Table des matières

Table des illustrations

Glossaire et abréviations

APA	American Psychological Association
DB	Base de données
Epic	Grande user story regroupant beaucoup de travail
EF	Entity Framework
FST	Fondation Suisse pour les Téléthèses
IT	Information Technology
KNIME	Konstanz Information Miner
LINQ	Language Integrated Query
n-uple	Elément du produit cartésien de n ensembles
NuGet	Gestionnaire de packages Microsoft Visual Studio
ORM	Object relational mapping
SInf	Service Informatique
SSIS	SQL Server Integration Services
Stemming	Réduire les mots à leur racine
User Story	Unité de délimitation de travail agile
TB	Travail de Bachelor
WCF	Windows Communication Fondation
Workflow	Séquence d'étapes d'analyse nécessaires pour atteindre un but donné

1 Introduction

De nos jours, un nombre sans cesse croissant de personnes sont confrontées de façon quotidienne à des tâches d'écriture sur un support informatisé. Nous passons en moyenne 28% de notre semaine de travail à lire ou écrire des e-mails, cela représente 13 heures par semaine ou encore 650 h par an, et ce chiffre ne prend pas en compte les divers travaux rédactionnels que nous sommes amenés à effectuer. *(Washington Post, 2012)*

Imaginez maintenant que vous souffriez d'un trouble de la lecture et de l'écriture comme 7% de la population *(Kooij, 2013),* et le temps supplémentaire que vous consacrez à des activités d'écriture simple peut vite devenir substantiel.

De par ces faits, des solutions sans cesse novatrices doivent être trouvées afin d'influencer de manière positive l'efficacité rédactionnelle de tout un chacun.

1.1 Contexte

Afin de répondre aux difficultés rencontrées par les élèves dyslexiques en classe, la Fédération Suisse pour les Téléthèses a crée en 2016 le logiciel PhonoWriter qui les assiste dans la rédaction de textes orthographiquement et grammaticalement corrects en se basant sur la prédiction de mot, améliorée et enrichie par la même mécanique qu'un logiciel de reconnaissance vocale, mais appliquée à l'écrit de l'utilisateur.

Dans le cadre de notre travail de Bachelor, nous avons reçu la mission d'étendre les spécifications du logiciel PhonoWriter en enrichissant sa capacité à effectuer de la prédiction de phrase ainsi que de mot suivant dans le but de généraliser l'application pour tous, en garantissant la compatibilité de la solution avec la structure actuelle.

1.2 Objectifs

Les objectifs de ce travail sont les suivants :

- Effectuer un état de l'art des solutions de correction orthographique et de prédiction de mot / mot suivant.
- Proposer sous forme de web service une infrastructure fournissant des propositions de mots suivants basés sur un mot ou un bout de phrase.
- Permettre à cette infrastructure de s'enrichir en fonction du vocabulaire utilisé par l'utilisateur, en lui fournissant un service de prédiction plus personnalisé.
- Donner des ouvertures sur les manières de traiter le texte plus en détail pour fournir des prédictions de plus en plus précises.

2 Etat de l'art – Applications

Dans ce chapitre, nous aborderons trois logiciels qui sont actifs dans le domaine de la prédiction de phrase, y compris le logiciel PhonoWriter qui constitue la base de ce travail de Bachelor. Nous extrairons les idées principales qui les constituent puis nous nous intéresserons à leur fonctionnement. Nous déterminerons par la suite dans quelle mesure nous pouvons utiliser ces informations dans notre prototype.

2.1 PhonoWriter

Ce logiciel développé par la Fédération Suisse pour les Téléthèses est un logiciel d'autocomplétion et de correction de textes. Il se distingue des autres solutions équivalentes par le fait qu'il ne se base pas uniquement sur les caractères tapés pour prédire un mot, mais il effectue une agrégation de différents types de prédictions pour atteindre un niveau de justesse plus élevé.

Il utilise pour se faire trois types de prédictions puis soumet les propositions générées à partir d'un modèle mathématique probabilistique à l'utilisateur qui les choisi à l'aide d'un outil visuel appelé prédiction pictographique.

Nous allons maintenant aborder les différents types de prédictions sur lesquels PhonoWriter se base.

2.1.1 Prédiction classique

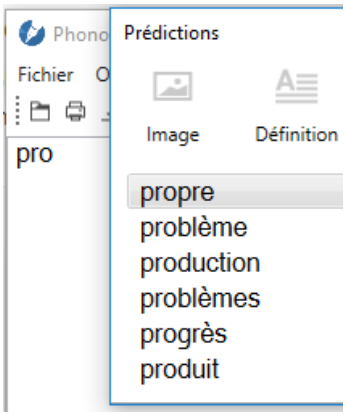

Figure 1 - Prédiction classique PhonoWriter

La prédiction de mot sous sa forme la plus simple et basique : à partir des lettres entrées, le logiciel propose tous les mots du dictionnaire contenant ces lettres

2.1.2 Prédiction phonétique

L'algorithme de prédiction se base cette fois-ci sur une correspondance entre la consonnance du mot en cours de rédaction et celle des mots du dictionnaire, cela permet tout particulièrement de gérer les fautes liées à des lettres à consonnance semblables (i-y, s-z, v-w notamment)

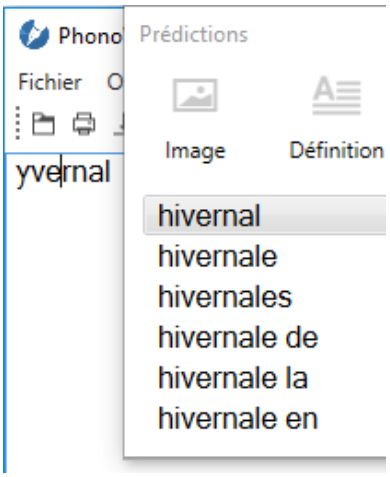

Figure 2 - Prédiction phonétique PhonoWriter

2.1.3 Prédiction floue (Fuzzy)

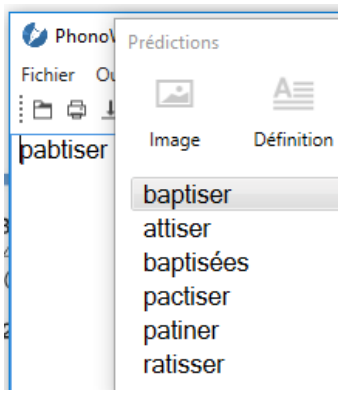

Figure 3 - Prédiction floue PhonoWriter

Cette méthode de prédiction est particulièrement efficace pour détecter les inversions de lettres courantes chez les personnes atteintes de dyslexie. Elle se base sur des modèles connus référençant ces erreurs.

2.1.4 Validation par l'utilisateur

Cet outil a deux objectifs ; il permet dans un premier temps de désambiguïser les homonymes, et dans un second temps de proposer une illustration pour les enfants dans un but éducatif et un peu plus ludique

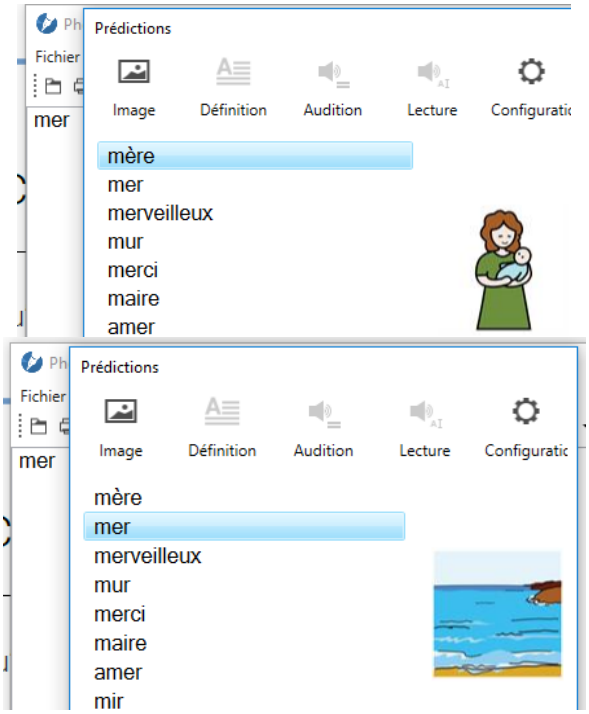

Figure 4 - Désambiguïsation pictographique PhonoWriter

2.1.5 Fonctionnement des prédictions

Comme spécifié dans la description du logiciel, PhonoWriter utilise par la suite une agrégation des différentes prédictions qui sont ensuite pondérées afin de déterminer lesquelles sont les plus pertinentes et par conséquent l'ordre de suggestion dans la liste.

Voici un exemple de fonctionnement : le mot « stat » est entré.

