à un problème d'assignation de la zone fraudée, en effet, actuellement un MétaBloc est dit fraudé lorsqu'il est le plus proche MétaBloc d'un mot fraudé (lu dans le fichier XML de fraude). Pour calculer cette distance physique entre un mot et un MétaBloc, il calculait la distance entre le barycentre du mot fraudé et le barycentre de tous les MétaBlocs de la page cherchant ainsi le plus proche. Le problème est que la zone contenant le mot fraudé n'est pas obligatoirement le MétaBloc le plus proche suivant cette définition, par exemple dans l'exemple ci-dessus le mot "être" (premier mot de la zone fraudée) son barycentre est plus proche du barycentre de "sans escompte" que du barycentre de sa zone.

**Figure 13 : Schéma représentatif du problème d'assignation de zones fraudées**

Pour résoudre ce problème, j'ai décidé d'adopter en priorité une autre approche, celle-ci consiste à vérifier si le mot fraudé ne dispose pas d'un MétaBloc dans lequel son barycentre est entièrement inclus, ainsi, désormais il parcourt toujours tous les MétaBloc vérifie s'il n'est pas inclus dans celui-ci si c'est le cas ([Cf. Annexe 5](#)), alors il le garde en mémoire sinon il vérifie sa distance avec celui-ci est fait la même chose qu'avant. Enfin au résultat de la fonction n'est renvoyé que le MétaBloc dans lequel il est inclus s'il existe, sinon il renvoie toujours le plus proche.

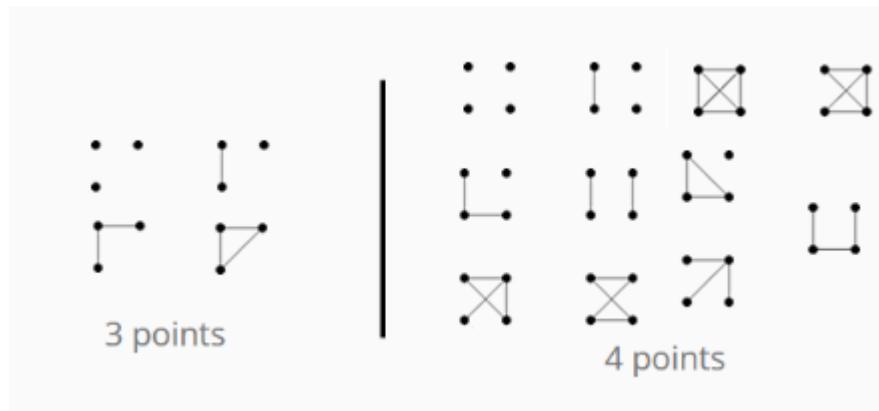
#### 4.2.2 - Amélioration du matching entre les MétaBlocs

Ensuite, comme dit précédemment il y a une étape de matching entre les MétaBloc, ce premier tri a pour but de trouver un MétaBloc ayant une bonne correspondance pour ensuite le transformer en graphe et le comparer aux graphes de l'entraînement. Auparavant, il ne cherchait pas le meilleur MétaBloc, mais le meilleur DataMatch et donc la seconde étape était de chercher dans ce DataMatch le meilleur graphe de MétaBloc. Aussi, l'algorithme était fait de sorte à ce que lors de la recherche du MétaBloc n'acceptait que le premier pouvant respecter cette condition. Cette méthode est donc imparfaite, en effet, si l'on applique cela pour des MétaBlocs cela pose un problème majeur, car le bloc sélectionné n'est peut-être pas le meilleur, cette méthode est appelée à la fois lors de l'entraînement et lors du mode mono. Cela pose un problème de par le fait que nous ne sommes pas garantie que les pages de l'entraînement et la page analysé par le mode mono contiennent le même nombre de MétaBloc et dans le même ordre, cependant, on souhaite toujours pouvoir identifier une zone fraudée redondante même si le reste du document est différent. Pour régler ce problème, la solution qui m'est apparue comme la plus évidente a été de chercher une distance minimum entre tous les MétaBloc tandis qu'avant, la méthode s'arrêtait dès lors qu'elle respectait le threshold sur les MetaBloc. Désormais la condition sur le threshold est toujours en place, mais la condition de minimum a été ajoutée afin de s'assurer d'avoir le meilleur résultat possible dans les deux cas. Cela assure de trouver le plus facilement possible une fraude qui doit être reconnue par l'entraînement et dans le cas contraire, cela permet de s'assurer qu'il n'y aurait pas pu avoir de cas plus correspondant dans la page. ([Cf. Annexe 6](#))