Mot suggéré	Pondération	Type de Prédiction	Signification de la pondération
star	3	FUZZY	La valeur représente la distance entre les deux mots. Plus elle est basse, plus ils sont semblables.
saute	4	FUZZY	
stase	4	FUZZY	
station	22	CLASSIC	La valeur est une expression de la fréquence d'apparition dans la DB. Plus un mot est hautement pondéré, plus il est fréquent.
statistiques	24	CLASSIC	
statut	28	CLASSIC	
date	0.4021414	PHONETIC	La valeur entre 0 et 1 est la pertinence phonétique telle que retournée par la reconnaissance vocale. Une valeur de 1 représente un mot avec une acoustique identique.
teinte	0.2649476	PHONETIC	
se teinte	0.5053205	PHONETIC	

Toutes ces prédictions seront traitées par un algorithme qui produira une pondération globale basée sur les valeurs présentées ci-dessus. Dans ce cas de figure, la suggestion sera le mot « date ».

2.1.6 Paramétrisation

En dehors des options de paramétrisation plus superficielles comme la lecture audio des mots entrés, leur taille ou encore leur police, il est possible de modifier les paramètres de l'algorithme de suggestion de mot.

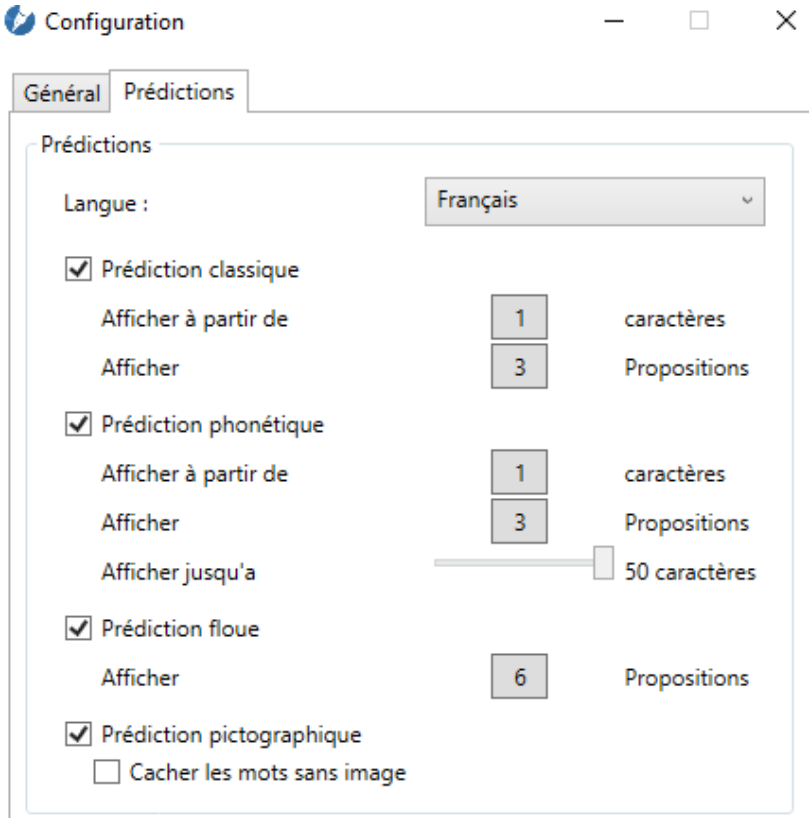

Figure 5 - Onglet de paramétrisation des prédictions

Nous pouvons obtenir deux catégories principales de paramétrisation : le nombre de prédictions à afficher par catégorie de prédictions (les trois meilleures correspondances), et à partir de combien de caractères nous devons considérer l'une ou l'autre prédiction.

L'utilisateur peut par exemple activer la prédiction classique à partir d'un caractère pour bénéficier d'une autocomplétion standard et afficher les suggestions issues de la phonétique uniquement à partir de 5 caractères.

Cela permet à l'utilisateur de ne pas se retrouver avec un nombre trop important de suggestions pour des mots en apparence simples.

2.2 PHRASE EXPRESS

Proposé uniquement en anglais, ce logiciel développé par la société *Bartels Media GmbH* ne propose pas de correction orthographique à proprement parler, mais se place dans le domaine des raccourcis de frappe dans le but de gagner en vitesse lors de la rédaction de documents au quotidien.

La façon de fonctionner du logiciel nécessite toutefois un temps de préparation de la part de l'utilisateur avant de pouvoir l'utiliser pleinement. Ce-dernier doit avant toute chose définir les raccourcis ainsi que les phrases qu'il utilise le plus souvent.

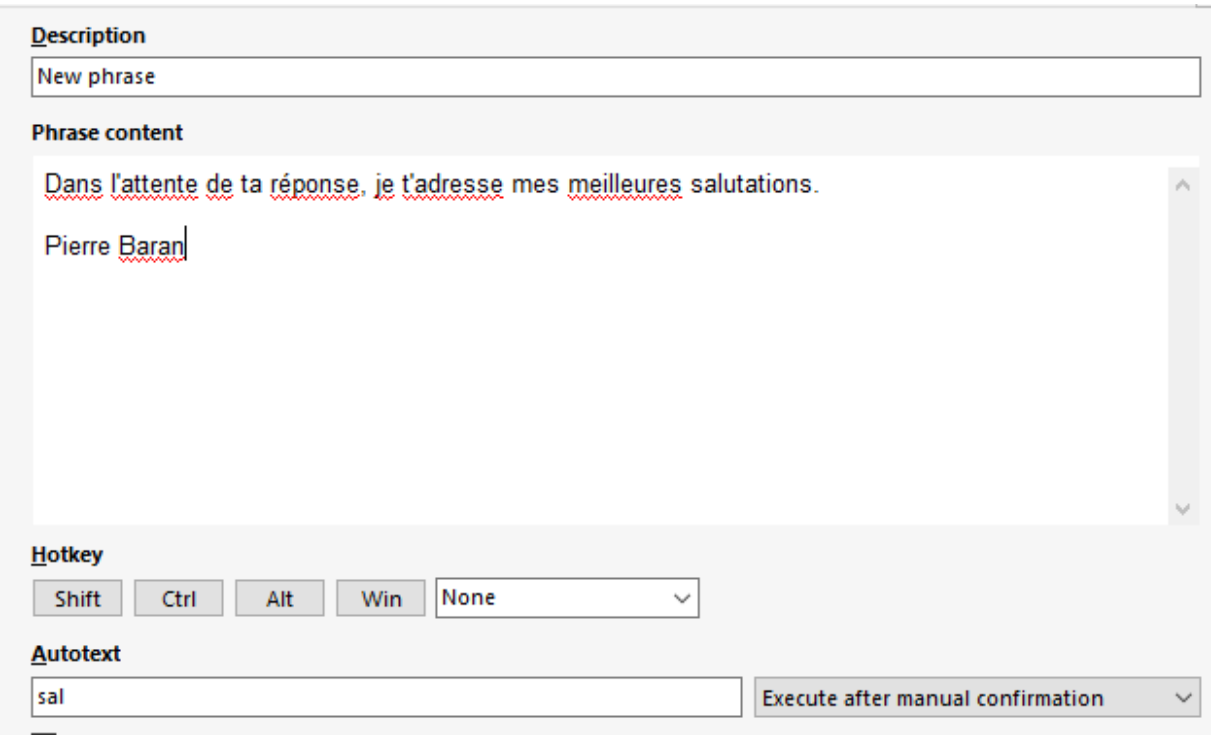

Figure 6 - Création de phrase rapide PHRASE EXPRESS

Dans l'image ci-dessus, nous pouvons voir l'écran de création pour les phrases rapides. Pour cet exemple, nous avons créé un raccourci qui s'activera lorsque nous entrerons le mot « sal », dans le but de rédiger un mail plus rapidement.

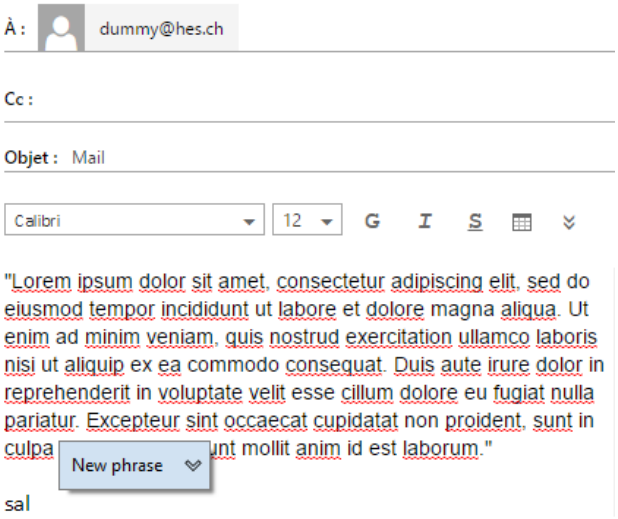

Figure 7 - Suggestion de phrase PHRASE EXPRESS

Comme le montre la figure ci-dessus, lors de l'écriture du mot clé « sal », le programme nous suggère automatiquement une autocomplétion. Un clic suffira pour remplacer le mot clé par la phrase entière pré-enregistrée.

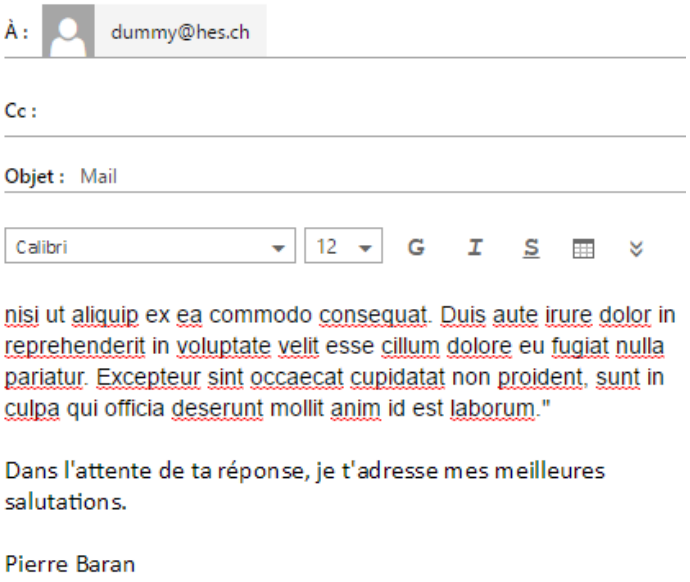

Figure 8 - Phrase générée PHRASE EXPRESS

L'idée de base de ce logiciel est très intéressante dans le cadre de notre travail, car cela permettra à l'utilisateur final d'avoir une prédiction de phrase qui prendra en compte ses phrases favorites.

2.3 Skippy

Skippy est un logiciel basé presque essentiellement sur la prédiction du mot suivant bien qu'il propose également une autocomplétion sur le mot en cours de rédaction. Il se présente sous forme de liste de suggestions une fois un mot entré. Il a la particularité d'avoir une option de paramétrisation qui va enregistrer les mots fréquemment utilisés dans le but de suggérer une liste de mots plus proches du vocabulaire qu'emploie l'utilisateur final, se rapprochant ainsi de notre objectif.

Il intègre également un système d'abréviations préenregistrées qui permettent à l'utilisateur de mettre au sommet de la prédiction des mots ou phrases clés qu'il a enregistrées.

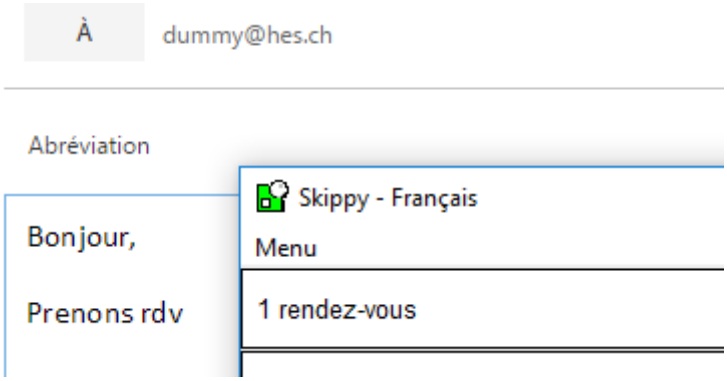

Figure 9 - Abréviation Skippy

Ce logiciel a toutefois le désavantage de coûter 210.- CHF dans sa version complète, ce qui le rend difficile d'accès pour des utilisateurs non-académiques. *(Cimis, s.d.)*

Il propose également dans sa suite un logiciel de lecture de document en audio qui est également utilisé pour apprendre de nouveaux mots au dictionnaire sur lequel Skippy se base.

3 Analyse de la structure de PhonoWriter

Avant de pouvoir nous atteler à la création d'une solution d'amélioration de PhonoWriter, nous devons en comprendre la structure. Il faut que nous nous intéressions en particulier à la base de données, car elle constitue notre principal outil de travail.

3.1 Base de données

3.1.1 Structure

L'avantage de PhonoWriter en ce qui concerne la gestion des données sur lesquelles l'application se base réside dans le fait que celles-ci sont toutes centralisées dans une seule base de données appelée « dictionnaries.db » .

Pour pouvoir étendre les fonctionnalités inhérentes à PhonoWriter et proposer une prédiction du mot suivant nous avons employé la structure de données suivante :

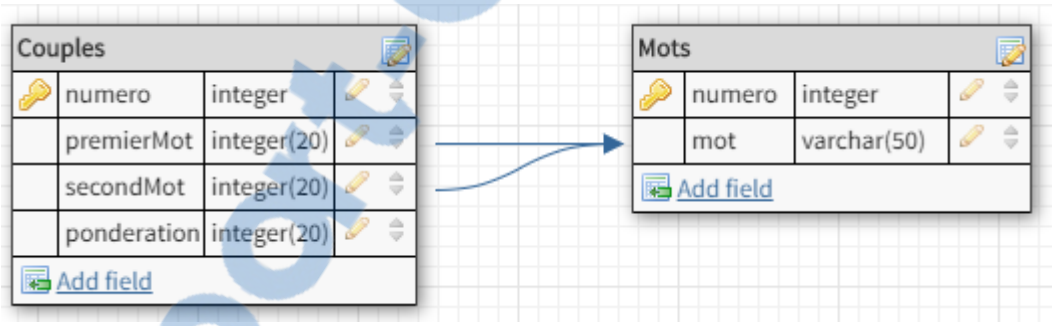

Figure 10 - Relations entre les tables pour la prédiction du mot suivant PhonoWriter

Il est également important de spécifier que dans le schéma ci-dessus, nous avons supprimé les champs relatifs aux définitions, aux images ou encore à la langue utilisée, car ceux-ci n'avaient pas d'incidence pour nous parce qu'ils n'ont aucune incidence sur la prédiction du mot suivant telle que nous souhaitons l'implémenter.

	numero	premierMot	secondMot	ponderation
	Filtre	Filtre	Filtre	Filtre
73	31	35144	13399	52
74	30	33794	32722	56

Figure 11 – Exemples de données issues de la table couples

3.1.2 Expansion

Grâce à ces informations nous pouvons élaborer nos processus de mise à jour de la base de données qui générera des données issues de nos extractions d'occurrences de mots, qui seront directement compatibles avec PhonoWriter. Il suffira en effet de jouer sur la pondération des mots (mesure de la fréquence) et d'insérer les mots inexistants dans la table des mots.

L'inconvénient de cette structure est qu'elle restreint la gestion du mot suivant à des couples de mots simples, et ne permet pas de traiter des suites de n mots (n-uples), sans des modifications fondamentales de la structure de données ainsi que du code source de l'application.

Nous avons donc géré ce problème dans notre solution de façon artificielle, en considérant au niveau de la structure de stockage des séquences comme « le fait de » comme des mots à part entière.

Cela nous permet de pouvoir offrir des prédictions pour « de », « fait de », « le fait de » en tant que trois entités distinctes, stockées de façon identiques dans la base de données.

4 Méthodologie de travail

Afin de respecter les prérequis du cadre imposé lors de notre travail de Bachelor mais également par affinité avec le processus, nous avons développé notre solution de manière agile et surtout itérative.

Nous retrouvant de façon hebdomadaire avec le Professeur Genoud et son assistant Jérôme Treboux, nous avons mis en place une optique de travail où nous présenterions lors de chaque séance une itération supplémentaire et fonctionnelle de développement qui serait soumise à leur validation.

Cela nous a permis d'organiser nos séances de développement en sprints courts ne dépassant pas une à deux semaines, et cela nous a également donné l'opportunité de mettre l'accent sur des solutions qui soient rapidement opérationnelles et discutables.

Dans une première phase, nous avons défini la ligne globale qu'allait prendre le développement de cette thèse. Nous avons tout d'abord défini les objectifs que nous souhaitions atteindre, et notamment les trois grands *epics* de notre solution, à savoir l'extraction des occurrences de n-uples dans une phrase, le web service ainsi que le processus de mise à jour de la base de données.

Puis dans une seconde phase, une fois ces trois solutions complètement implémentées, nous avons mis en place une méthodologie de test, afin de pouvoir extraire des résultats d'efficacité des différents processus.