#### 4.2.3 - Élargissement des Graphes pas assez dense

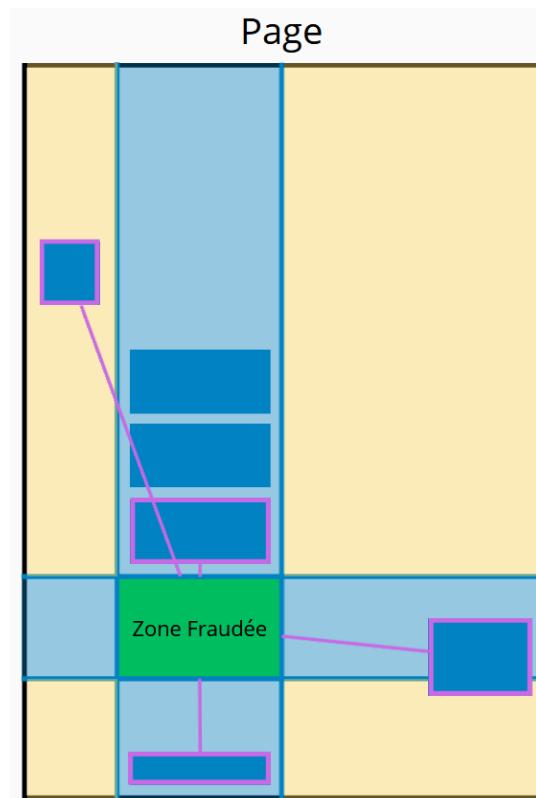
Dans certains cas, les zones fraudées pouvaient être trop petites en nombre de mots pour être correctement différenciées d'autres graphes, cela générerait donc des distances qui étaient trop faibles, car on manquait d'informations pour différencier les graphes. L'idée qui m'a été proposée par mon maître de stage était de récupérer les voisins d'un MétaBloc lorsqu'il était trop petit. Le problème initial vient du fait que lorsqu'on souhaite comparer deux entités peu importe lesquels plus

on a d'informations plus on sera précis pour différencier ces deux-là, on peut représenter le problème avec de la géométrie : le nombre de figures possibles avec trois points reliés dans l'espace, sont réduites à un (un triangle), lorsqu'on en a quatre, on peut faire soit un rectangle (qui peut avoir des propriétés et donc représenter un carré, un losange, un parallélogramme...) soit deux triangles et plus le nombre de points reliés augmente plus les possibilités de figures seront élevées et donc plus on pourra être précis sur la nature de cette figure.



**Figure 14 :** Représentation problème densité des graphes

Ainsi, le problème est le même pour les graphes, un graphe de deux nœuds comme “sans escompte” peut facilement matcher avec un autre graphe d'un autre document d'autant plus dans le contexte de l'analyse de factures des mots isolés comme “TVA” ou “montant” sont assez récurrent pouvant ainsi constituer des MétaBlocs à eux seuls. Pour pouvoir corriger ce problème lorsqu'une zone fraudée est trop petite pour pouvoir être capable de la différencier des autres fraudes, j'ai dû mettre en place un moyen pour élargir ces graphes et comme dit au début de cette section, la solution retenue était de prendre les voisins les plus proches dans chacune des 8 directions cardinales. ([Cf. Annexe 7](#))



**Figure 15 :** Schéma représentant l'élargissement d'un graphe

On crée ensuite un graphe pour chaque MétaBloc voisins qui a été récupéré et on fusionne les graphes entre eux, pour cela j'ai dû créer des fonctions capable de le faire puisque cela n'existe pas, l'idée était donc d'ajouter dans un premier temps la totalité des nœuds du graphe à ajouter puis la totalité des arêtes et enfin selon l'ordre de lecture (puisque les graphes étaient déjà orientés ainsi) on relie le dernier nœud du graphe en haut à gauche avec le premier nœud du graphe en haut, et cela, pour tous les graphes, tant qu'ils existent ([Cf. Annexe 8](#)). Ainsi, on obtient des graphes plus complexes permettant de les différencier ([Cf. Annexe 9](#)), par exemple pour le graphe "sans escompte" on passe de 2 à 60 nœuds et là où auparavant, il pouvait être considéré comme très proche de "total", un graphe composé d'un seul nœud, ce dernier passe de 1 à 109 nœuds, désormais leur distance sera bien plus importante et ils pourront être différencier de par leur nombre de nœuds très différents, nous appliquons donc ce procédé à tous les graphes dont leur ordre est inférieur à 10 (qui est une valeur qui peut évoluer et qui a été pour le moment choisie arbitrairement).

#### 4.2.4 - Erreur sur la segmentation de certaines classes

À ce stade, je suis arrivé à des résultats très convaincants, nous avons donc intégré d'autres classes. En effet, nous avons élargi la base de tests désormais nous entraînons sur 7 classes et certaines d'entre elles ne reconnaissent pas les fichiers de l'entraînement lorsqu'ils sont donnés en analyse (pourtant totalement identique). Pour régler ce problème, il a donc fallu debugger et remonter à la source du problème. À ce stade, il est apparu qu'il s'agirait d'une erreur lors de la segmentation, en effet, certains documents lorsqu'ils possèdent un type de fraude de surcouche concernant une ligne (par exemple un filigrane qui recouvre avec une partie d'une de ses lettres, un mot du document, ou bien lorsqu'un mot se superpose aux bords d'un tableau). Ce cas a été traité par l'un des précédents stagiaires, mais il semble cependant présenter quelques anomalies. La personne qui avait traité ce problème avait ajouté un paramètre dans le fichier de paramètres qui dupliquait du texte dans ce cas faussant donc les graphes ([Cf. annexe 9](#)). Ainsi avec l'aide de Romain Ravaille qui est un autre membre de l'équipe de recherche nous avons décidé après avoir analysé le problème de désactiver pour ces classes-là, ce paramètre. En le désactivant, nous avons donc effectivement réussi à supprimer la totalité de ces erreurs, cependant, nous ne savons, pour le moment, pas si l'absence de ce paramètre peut avoir des conséquences sur des classes pas encore étudiées.