Dans une dernière phase nous avons procédé à l'analyse ainsi qu'à l'explication des résultats obtenus, puis fourni des lignes directrices quant aux possibilités d'optimisation futures. Nous avons également exprimé en parallèle nos recommandations pour l'utilisation de notre solution avec PhonoWriter.

5 Extraction de l'occurrence de mots

Le but de cette première étape est de fournir les données qualitatives nécessaires au développement futur. Pour ce faire nous avons mis en place un processus extrayant d'une liste de fichiers donnés toutes les occurrences de n-uples existants (n de deux à cinq), et générant par la suite un fichier .CSV utilisable par les étapes suivantes de notre solution.

5.1 Choix des sources

5.1.1 Langage

Dans le processus de sélection des documents qui allaient servir de source aux algorithmes d'extraction mis en place par la suite, nous avons effectué le choix de nous baser uniquement sur du texte écrit en français.

Cela permet d'éliminer des incohérences pouvant survenir si de multiples langages étaient traités de façon simultanée. En effet, certains mots peuvent avoir des significations différentes en fonction de la langue et de ce fait entrainer des erreurs de classification. Par exemple, le mot *gift* signifie cadeau en anglais, mais poison en allemand ; ou encore la simple lettre *a* qui peut être interprétée comme le verbe avoir en français ou alors comme un déterminant en anglais. *(Harris, 2014)*

5.1.2 Réflexion sur les accents

Suite au choix de la langue française comme base pour nos algorithmes d'apprentissage et de mise à jour, une spécificité de la langue devait être gérée : les accents.

En effet, la langue française est une langue relativement riche en accents et caractères spéciaux (ç, œ, etc…), ceux-ci doivent donc être interprétés comme des lettres à parts entières, pour éviter la perte de données et de sens qu'entrainerait la standardisation de ces caractères.

Par exemple, supprimer l'accent grave sur la lettre « a », entraine une importante perte de sens, car la différenciation entre la forme conjuguée du

verbe avoir et la préposition indiquant une appartenance ou une destination ne peut plus se faire.

A l'opposé, enlever une cédille à « français », n'entraine pas de perte de signification.

Suite à l'évaluation de la perte importante potentielle en ce qui concerne la syntaxe et le sens des phrase, nous avons fait le choix de garder tous les accents dans les textes traités.

5.1.3 Réflexion sur les apostrophes

Afin de gagner en précision, nous nous sommes orientés vers une implémentation considérant les apostrophes comme partie intégrantes d'un mot. De ce fait, des mots tels que « c'est », « d'avoir » ou encore « l'informatique », sont considérés comme un seul mot, ce qui entraine une amélioration de la pertinence d'une suggestion, car ce sont là des mots qui apparaissent très fréquemment. Cela nous permet en conséquence de « gagner » un mot dans le contexte de notre prédiction (nous n'avons en effet pas besoin de prédire le couple « c » « est », car celui-ci ne représente qu'un mot).

Cette façon de procéder a toutefois l'inconvénient de nécessiter une plus grande base de données avec plus de sources, afin de pouvoir contextualiser les mots de façon plus adéquate.

5.1.4 Type de documents

En ce qui concerne les documents que nous avons sélectionné pour ce processus, nous avons choisi des documents ayant été soit rédigé par des étudiants en informatique de gestion à la HES SO Valais, ou alors des travaux récoltés sur internet, mais toujours dans le domaine global de l'IT.

Ce choix est motivé par un souci de cohérence des données. Le modèle qui serait généré à partir de données issues de sources très différentes comme de la poésie, un fascicule de chimie ou encore un roman policier par exemple, ne serait tout simplement pas viable. Ceci est dû au fait que des sources issues

de domaines différents ont une grande proportion de mots qui sont utilisés presque uniquement dans ceux-ci (un vocabulaire technique, des anglicismes…), et que le thème principal d'un ouvrage non-technique influencera grandement le vocabulaire employé.

Le parti pris de nous restreindre aux documents IT permet donc de gérer en détail les spécificités qui sont propres à ce domaine, car le vocabulaire dans lequel ils sont écrits reste semblable dans tous les textes.

5.1.5 Orthographe

Dans l'objectif de consistance des données nous avons opté pour des textes dont nous étions sûrs qu'ils ne contenaient pas ou peu de fautes d'orthographe. Ce choix est d'autant plus consolidé par le fait que nos sources sont des documents scientifiques et sont donc, en principe, relus à plusieurs reprises avant leur publication.

Toutefois, si des fautes devaient passer à travers les workflows d'extraction et de mise à jour de la base de données cela ne constituerait pas une erreur critique mais entrainerait une simple perte de données, au même titre que si le couple n'existait pas. Nous aurions alors en parallèle des n-uples avec des pondérations très faibles, souvent de un (les n-uples avec des fautes), et les n-uples corrects qui sont eux plus fréquents, et donc avec une pondération plus haute.

5.1.6 Encoding

La base de données sur laquelle se base notre solution étant encodée en UTF-8, nous avons en conséquence travaillé uniquement avec des textes respectant cette norme, dans un soucis évident de compatibilité, notamment au niveau des caractères spéciaux.

5.2 Occurrence de n-uples dans une phrase

5.2.1 Occurrence de couples dans une phrase

Pour pouvoir aiguiller l'utilisateur vers une suggestion de mot adaptée à son vocabulaire et à sa façon d'écrire, il est nécessaire d'obtenir des données

qualitatives et quantitatives sur les phrases et les combinaisons de mots qu'il utilise fréquemment.

Pour ce faire nous avons mis en place un processus KNIME utilisant un fichier texte fourni par l'utilisateur (une transformation est nécessaire depuis un .**docx** ou .**pdf**) pour réaliser cet apprentissage.

Le processus utilise la marche à suivre suivante :

1. Lecture du fichier et concaténation dans une seule cellule.
2. Identification de la ponctuation (fin de phrases) et marquage avec un caractère spécial.
3. Transformation du texte en minuscules pour éviter des duplicatas (par exemple les mots « *Ce* » et « *ce* » ne sont pas considérés comme identiques.)
4. Extraction de tous les couples de mots existants dans chaque phrase, et concaténation dans une seule cellule selon un format facilement divisible. (mot*1,mot2 ;*)
5. Extraction de chaque couple de mots et comptage du nombre d'occurrences de chacun.
6. Export sous format **.csv** et facilement utilisable par une base de données comme celle de PhonoWriter. (nbreOccurences.mot1, mot2, ;).

```
1,"aspects","de"
1,"aspects","juridiques"
1,"aspiré","par"
4,"assemblée","générale"
1,"assez","grande"
1,"assez","peu"
1,"assistant","aux"
1,"associations","dans"
```

Figure 12 - Exemple de fichier .CSV généré par l'extraction de couples dans une phrase

Dans une première itération, chaque fois qu'un nouveau fichier sera rajouté à la liste des fichiers de sources, le fichier d'export sera régénéré en concaténant les occurrences des différents fichiers dans un seul.

Par la suite chaque ajout de fichier mettra à jour la base de données sans devoir recalculer les occurrences de tous les fichiers, ceci pour un gain indéniable de temps et de puissance de calcul.

5.2.2 Occurrence de 5-uples de mots dans une phrase

Dans le but de complémenter la prédiction basée sur des couples de mots, nous avons mis en place un processus équivalent au précédent, à ceci près qu'il extrait des 5-uples et non pas des couples de mots.

Le fait d'utiliser cette méthode d'extraction pour des n-uples plus grands que deux requiert toutefois une étape de traitement en plus.

En effet, suite à l'adaptation du workflow précédent, nous obtenons cinq colonnes contenant chacune un mot, l'objectif étant de prédire la cinquième en nous basant sur les quatre premières.

Le problème qui survient est que les algorithmes de prédictions ne tiennent pas compte de l'ordre dans lequel les colonnes apparaissent pour établir leur prédiction.

Cependant, il est clair qu'en ce qui nous concerne l'ordre des colonnes, et par conséquent des mots, est crucial pour effectuer une prédiction qui soit la plus précise possible.

Pour pallier à cette éventualité, nous avons pris la décision d'effectuer une concaténation de quatre premières colonnes, afin que celles si soient traitées comme une seule colonne d'input dans le but de prédire la cinquième.

Nous avons cependant constaté que la viabilité de cette prédiction dépend grandement de la taille de la base de données, et nécessite donc une grande quantité de sources.

Il est en effet simple de concevoir que dans une collection de textes donnés, l'occurrence exacte d'une séquence de cinq mots est rare, et que la base de données sera donc peuplée par un grand nombre de 5-uples présents une seule fois.

Dans le cadre de notre travail, une base de données assez conséquente pour fournir des prédictions améliorées était trop difficile à atteindre, et ce tout

en respectant les contraintes imposées par la structure et le format de la base de données PhonoWriter.