#### 4.2.5 - Application de la classification à la page

Enfin, nous étions donc arrivés à des résultats ne présentant aucune erreur de classification, cependant la sortie de la solution à ce stade, permet de classifier les fraudes une à une, nous sommes donc capables de déterminer, avec un bon indice de confiance, si une fraude est connue de l'entraînement ou non. Cependant, comme dis au début de ce rapport, nous souhaitons classifier des pages qui nous sont passées en entrée de cet algorithme. La règle qui régit la classification d'une page consiste à dire qu'une page est considérée comme faux positif (donc connue de l'entraînement) que si la totalité de ses zones fraudées étaient connues de l'entraînement. Ce qui signifie que si une page comporte une fraude inconnue de la base d'entraînement alors la page est considérée comme fraudée. Pour cela, j'ai décidé d'appliquer une moyenne entre la distance de toutes les zones fraudées, puisqu'une zone totalement inconnue de la base d'entraînement aura une distance à "2 147 483 647" correspondant au plus grand entier représentable sur 32 bits, car c'est la distance graphe qui a été décidée d'être assignée aux zones fraudées n'ayant pas présenté une distance inférieure au threshold lors du premier calcul de distance. En appliquant une moyenne malgré un grand nombre potentiel de zones fraudées connues elle ne pourra jamais descendre en dessous du threshold graphe qui est pour le moment fixé à 500, pour le prouver, il suffit de faire le calcul de par combien il faudrait diviser ce chiffre pour atteindre 500 ou moins. Pour cela, il faut diviser "2 147 483 647" par 500, on trouve 4 294 968 en arrondissant au supérieur. Il

faudrait donc un total de 4 294 968 de MétaBlocs dans la page pour descendre à un coût d'environ 499,999918. En sachant qu'une page représentant une facture bien remplie comporte entre 150 et 200 MétaBloc, ce cas-ci est impossible et la moyenne apparaît donc comme être une bonne solution.

Pour se convaincre que cette méthode de classification par page fonctionne bel et bien, nous avons volontairement falsifié une page qui a servi à construire l'entraînement d'une de nos classes en ajoutant un texte dans celle-ci en plus de la fraude déjà existante. Le résultat par fraude à montrer que nous avions une fraude connue et une fraude inconnue et la moyenne a donné plus d'un milliard pour la distance de la page et la page a donc bien été classée comme inconnue (et donc vrai positif) comme montrée ci-dessous.

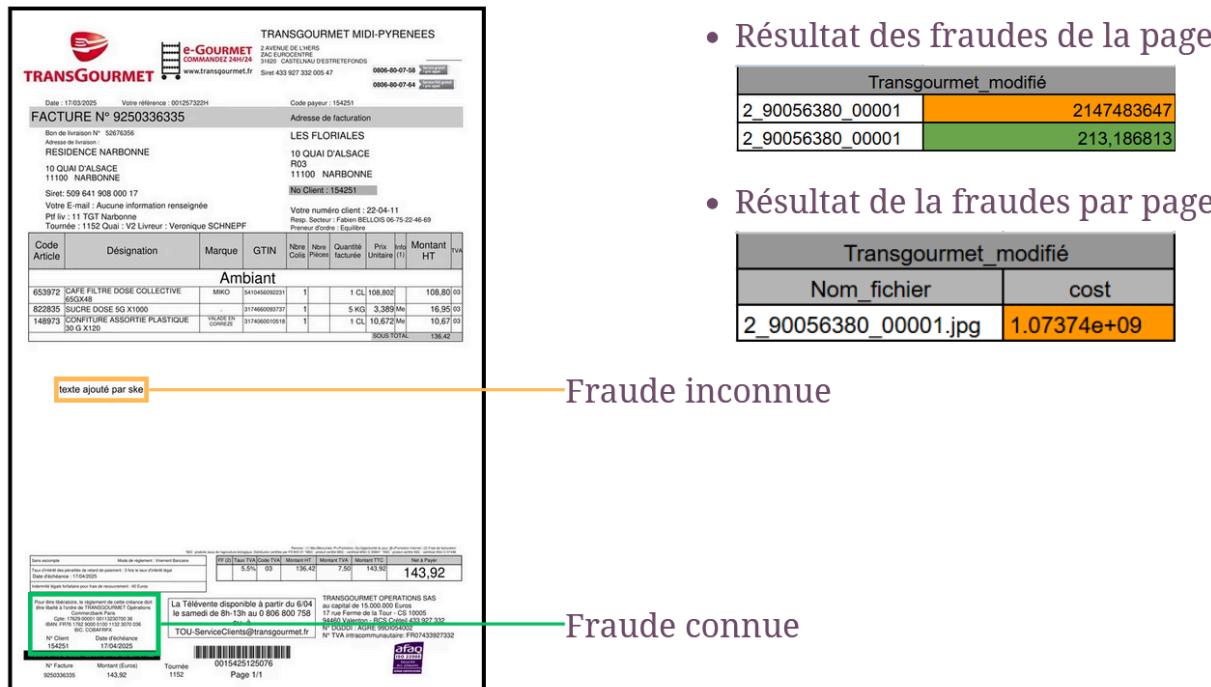


Figure 16 : résultat de la classification lors d'une fraude volontaire

### 4.3 - Perspectives à venir :

Concernant la suite du projet, durant les dernières semaines de mon stage, je vais utiliser une nouvelle approche afin d'essayer de réduire le nombre de faux positifs. Cette approche repose sur la fréquence d'apparition d'une zone dans un document. Les utilisations possibles sont multiples, par exemple, nous pouvons l'utiliser comme nouvelle méthode afin d'élargir les graphes trop petits (dont le procédé actuel a été décrit précédemment). On pourrait en effet décider de prendre les MétaBlocs les plus fréquents plutôt que les plus proches ou bien combiner les deux et décider de prendre les MétaBlocs les plus proches qui sont les plus

fréquents en fixant une limite minimale à la fréquence. Une autre utilisation possible des fréquences qui est celle sur laquelle je travaille sur la fin de mon stage est de créer un graphe des MétaBlocs les plus fréquents sur une page. L'idée est de créer des DataMatch dans l'entraînement et de faire un graphe des X DataMatch les plus fréquents où X dépend de deux critères. Premièrement, une fréquence qui devra être supérieure à une limite qui sera fixée puis un nombre minimum de nœuds souhaités. Ainsi, nous aurions un méta-graphe des zones les plus fréquentes dans un ensemble de pages, ce même procédé serait ensuite appliqué sur la page à analyser, mais cette fois-ci sur les MétaBlocs directement, car on ne peut pas calculer la fréquence d'un MétaBloc sur une même page. Enfin, comparer les deux méta-graphes avec les méthodes de comparaisons déjà programmées et définir un seuil selon lequel on les considérait comme similaires. Cela pourrait notamment être utilisé au début de l'algorithme afin de pouvoir identifier le type de document et ainsi automatiser le choix de la base d'entraînement à appliquer au fichier. Actuellement, la solution consiste à mettre tous les entraînements à la chaîne ce qui constitue une perte de temps sur proportionnellement au temps d'exécution. L'étape de la vérification humaine pour les alertes émises sera conservée et permettra donc de différencier une vraie fraude d'une fausse fraude, mais en ayant éliminé la plupart des faux positifs grâce à l'algorithme. Ce procédé permettrait donc de choisir s'il faut créer son entraînement, car il n'existe pas encore ou lequel il faut choisir dans le cas où il existerait. Une autre perspective du projet sera d'essayer de convertir la sortie de l'algorithme dans un format que l'intelligence artificielle serait capable de comprendre pour, par la suite, s'en servir afin de généraliser le modèle. Cela signifie être capable de reconnaître une classe, créer ou récupérer l'entraînement grâce à une IA plus complexe et enfin analyser les pages quand on le lui demande. Pour cela, une piste envisagée pour transformer les graphes sortis par l'application en données numériques pour qu'elle soit capable d'interpréter celle-ci serait l'utilisation d'un vecteur et d'une matrice. Le vecteur pourrait contenir la liste des nœuds du graphe et la matrice pourrait être une matrice d'adjacence triangulaire supérieure [3]. Enfin, la dernière perspective serait de réduire les doublons présents dans la base d'entraînement, en effet, actuellement un certain nombre de pages servent à créer une base d'entraînement où chacune de leurs zones fraudées est parcourue et stocker dans un fichier XML d'entraînement. Leur type est vraisemblablement le même, mais présente parfois quelques variations, il faudrait donc être capable de stocker uniquement les zones fraudées qui ne sont pas déjà présentes dans la base d'entraînement. Ce problème peut être important à résoudre puisque dans le cadre de mon stage nous avons réalisé des tests sur une faible quantité de documents présents en entraînement (une dizaine environ à chaque fois) mais que se passerait-il dans l'éventualité où cette base présenterait plusieurs milliers de fichiers. Le fichier XML serait lourd et le temps d'exécution serait décuplé.

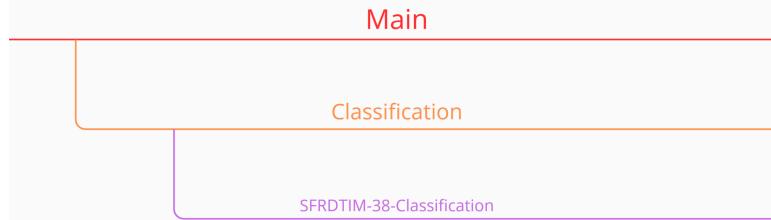
## 5 - Méthodologie et organisation du projet

Pour l'organisation des projets, l'entreprise utilise Jira qui est un outil permettant de découper les projets en tâches et de planifier celles-ci, tous les matins nous avons des daily meeting avec les membres de l'équipe dirigé par Saddok Kébairi où nous présentons les tâches que nous avons réalisé lors de la journée précédente et nous mettons à jour le tableau. Celui-ci est composé de trois colonnes : "à faire", "en cours" et "finis", ainsi lors du daily nous mettons dans "finis" les tâches qui ont été réalisées, dans "en cours" les tâches sur lesquelles nous allons travailler aujourd'hui et nous rajoutons dans "à faire" les nouvelles tâches qui seront à faire par la suite.

The screenshot shows the Jira interface for the project 'Tableau SFRDT'. The left sidebar shows project navigation options like 'Votre travail', 'Projets', 'Filtres', 'Tableaux de bord', 'Équipes', 'Appli', 'Créer', 'Rechercher', and team information. The main area displays a board with three columns: 'A FAIRE 5', 'EN COURS 4', and 'FINI 12'. The 'A FAIRE' column contains tasks such as 'Détection des incohérences dans les documents PDF • Partie 1 • Finaliser le « Graph Matching » • Tests de validation de l'approche • Packager le code source en un service API REST • Outils de dev : Visual Studio C++, Python' and 'jpg en dur à retirer afin de l'adapter selon le format de l'image en cours'. The 'EN COURS' column contains tasks like 'Regrouper les zones du training "représentatives" dans un graph' (with a checked checkbox for 'SFRDTIM-68') and 'Formater les 11 classes correctement' (with a checked checkbox for 'SFRDTIM-71'). The 'FINI' column contains tasks such as 'Récupérer les métablocs fréquents dans le training et les stocker dans le XML' (with a checked checkbox for 'SFRDTIM-69'), 'Comprendre et corriger les fichiers qui match avec Nescafé' (with a checked checkbox for 'SFRDTIM-64'), and 'Cibler le métabloc le plus proche'. The sidebar also includes sections for 'PLANIFICATION' (Résumé, Chronologie) and 'Tableau' (Calendrier, Liste, Formulaires, Objectifs).