Nous nous sommes donc orientés vers les occurrences de 3 et 4-uples dans l'objectif d'avoir de meilleurs résultats.

```
1,"maison tout en prenant","soin"
1,"maitre de gestion d'entreprise","maitrise"
2,"maitrise des outils microsoft","office"
2,"maitrise du français de","l'anglais"
1,"maitriser seule loin des","soucis"
1,"majeur sur les revenus","de"
1,"majeur vous êtes tenu","de"
1,"majeure il en va","de"
```

Figure 13 - Exemple de fichier .CSV généré par l'extraction de 5-uples dans une phrase

5.2.3 Occurrence des 3-uples de mots dans une phrase

Suite aux résultats peu probants obtenus avec l'extracteur de 5-uples dûs aux raisons évoquées dans le chapitre précédent, nous avons décidé de tenter d'extraire des 3-uples pour voir si les données s'avéraient meilleures.

Il va de soi que les règles utilisées pour les 5-uples doivent être également suivies, à savoir les deux premiers mots concaténés en une seule colonne.

Nous avons constaté que les occurrences étaient meilleures, dû au fait que les séquences de trois mots sont moins rares, et donc moins de 3-uples sont uniques, améliorant ainsi la prédiction.

```
1,"des rares","cartes"
1,"des relations","partenaires"
1,"des responsables","de"
4,"des ressources","humaines"
1,"des restaurants","des"
1,"des revenus","élevés"
1,"des routes","ainsi"
```

Figure 14 - Exemple de fichier .CSV généré par l'extraction de 3-uples dans une phrase

5.2.4 Occurrences de 4-uples de mots dans une phrase

Pour avoir à notre disposition le spectre complet, nous avons également intégré à notre processus d'extraction la gestion des 4-uples pour garantir que toutes les solutions aient été explorées.

```
1,"en  integrated","for"
1,"en  raison","des"
2,"en allemand qualités","requises"
1,"en asie amérique","sud"
1,"en aucun cas","éveiller"
1,"en ayant le","path"
1,"en binôme et","possédant"
1,"en bus l'agence","n'offre"
```

Figure 15 - Exemple de fichier .CSV généré par l'extraction des 4-uples dans une phrase

5.3 Implémentation

5.3.1 Emplacement

L'implémentation de notre procédé d'extraction d'occurrence est disponible sur le support physique fourni en annexe de ce document, dans le répertoire « *Worfklows et Applications /1.Workflows/1.Sequence Occurrence Extractor* »

Ce répertoire contient notamment un fichier *readme.txt* détaillant les prérequis pour pouvoir exécuter le processus.

Il contient également deux répertoires contenant des exemples de données d'entrée, et des exemples de données de sortie.

Parallèlement, le sous-dossier *Code Source* contient le fichier .**knfw** nécessaire à l'import dans KNIME du workflow.

5.3.2 Préconfiguration

Pour exécuter le processus, il suffit d'importer le processus dans une instance de KNIME (les détails du versioning se trouvent dans le fichier readme.txt), puis de changer les chemins d'accès aux répertoires d'entrée et de sortie. Il s'agit du nœud « *List Files – All txt files in folder* » pour l'input, et de tous les nœuds CSV Writer dans les méta nœuds « *x-uples Treatment : Occurrence count and all x-uples csw writer* ».

5.3.3 Structure

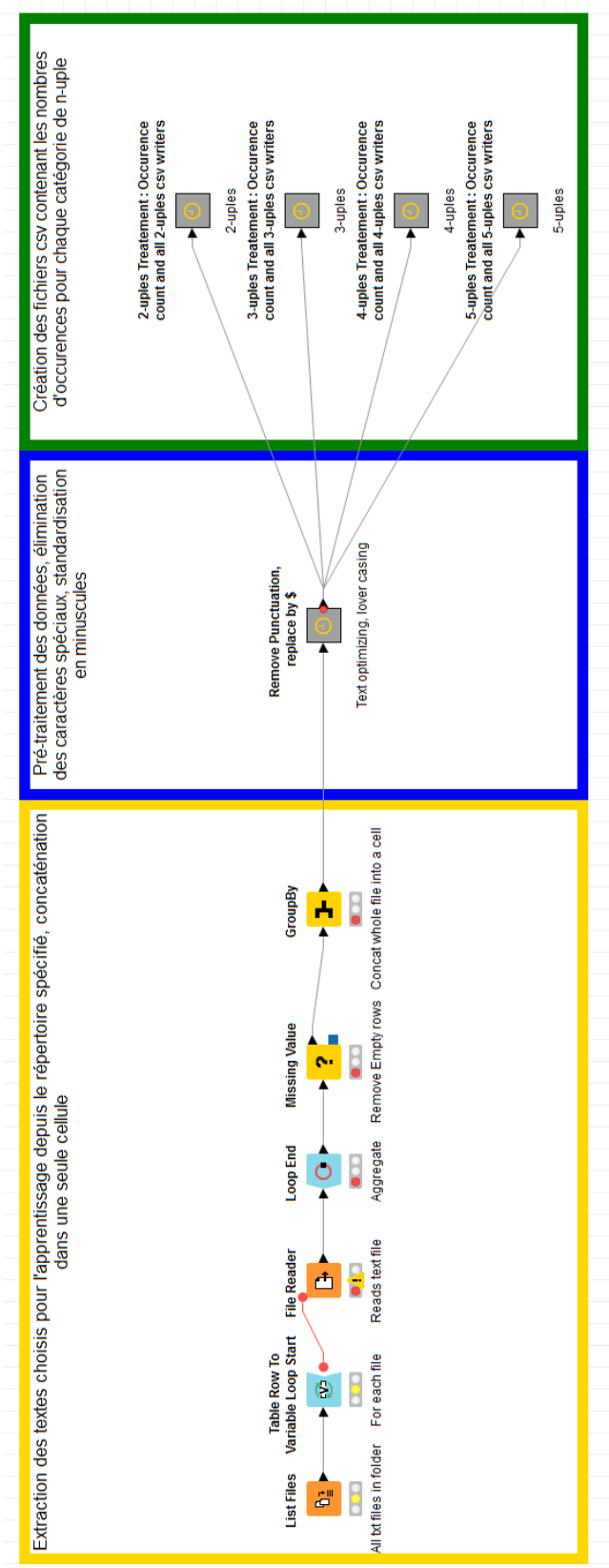

Figure 16 - Workflow : Sequence Occurence Extractor

Le processus s'articule autour de trois opérations principales :

- La partie jaune gère l'extraction et la concaténation des fichiers sources.
- La partie bleue transforme les données en éliminant notamment les caractères spéciaux et les majuscules,
- La partie verte extrait les couples, les filtre, les compte, puis les écrits dans des fichiers csv.

5.3.4 Particularités – Eléments intéressants

5.3.4.1 Fonction d'extraction des n-uples

Cette fonction prend en paramètre une chaine de caractères dont la ponctuation a été remplacée par le symbole $, puis extrait les n-uples en considérant les espaces comme séparateurs de mots

```
//Création d'un tableau contenant tous les mots
String[] temp  = c_ConcatenateCol0.split(" ");
String result = "";
// Si l'élément i+1,2,3 ou 4 est un $, cela signifie une fin de phrase, le n-uple n'est donc pas valable
// Dans le cas contraire, on extrait le n-uple que l'on ajoute au résultat, en le séparant des autres avec un ";"
for(int i=0;i <temp.length; i++){
     try{
          if(!temp[i+1].equals("$") && !temp[i+2].equals("$") && !temp[i+3].equals("$") && !temp[i+4].equals("$"))
               result += temp[i]+","+temp[i+1]+","+temp[i+2]+","+temp[i+3]+","+temp[i+4]+";";
     }
     catch(Exception e){
          }
}
```

Figure 17 - Méthode d'extraction de n-uples (5-uples dans cet exemple)

Voici un exemple de chaine produite par cette méthode :

« indépendamment,de,savoir,si,vous;de,savoir,si,vous,êtes ;savoir,si,vous, êtes,utilisateur; »

5.3.4.2 Nœud de suppression des lignes inutilisables

Ce nœud est responsable de purifier la chaîne produite par la fonction ci-avant en éliminant tous les n-uples qui ne sont pas pertinents. Il filtre par exemple les n-uples contenant des nombres, des adresse mail ou web, ou encore des caractères spéciaux qui pourraient encore être présents.

6 Web Service

6.1 Objectif

Suite à la complexité que nous avons rencontrée et l'impossibilité de rajouter directement nos processus à l'application PhonoWriter, nous avons pris la décision de fournir le résultat de notre travail sous forme d'un web service.

Ce service est conçu de manière modulaire pour pouvoir être employé facilement par PhonoWriter ou tout autre logiciel utilisant une structure de données équivalente. Ceci permet notamment à notre solution de faire abstraction de la couche de présentation des données (entrées utilisateur, affichage des suggestions de mots, …), et de nous concentrer sur le processus métier c'est-à-dire la prédiction du mot suivant

6.2 Implémentation

6.2.1 Emplacements

6.2.1.1 Déploiement

Le web service est déployé sur une machine virtuelle fournie par le service informatique de la HEVS (SInf), qui répond à l'adresse *« http://vmhphonowriter.hevs.ch/WordPredictorService/WordPredictorService.svc* ». Il est déployé sur un serveur Microsoft Internet Information Services (IIS 8.0) et est accessible sans aucun type d'authentification, ni dispositions particulières.

6.2.1.2 Code source

Le code source de ce service est disponible sur le support physique fourni en annexe de ce document, dans le répertoire « *Worfklows et Applications /2.Applications/ 1.WPS(Word Predictor Service)*»

Ce répertoire contient notamment un fichier *readme.txt* décrivant les étapes à effectuer pour pouvoir exécuter le service, les informations d'identifiant pour accéder directement à l'implémentation dans la machine virtuelle, ainsi que les emplacements du code source sur la machine virtuelle.

Nous signalons également que ce service nécessite un serveur IIS afin d'y être déployé, qui doit être préconfiguré pour autoriser les services WCF.

6.2.2 Préconfiguration

6.2.2.1 Environnement de développement et langage

Pour développer le web service, nous avons fait le choix d'utiliser l'environnement de développement Microsoft Visual Studio, dans sa version 2015, en utilisant le langage C#.

Ce choix a été fait non-seulement pour des raisons d'interopérabilité des différents composants (WCF, Serveur Web, langage de développement), car tous ceux si sont propriétaires de Microsoft et sont donc conçus pour fonctionner ensemble, mais également par affinité car nous maîtrisons le logiciel et apprécions le développement en C#.

6.2.2.2 Serveur IIS

Pour pouvoir exécuter notre web service dans des conditions optimales, il est indispensable de disposer d'un serveur IIS (qu'il soit distant ou local), capable d'accueillir les services WCF.

6.2.2.3 Packages NuGet

Afin d'installer les fonctionnalités indispensables au bon fonctionnement de notre web service, nous avons utilisé le gestionnaire de package NuGet, qui est présent par défaut dans la plateforme de développement Microsoft .NET.

6.2.2.4 Entity Framework

Comme notre solution nécessite des interactions régulières avec la base de données PhonoWriter « *dictionnaires.db* », nous avions besoin d'un Framework nous facilitant l'opération en évitant l'écriture « à la main » de toutes les requêtes *SQL*, ce qui représente un gain de temps colossal en évitant de nombreuses erreurs et des centaines de lignes de code souvent floues et peu modulables.

L'autre avantage majeur d'Entity Framework est que c'est également l'ORM (*object-relational mapping*) utilisé par PhonoWriter, ce qui constitue une plus-

value non-négligeable pour assurer la compatibilité de notre service avec l'application de la Fédération Suisse pour les Téléthèses.

6.2.2.5 System.Data.SqLite

Le choix de SQLite comme base de données nous est obligatoire car la structure de stockage PhonoWriter se base sur un fichier SQLite. Visual Studio et Entity Framework n'offrant pas un support natif de SQLite comme source de données, il a été nécessaire d'installer les packages NuGet correspondants.

Nous avons utilisé les versions 1.0.105.1 de SQLite, SQLite.Core, SQLite.EF6 et SQLite.LINQ (langage de requête indépendant de la base de données).

6.2.2.6 LINQ

Dans l'objectif de rendre notre web service le plus polyvalent et solide possible, nous avons pris le parti d'utiliser le modèle de programmation LINQ.

Cette technologie nous offre la capacité de remplir le vide existant entre l'approche relationnelle (base de données) et l'approche orientée objet (notre code C#). Elle nous permet de faire cela à travers un langage et une syntaxe indépendants des types de sources de données utilisées. En résumé, LINQ nous permet d'extraire des objets d'une base de données, en utilisant une syntaxe similaire à celle du C#. *(M, 2013)*

6.2.2.7 Prétraitement de la base de données

Avant de pouvoir extraire des objets C# avec Entity Framework, nous avons dû effectuer certaines opérations directement sur la base de données PhonoWriter. En effet, des incompatibilités entre la structure de la base de données et la version utilisée d'Entity Framework empêchant une utilisation optimale sont apparues.

Nous avons dû dans un premier temps dénormaliser la pondération des mots dans leur table, afin d'éviter des erreurs de casting avec des valeurs pouvant être décimales. Pour faire ceci, nous avons simplement modifié d'un

facteur 100 la magnitude de la pondération des mots. De plus, il a fallu que nous supprimions quelques lignes qui ne possédaient pas de définition, et donc généraient des erreurs de dépendances par rapport aux clés étrangères de la table « définitions ».

6.3 Méthodes

Notre webservice n'expose qu'une seule et unique méthode qui constitue la pierre angulaire de la capacité de prédiction.

6.3.1 getCouplesForWord(String mot, int nombreDeCouples)

Cette méthode fournira au client une liste de couples de taille équivalente au paramètre nombreDeCouples, contenant des Objets Couples dont le premier mot est équivalent au paramètre mot.

Elle est entièrement compatible avec le logiciel PhonoWriter, car elle se base sur un clone de sa base de données métier et donc les Objets EF utilisés sont identiques.

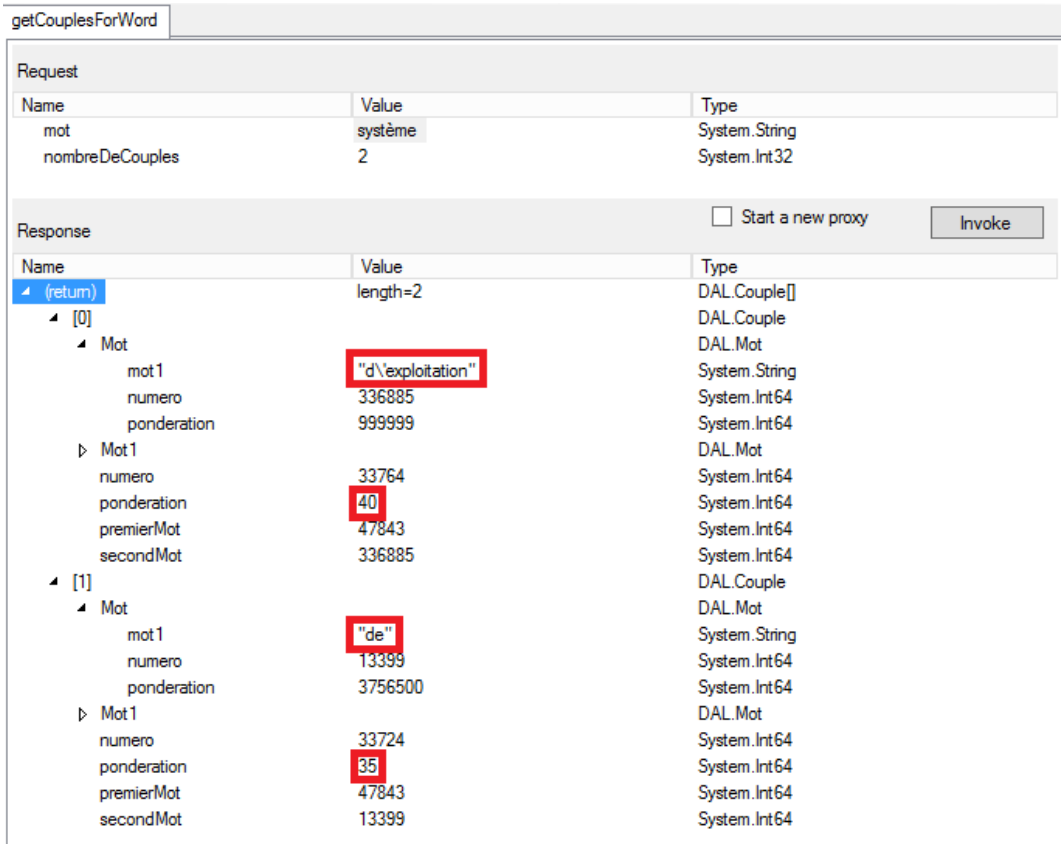

Figure 18 - Exemple d'appel du Word Predictor Service pour les paramètres (système,2). Les encadrés rouges correspondent aux mots prédits, ainsi qu'aux pondérations respectives des couples

La méthode utilise en pseudo code la logique suivante :

1. Initialisation de la connexion avec la DB à travers EF.
2. Récupération du mot correspondant à la requête dans le dictionnaire français
3. Récupération des couples contenant ce mot en tant que premier mot, triés par pondération, puis par pondération individuelle du second mot.
4. Ajout à la liste de retour du nombre de couples correspondants à la requête.
5. Dans le cas où le nombre de couples retournés ne serait pas suffisant, addition de couples vides pour atteindre le compte.