**Figure 17 :** Tableau Jira du projet de mon stage

Le projet est aussi versionné sur GitLab, le projet se nomme `SupplierLayoutVerifier` il dispose d'une branche principale "main" de laquelle découlent plusieurs autres branches. Celle sur laquelle se situe ma branche se nomme "classification" et j'ai donc créé une branche "SFRDTIM-38-Classification" dessus.



**Figure 18 :** Schéma représentant l'architecture des branches du projet

Durant la première moitié de mon stage, j'ai été en autonomie afin de comprendre au mieux le projet je me documentais beaucoup sur le C++ et le principe objet en C++ puis j'ai réalisé le refactor du code en autonomie également. Durant la seconde moitié du stage il a été décidé qu'un des membres de l'équipe de recherche, Romain Ravaille, assurerait plus en détails mon projet notamment en organisant une réunion quotidienne d'une heure environ en début d'après-midi pour plusieurs raisons parmi lesquelles, mes difficultés à avancer à la vitesse prévu, mais également, car c'est un chercheur spécialisé dans les Mathématiques et ses applications ce qui correspond aux thématiques de mon stage orientées sur les graphes et l'utilisation prochaine de l'intelligence artificielle. Cette décision m'a effectivement permis de progresser bien mieux, cela m'a permis de clarifier mes idées et de recevoir un soutien supplémentaire notamment sur la partie mathématique du stage. Ainsi, j'ai rapidement pu me mettre à niveau et discuter de nouvelles idées avec lui, ce qui nous permettait de mettre en commun nos idées et de tirer le meilleur de chacune d'entre elles.

## 6 - Conclusion

En conclusion, le sujet de mon stage qui portait sur la réduction de faux positifs a bien progressé, il reste une étape de validation en volume et de généralisation. En effet, j'ai refactorisé l'application, et contribué à réduire les faux positifs ce qui est en bonne voie. Tout au long de mon stage, la méthodologie pour la réalisation des tests consistait à développer une fonctionnalité puis à d'abord tester qu'il n'y avait pas de régression sur les cas validés, puis tester les cas que nous souhaitions régler. Pour ce qui est des statistiques au début du stage, il n'existe pas de résultats que sur une seule classe de document, donc voici la table de confusion :

Training Transgourmet (threshold = 1000) première version			
	Vérité terrain		
Résultat algo	Transgourmet	Autres	
Transgourmet	36	22	
Autres	52	102	

**Figure 19** : Table de confusion pour la classe Transgourmet

Nous avions donc 41% des documents avec des fraudes type Transgourmet qui étaient reconnus comme tel sur 88 documents et 82% des documents présentant un autre type de fraude sur 124 et qui étaient reconnus comme tel. Aujourd'hui nous avons pu tester sur plus de classes et les résultats sont prometteurs puisqu'on a cinq classes où 100% des fraudes de cette même classe sont détectées et aucune erreur de détection sur d'autres classes. 2 classes restantes présentent quelques erreurs avec entre 5 et 15% d'erreurs sur des classes en tout genre, les 2 semaines restantes de mon stage seront l'occasion de réduire ces cas-là et de réaliser davantage de tests. Ces résultats montrent qu'une très grande partie des documents sont classifiés correctement réduisant donc grandement le coût humain obligatoire afin de faire une deuxième vérification entre vraies fraudes et fausses fraudes.

	Transgourmet	Nescafé	Techniccar	Svp	Myosotis	Hess	Herast	Nombres de documents
Transgourmet	100	0	0	0	0	0	0	74
Nescafé	0	100	0	0	0	0	0	22
Techniccar	0	0	100	0	0	0	0	15
Svp	0	0	0	100	0	0	0	9
Myosotis	0	0	0	0	100	0	0	10
Hess	0	0	0	0	0	100	0	14
Herast	0	0	0	0	0	0	100	2
Autres	0	0	0	0	0	82,97	95,13	41

**Figure 20** : Tableau répertoriant les résultats des tests (en %)

J'ai également débuté la rédaction d'une documentation qui sera aboutie à la fin du stage, pour que le projet soit reprenable. Pour ce qui est des compétences acquises j'ai pu apprendre à :

- Comprendre et à me réapproprier le code d'un projet complexe afin de l'améliorer
- Gérer un projet informatique et gérer les contraintes de temps
- Communiquer efficacement l'avancé d'un projet, les contraintes et les difficultés rencontrées
- Travailler en équipe sur un projet informatique

Pour illustrer chacune de ces compétences, l'apprentissage du code et le refactoring de celle-ci m'a permis de m'approprier un nouveau langage dans le contexte d'un projet complexe. Les daily meeting, les réunions bi-mensuelle ainsi que les réunions ponctuelles m'ont appris à gérer le temps et l'avancé du projet sur lequel j'ai travaillé ainsi qu'à communiquer sur les avancés et les difficultés du projet. Enfin ma collaboration avec un membre de l'équipe du Département de la Recherche m'a permis d'apprendre à travailler plus efficacement en équipe.

## Signatures

Vu le

## 7 - Références

- [1] "Etude Fraude 2022 - Trustpair, Accenture et option Finance," *Trustpair*.  
<https://trustpair.fr/etude-fraude-2022-accenture-sap-trustpair/>
- [2] "Detection Techniques of Fraud in Accounting | European Journal of Economics and Business Studies", *revistia.com*  
<https://revistia.com/index.php/ejes/article/view/5219>
- [3] "Deep Learning on Graphs: A Survey", Ziwei Zhang, Peng Cui and Wenwu Zhu - 2015  
[\[1812.04202\] Deep Learning on Graphs: A Survey](#)

## 8 - Annexes

### Annexe 1 : Exemple fichier XML contenant la couche texte

```
<?xml version="1.0" encoding="UTF-8"?>
<kocr DataBottom="3447" DataLeft="104" DataRight="2357" DataTop="81" Dpi="300" Version="10" bottom="3508" lastid="3535" 
|   <block bottom="817" id="1" left="1468" right="1877" streamId="0" top="729">
|     <line bottom="764" id="2" left="1471" right="1825" top="729">
|       <word bottom="763" confidence="255" font="MP" id="3" left="1471" right="1533" top="730" value="ILE">
|         <char bottom="763" confidence="255" id="4" left="1471" right="1476" top="730" value="I"/>
|         <char bottom="763" confidence="255" id="5" left="1483" right="1504" top="730" value="L"/>
|         <char bottom="763" confidence="255" id="6" left="1509" right="1533" top="730" value="E"/>
|       </word>
|       <word bottom="764" confidence="255" font="MP" id="7" left="1552" right="1640" top="729" value="DES">
|         <char bottom="763" confidence="255" id="8" left="1552" right="1579" top="730" value="D"/>
|         <char bottom="763" confidence="255" id="9" left="1585" right="1609" top="730" value="E"/>
|         <char bottom="764" confidence="255" id="10" left="1614" right="1640" top="729" value="S"/>
|       </word>
|       <word bottom="763" confidence="255" font="MP" id="11" left="1659" right="1825" top="730" value="EMBIEZ">
|         <char bottom="763" confidence="255" id="12" left="1659" right="1683" top="730" value="E"/>
|         <char bottom="763" confidence="255" id="13" left="1689" right="1721" top="730" value="M"/>
|         <char bottom="763" confidence="255" id="14" left="1728" right="1752" top="730" value="B"/>
|         <char bottom="763" confidence="255" id="15" left="1759" right="1763" top="730" value="I"/>
|         <char bottom="763" confidence="255" id="16" left="1771" right="1795" top="730" value="E"/>
|         <char bottom="763" confidence="255" id="17" left="1799" right="1825" top="730" value="Z"/>
|       </word>
|     </line>
|   </block>
</kocr>
```

### Annexe 2 : Exemple fichier XML contenant les fraudes de surcouche

```
<page idx="1">
  <frd_area tag="404">
    <data tag="404" class="FRD_WORD_BB">148,3067,189,3086</data>
    <data tag="404" class="FRD_WORD_VAL">être</data>
    <data tag="404" class="ORG_WORD_BB">144,3076,422,3114</data>
    <data tag="404" class="ORG_WORD_VAL">ATTENTION</data>
    <data tag="404" class="CONFIDENCE">75</data>
    <data tag="404" class="TAG ">404</data>
  </frd_area>
  <frd_area tag="404">
    <data tag="404" class="FRD_WORD_BB">199,3067,260,3086</data>
    <data tag="404" class="FRD_WORD_VAL">libellé</data>
    <data tag="404" class="ORG_WORD_BB">144,3076,422,3114</data>
    <data tag="404" class="ORG_WORD_VAL">ATTENTION</data>
    <data tag="404" class="CONFIDENCE">75</data>
    <data tag="404" class="TAG ">404</data>
  </frd_area>
</page>
```

## Annexe 3 : Exemple fichier XML contenant les paramètres d'une classe

```
<?xml version="1.0" encoding="UTF-8"?>
<SUPPLYLAYOUTVERIFIER>
    <BATCH_NAME>techniccar</BATCH_NAME>
    <SETTINGS>

        <SEGMENTATION>
            <MAXGAP> <!-- pixel gap treshold for aggregation of LineFragments in MetaBlocks -->
                <X>5</X>
                <Y>128</Y>
            </MAXGAP>
            <MAXSHIFT> <!-- pixel gap treshold for aggregation of LineFragments in MetaBlocks -->
                <LEFT>210</LEFT>
                <RIGHT>5</RIGHT>
            </MAXSHIFT>
            <CENTERSHIFTCOEF>200</CENTERSHIFTCOEF> <!-- coefficient : smaller value result in easier aggregation -->
            <MAXRATIOFPAGEWIDTH>50</MAXRATIOFPAGEWIDTH> <!-- value between 0 and 100 -->
            <MAXFONTSIZERATIO>1.5</MAXFONTSIZERATIO> <!-- max gap treshold for aggregation of LineFragments in MetaBlocks -->
            <LINEEXTRACTOR>
                <HORIZONTAL>
                    <MINLENGTH>400</MINLENGTH>
                    <GAP>2</GAP> <!-- max gap treshold for aggregation of two lines in one -->
                    <THICKNESSFILTER>1</THICKNESSFILTER>
                </HORIZONTAL>
                <VERTICAL>
                    <MINLENGTH>60</MINLENGTH>
                    <GAP>2</GAP>
                    <THICKNESSFILTER>1</THICKNESSFILTER>
                </VERTICAL>
            </LINEEXTRACTOR>
            <ALLOWBARCODE>0</ALLOWBARCODE> <!-- 1 = allowed / 0 = not allowed -->
            <ENABLELINECUT>0</ENABLELINECUT> <!-- 1 = enabled / 0 = disabled -->
        </SEGMENTATION>

        <DATA_MATCH>
            <DISTANCE_THRESHOLD>1000</DISTANCE_THRESHOLD> <!-- smaller value result in more match and more matching errors -->
            <STABILITY_THRESHOLD>80</STABILITY_THRESHOLD> <!-- value between 0 and 100 to consider a DataMatch as stable -->
        </DATA_MATCH>

        <CALSSIFICATION>
            <GRAPH_MATCHING_THRESHOLD>400</GRAPH_MATCHING_THRESHOLD>
        </CALSSIFICATION>
    </SETTINGS>
</SUPPLYLAYOUTVERIFIER>
```