7 WPS Tester

7.1 Objectif

Afin de pouvoir tester de manière plus visuelle notre web service, nous avons souhaité mettre en place une application simple mais efficace, ayant l'avantage de montrer les différences prédictionnelles s'il nous décidions de considérer le dernier, ou les deux, trois, ou quatre derniers mots d'une phrase. Nous précisons également que cette application a uniquement pour but de proposer un « terminal humain » pour pouvoir tester le service durant la phase de développement. Il ne constitue pas la solution utilisée pour tester l'efficacité du service elle-même.

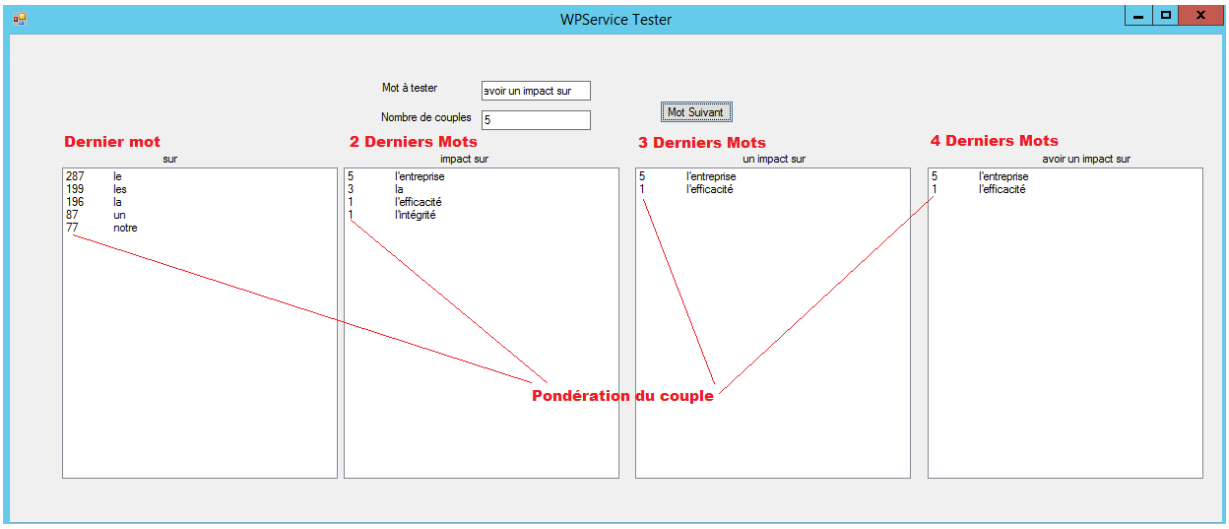

Figure 19 - WPS Tester - Ecran principal, exemple de prédiction

7.2 Implémentation

7.2.1 Emplacement

Le code source de cette application est disponible sur le support physique fourni en annexe de ce document, dans le répertoire « *Worfklows et Applications /2.Applications/ 2.WPService Tester*».

Ce répertoire contient également l'exécutable « WPS_TESTER.exe » qui permet de lancer l'application en standalone.

Il est également doté d'un fichier readme.txt expliquant les fondamentaux de l'utilisation de l'application.

7.2.2 Préconfiguration

Aucune préconfiguration n'est nécessaire, il suffit de disposer d'une connexion internet et de lancer l'exécutable fourni.

La première requête peut prendre quelques secondes car le programme doit initialiser la connexion avec le web service, les suivantes sont instantanées.

Nous précisons également que ce tester est lié directement au service déployé à l'adresse « *http://vmhphonowriter.hevs.ch/WPS/WPService.svc* » et que ceci est immuable.

Dans le cas où le lecteur souhaiterait parcourir le code source de l'application, il est nécessaire de le faire avec Microsoft Visual Studio, dans sa version 2015.

8 Mise à jour de la base de données (ETL)

8.1 Objectif

Ce workflow constitue le cœur de notre solution bien que son objectif soit très simple : mettre à jour la base de données PhonoWriter en fonction des fichiers .CSV générés par l'extraction de l'occurrence de mots, tout en jouant sur les modifications possibles de pondération.

8.2 Implémentation

8.2.1 Emplacement

Le code source de ce workflow est disponible sur le support physique fourni en annexe de ce document, dans le répertoire « *Worfklows et Applications /1.Workflows/ 2.ETL WordPredictorProject*».

Ce répertoire comprend également un fichier « readme.txt », contenant des recommandations d'utilisation et de paramétrisation.

Le support physique contient également des exemples de données d'entrée, ainsi que de sortie utilisées/générées par le workflow.

8.2.2 Préconfiguration

8.2.2.1 Microsoft Visual Studio

L'environnement de développement utilisé pour ce workflow est intégré à Visual Studio dans sa version 2015, qui est donc un élément indispensable de développement de la solution.

8.2.2.2 SQL Server Integration Services

Ce composant appartenant à la suite Microsoft SQL Server Database, disponible uniquement dans les versions Standard, Business Intelligence et Entreprise, représente l'outil clé de la tâche de mise à jour.

Il constitue une plateforme d'intégration de données et de processus métiers, couplé à un outil de data-warehousing que nous avons utilisé pour l'extraction, la transformation et le chargement (ETL) de nos données.
Nous avons utilisé la version 14.0.600.250

8.2.2.3 SQLite ODBC Driver

SSIS étant une structure logicielle prévue pour fonctionner comme une couche de plus haut niveau basée sur SQL Server, et n'offrant pour ainsi dire aucune fonctionnalité de prise en charge d'autres SGDB, nous avons dû installer un driver pour pouvoir utiliser des sources de données issues de fichiers de données SQLite.

Pour ce faire, nous avons employé le SQLite ODBC Driver dans sa version 0.9995.

8.2.2.4 Fichiers d'occurrences

Ce workflow s'exécute en se basant sur les fichiers d'occurrences générés par le processus d'extraction d'occurrences de mots, qui sont d'abord concaténés par l'utilisateur, dans un souci de simplification.

8.2.3 Structure

En termes de structure notre workflow SSIS reste très simple. Il nécessite toutefois quelques ajustements au niveau des nœuds ou tâches utilisés, car certains nœuds très puissants de SSIS ne sont malheureusement pas disponibles pour des sources SQLite, même avec le driver de compatibilité installé.

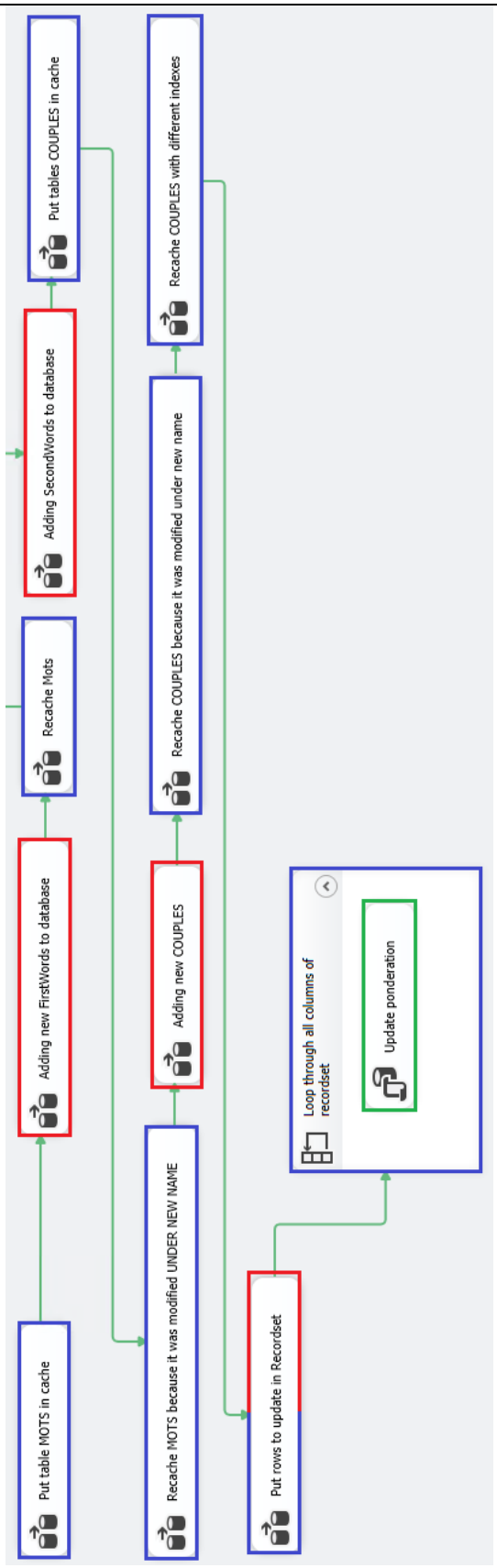

Figure 20 - Workflow ETL Word Predictor Project

Dans la figure ci-dessus, les tâches bleues représentent des opérations supplémentaires que nous avons dû effectuer pour assurer la compatibilité SQLite. Ce sont principalement des opérations de mise en cache de données, car SSIS ne peut pas effectuer d'opérations de transformations de recherche directement dans une base de données SQLite.