## Annexe 4 : Exemple de fichier XML entraînement

```
<?xml version="1.0" encoding="UTF - 8"?>
<FAKE_POSITIVES_TRAINING>
    <FP index="0" isRepresentative="false">
        <Metablock Left="101" Top="3038" Right="774" Bottom="3339">
            <Header></Header>
            <X_Index>0</X_Index>
            <Y_Index>-1</Y_Index>
            <LineNbMetablock>3</LineNbMetablock>
            <FontSize>25.515625</FontSize>
            <NbChar>344</NbChar>
            <TypeTable>
                <Type name="0">false</Type>
                <Type name="1">false</Type>
                <Type name="2">false</Type>
            </TypeTable>
            <values>PasdecommandeetLivraisonles25/12etPourretolibératoire,lerèglementdecettecréance doitêtrelibelléàl'ordrec
        </Metablock>
    <Graph order="70" size="112">
        <Node index="1" val="Pas" isFraud="false" left="148" top="3075" right="247" bottom="3122" fontSize="47">
            <Edge target="2" distance="104" overlap="0"/>
            <Edge target="67" distance="311" overlap="0"/>
        </Node>
        <Node index="2" val="de" isFraud="false" left="269" top="3075" right="334" bottom="3122" fontSize="47">
            <Edge target="3" distance="207" overlap="0"/>
            <Edge target="67" distance="207" overlap="0"/>
        </Node>
        <Node index="3" val="commande" isFraud="false" left="356" top="3075" right="661" bottom="3122" fontSize="47">
            <Edge target="4" distance="199" overlap="0"/>
        </Node>
    </Graph>
</FAKE_POSITIVES_TRAINING>
```

## Annexe 5 : Fonctions permettant de vérifier qu'un mot est inclus dans un MétaBloc

```

for (MetaBlock* m : *plstAMetaBloc)
{
    distance = frd->dataMatchDistance(m->BarycentreX, m->BarycentreY);
    if (m->isIncluded(frd) && m->getArea(m->getLeft(), m->getTop(), m->getRight(), m->getBottom()) < minTaille) {
        matchIncluded = m;
        minTaille = m->getArea(m->getLeft(), m->getTop(), m->getRight(), m->getBottom());
    }
    else {
        if (distance < minDistance) {
            minDistance = distance;
            bestMatch = m;
        }
    }
}
if (matchIncluded != nullptr) {
    matchIncluded->hasIncoherences = true;
    matchIncluded->addFraudOverlap(frd);
    frd->setParentMetablock(matchIncluded);
    cout << "match included : " << matchIncluded->getValue() << endl;
}
else if (bestMatch != nullptr) {
    bestMatch->hasIncoherences = true;
    bestMatch->addFraudOverlap(frd);
    frd->setParentMetablock(bestMatch);
    cout << "match best : " << bestMatch->getValue() << endl;
}

bool MetaBlock::isIncluded(FraudOverlap* frd) {
    if (frd->getCenterX() > getLeft() && frd->getCenterX() < getRight() && frd->getCenterY() > getTop() && frd->getCenterY() < getBottom()) {
        return true;
    }
    return false;
}

```

## Annexe 6 : Partie du code dans laquelle on cherche le minimum lors du matching par MétaBloc

```

trainingPair->size();
float minimum = distanceThreshold;
DataMatch* bestDM = nullptr;
//we find if the dataMatch has already been seen while training
MetaBlock* mb1 = this->getMetablocks()->at(0);
vector<bool>* templateTypeTable = mb1->setTypeTable(mb1->Value);
double avgFontSize = mb1->getAvgFontSize();

ofstream* logFile = new ofstream(outputDirectory + "_logs.txt", std::ios::app);

for (map<DataMatch*, vector<Graph*>>::iterator it = trainingPair->begin(); it != trainingPair->end(); ++it) {
    DataMatch* dm = it->first;
    if (dm->getMetablocks()->size() > 0 && this->getMetablocks()->size() > 0)
    {
        MetaBlock* mb2 = dm->getMetablocks()->at(0);
        float dist = mb1->distCalculator(mb2, FLT_MAX, avgFontSize, templateTypeTable);
        if (dist <= minimum)
        {
            minimum = dist;
            bestDM = dm;
        }
    }
}

```

## Annexe 7 : Partie du code permettant de repérer les voisins d'un MétaBloc

```

vector<MetaBlock***>* plstMB = parentPage->getListMetaBlocks();
for (vector<MetaBlock**>::iterator it = plstMB->begin(); it != plstMB->end(); ++it) {
    MetaBlock* mb = *it;
    if (this != mb) {
        // vérifie que le bloc soit dans l'intervalle de hauteur du MetaBloc de base
        if ((BarycentreX - mb->BarycentreX < threshold || mb->BarycentreX - BarycentreX < threshold) && mb->BarycentreY >= yFrontiereMinimum && mb->BarycentreY <= yFrontiereMaximum) {
            int distanceHorizontal = INT_MAX;
            // permet de voir si le bloc se trouve à droite du MetaBloc de base
            if (mb->BarycentreX > BarycentreX) {
                int xSuperieur = mb->BarycentreX - (mb->Width / 2);
                int xInferieur = xFrontiereMaximum;
                distanceHorizontal = xSuperieur - xInferieur;
                if (distanceHorizontal < distanceHorizontalDroiteMinimum) {
                    distanceHorizontalDroiteMinimum = distanceHorizontal;
                    mbDroite = mb;
                }
            }
            // permet de voir si le bloc se trouve à gauche du MetaBloc de base
            else if (mb->BarycentreX < BarycentreX) {
                int xSuperieur = xFrontiereMinimum;
                int xInferieur = mb->BarycentreX + (mb->Width / 2);
                distanceHorizontal = xSuperieur - xInferieur;
                if (distanceHorizontal < distanceHorizontalGaucheMinimum) {
                    distanceHorizontalGaucheMinimum = distanceHorizontal;
                    mbGauche = mb;
                }
            }
        }
    }
}

```

## Annexe 8 : Partie du code permettant de construire un méta-graphe à partir des voisins d'un MétaBloc

```

if (nearMetaBlocks->find("mbHG")->second->second != nullptr) {
    graphHG = nearMetaBlocks->find("mbHG")->second->second->initMetaGraph();
    lastNode = g->getOrder();
    lastEdge = g->getSize();
    g->addGraph(graphHG);
    if (isFirst) {
        isFirst = false;
    }
    else {
        g->addEdge(g->getPoint(lastNode), g->getPoint(lastNode + 1), lastEdge + 1);
    }
}
if (nearMetaBlocks->find("mbH")->second->second != nullptr) {
    graphH = nearMetaBlocks->find("mbH")->second->second->initMetaGraph();
    lastNode = g->getOrder();
    lastEdge = g->getSize();
    g->addGraph(graphH);
    if (isFirst) {
        isFirst = false;
    }
    else {
        g->addEdge(g->getPoint(lastNode), g->getPoint(lastNode + 1), lastEdge + 1);
    }
}

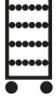
```

## Annexe 9 : Exemple de résultat lors de la récupération des voisins d'un MétaBloc

**TRANSGOURMET**



**e-GOURMET**  
COMMANDÉZ 24H/24



www.transgourmet.fr

ZA Ecopole  
CS 70054  
13558 ST MARTIN DE CRAU - CEDEX

Siret 433 927 332 004 48

Code payeur : 1067910

N° Indigo 0826-10-17-04  
0,15€ TTC / MIN

N° Indigo FAX 0826-10-17-05  
0,15€ TTC / MN

Date : 02/12/2024      Votre référence : wedg

**FACTURE N° 141222448**

Bon de livraison N° 51654408

Adresse de livraison :

CONTROLE ESAT JEAN MEDECIN HP  
65 AVENUE HENRI MATISSE  
06300 NICE

Siret: 775 552 268 000 85

Votre E-mail : smaria@adapeiam.fr

Ptf liv : 89 TGT Le Muy ( Agrément:F83-086-10CE)  
Tournée : 8908 Quai : 62 Livreur : Moumni TGT ZINE

Code payeur : 1067910

Adresse de facturation

ESAT JEAN MEDECIN HP  
44/46 AVENUE DENIS SEMERIA  
06300 NICE

No Client : 1067910

Votre numéro client : 31107062761  
Resp. Secteur : Eric FERRI 06-32-30-87-11  
Preneur d'ordre : Jennifer C

Code Article	Désignation	Marque	GTIN	Nbre Colis	Nbre Pièces	Quantité facturée	Prix Unitaire	Info (1)	Montant HT	TVA
<b>Frais</b>										
971404	CLEMENTINE CORSE 3 IGP FR.CAT1 12KG	STKFL		1		12 KG	3,690	Pr	44,28	03
SOUS TOTAL										44,28

**mbHD**

**mbM** sans escompte      **mbD** Mode de règlement : Prelevement

Taux d'intérêt des pénalités de retard de paiement : 3 fois le taux d'intérêt légal  
Date d'échéance : 31/12/2024  
Indemnité légale forfaitaire pour frais de recouvrement : 40 Euros

**mbHD**

BIO : produits issus de l'agriculture biologique. Distribution certifiée par FR BIO 01 "MSC" : produit certifié MSC - certificat MSC C 5947 "ASC" : produit certifié ASC - certificat ASC C 0144d

FF (2)	Taux TVA	Code TVA	Montant HT	Montant TVA	Montant TTC	Net à Payer
	5,5%	03	44,28	2,44	46,72	<b>46,72</b>

Pour être libérateur, le règlement de cette créance doit être libellé à l'ordre de TRANSGOURMET Opérations  
Comptoirbank Paris  
Cpte: 17629 00001 00113231500 61  
IBAN: FR76 1762 9000 0100 1132 3150 061  
BIC: COBAFRPX

N° Client Date d'échéance  
1067910 31/12/2024

LES MERCREDIS 25/12 ET 01/01/2025  
ETANT FERIES PAS DE PRISE DE  
COMMANDE PAS DE LIVRAISON  
CONTACTER VOS EQUIPES COMMERCE

TRANSGOURMET OPERATIONS SAS  
au capital de 15.000.000 Euros  
17 rue Ferme de la Tour - CS 10005  
94460 Valenton - RCS Créteil 433 927 332  
N° DGDDI : AGRE 99DI054002  
N° TVA intracommunautaire: FR07433927332

A ENVOYER L'ADRESSE FIGURANT EN HAUT DE CETTE PAGE

N° Facture Montant (Euros) Tournée 0106791024337

141222448 46,72 8908

Page 1/1