Les tâches rouges constituent des opérations d'ajout dans la base de données, et enfin la tâche verte est une tâche de modification de la base de données, notamment au niveau de la pondération des différents couples.

8.2.3.1 Ajout des nouveaux mots dans la base de données

Le processus se basant sur la liste de couples issue de l'algorithme d'extraction que nous avons précédemment décrit, doit maintenant vérifier l'existence de tous les mots dans la base de données. Si ceux-ci n'existent pas, il effectue alors une opération d'insertion.

Les valeurs qui ne sont pas directement pertinentes pour notre solution de prédiction sont fixées a des valeurs constantes. Comme la DB requiert des valeurs non-nulles pour celles-ci, nous ne pouvons pas simplement les ignorer.

Cette opération est relativement rapide, car plus le processus est utilisé, moins nous observerons de mots qui sont encore inconnus (Lorsque nous parlons de mots, nous incluons des n-uples).

8.2.3.2 Ajout des nouveaux couples dans la base de données

Un couple étant constitué de deux mots, et tous les mots inexistants ayant été désormais insérés dans la base de données par l'opération précédente, il nous est alors possible d'obtenir les clés étrangères des mots constituant les couples pour pouvoir insérer ces derniers à leur tour.

L'attribut de pondération de ces couples est fixé à zéro, et sera mis à jour plus tard dans le processus.

8.2.3.3 Mise en cache des couples à modifier

Nous devons maintenant déterminer quels sont les couples dont la pondération doit être mise à jour. Pour ce faire, nous récupérons les valeurs de pondération de tous les couples de notre collection de données issue du fichier .CSV.

Nous allons maintenant pouvoir préparer tous ces couples pour la mise à jour directement dans la base de données en fixant leur pondération égale à une fonction de l'ancienne pondération (pour rappel, tous les couples qui viennent d'insérer ont cette valeur égale à zéro, pour les autres elle est supérieure à zéro), et de la nouvelle pondération (tirée du fichier d'occurrences).

C'est précisément à cette étape que les différents paramètres pondérationnels (dont nous parlerons dans la partie processus de test du service) sont mis en place.

Nom de la colonne dérivée	Colonne dérivée	Expression	Type de données
finalPonderation	<ajouter comme nouvelle...	(Occurence * 5) + ponderation	four-byte signed integer [DT_I4]

Figure 21 - Exemple de détermination de la nouvelle pondération selon le paramètre pondérationnel X5 : La nouvelle pondération est égale à l'ancienne + 5 fois la nouvelle

8.2.3.4 Mise à jour de la base de données

Maintenant que toutes les pondérations à modifier ont été mises en cache, la dernière étape est d'itérer sur chaque ligne de la table Couples dans la base de données, et de mettre à jour celle-ci si elle possède un des couples mis en cache.

Cette opération est particulièrement gourmande en ressources car elle n'est en effet pas du tout conçue pour être optimale. Il aurait été beaucoup plus simple d'utiliser une simple commande OLEDB, mais celles-ci n'étaient pas disponibles, car notre source de données n'était pas compatible.

Nous déconseillons donc fortement l'utilisation de ce processus pour plus de 50'000 couples à la fois, car les performances en sont exponentiellement réduites. A l'heure actuelle, sur la machine virtuelle ou nous l'avons exécuté, il

nous a fallu compter environ 30 minutes par tranche de 10000 couples à mettre à jour.

8.2.4 Particularités – Eléments intéressants

8.2.4.1 Scalabilité

Comme nous l'avons défini plus haut, ce processus gère de manière simultanée et équivalente tous les n-uples possible. La seule limitation est la source de données générée dans le fichier d'occurrences, qui doit fournir des données pour le n-uple voulu.

S'il était nécessaire de gérer des suites de mots plus grandes que cinq, aucune modification du processus ne serait nécessaire.

8.2.4.2 Migration

Si d'aventure le système de base de données sur lequel s'appuie PhonoWriter devait changer pour être migré vers un SGBD différent, il serait facile et rapide de modifier le processus pour prendre en compte cette nouvelle structure. Il serait en effet uniquement nécessaire de modifier les sources de données.

Dans le cas où le système de migration serait fixé à un Microsoft SQL Server, il serait possible de supprimer toutes les opérations de mise en cache ainsi que les solutions de contournement et de les remplacer par des commandes optimisées, qui sont partie intégrante de SSIS.

Cela engendrerait un gain conséquent d'efficacité, ainsi que de flexibilité et de facilité de programmation. *(Esposito, 1999)*

8.2.4.3 Sources de données

Afin d'obtenir un résultat qui soit au plus près de nos objectifs, nous allons utiliser pour la phase d'apprentissage et de mise à jour des textes écrits par la même personne et sur le même sujet que le texte utilisé pour tester l'efficacité du service. Nous utiliserons donc dans notre cas des textes écrits par l'auteur. De manière générale, les sources de données répondant aux mêmes autres critères de sélection que celles du processus d'extraction d'occurrences.

9 Processus de test qualitatif du service

9.1 Objectif

Maintenant que notre service est configuré et disponible, il nous faut être en mesure de tester à quel point il est performant. En d'autres termes, nous devons déterminer quel est le taux de prédictions correctes pour chaque paramètre de prédiction possible, pour en extraire le plus efficace.

9.2 Implémentation

9.2.1 Emplacement

L'implémentation de notre workflow de test est disponible sur le support physique fourni en annexe de ce document, dans le répertoire « *Worfklows et Applications /1.Workflows/3.WPS_Accuracy_Tester*»

Ce répertoire contient notamment un fichier *readme.txt* détaillant les prérequis pour pouvoir exécuter le processus.

Il contient également deux répertoires contenant des exemples de données d'entrée, et des exemples de données de sortie.

Parallèlement, le sous-dossier Code Source contient le fichier .**knfw** nécessaire à l'import dans KNIME du workflow.

9.2.2 Préconfiguration

Pour exécuter le processus, il suffit d'importer le processus dans une instance de KNIME (les détails du versioning se trouvent dans le fichier « readme.txt », puis de changer les chemins d'accès aux répertoires d'entrée et de sortie, en fonction de l'environnement de test.

Il est important de signaler que pour que ce processus fonctionne, le WPS déployé sur « http://vmhphonowriter.hevs.ch/WPS/WPService.svc » doit être atteignable, les nœuds de web service étant configurés pour s'y adresser.

9.2.3 Paramètres prédictionnels

9.2.3.1 Nombre de prédictions

9.2.3.1.1 Maximisation

Lorsqu'une application contacte notre web service afin d'obtenir une prédiction, elle est en mesure de spécifier la quantité de prédictions qu'elle souhaite obtenir. De par ce fait, il est possible d'obtenir une efficacité maximum lors que **le nombre de prédictions demandées est égal au nombre d'entrées dans la base de données pour un n-uple donné**.

En effet, il est facile de concevoir que si le n-uple correct est déjà connu de la base de données, et que celle-ci retourne tous les n-uples connus pour une expression donnée, celle-ci sera forcément dans la liste des prédictions, et donc, considérée comme correcte.

Inversément, si le service ne fournit qu'un seul n-uple à chaque requête, les prédictions correctes ne seront que celles où le couple le plus fréquent (possédant la pondération la plus haute) était le bon.

9.2.3.1.2 En réalité

Malheureusement, il n'est pas admissible pour un logiciel proposant des suggestions de mots d'en proposer un trop grand nombre. Il n'est en effet pas humainement possible pour un utilisateur écrivant du texte d'analyser 30 n-uples possibles proposés par le service afin de trouver celui qui l'intéresse, et ce dans un temps raisonnable, à chaque fois qu'il écrit un nouveau mot.

Nous avons donc dû déterminer pour tester l'efficacité du service, quels étaient les nombres de suggestions adéquates à fournir à l'utilisateur.

N'ayant pu trouver aucune étude ayant été effectuée directement à ce sujet, nous nous sommes rabattus sur une analyse similaire concernant le nombre d'auto-complétions proposées par différents moteurs de recherches, dont nous avons pu extraire les résultats suivants *(rpauldesign, 2014)* :

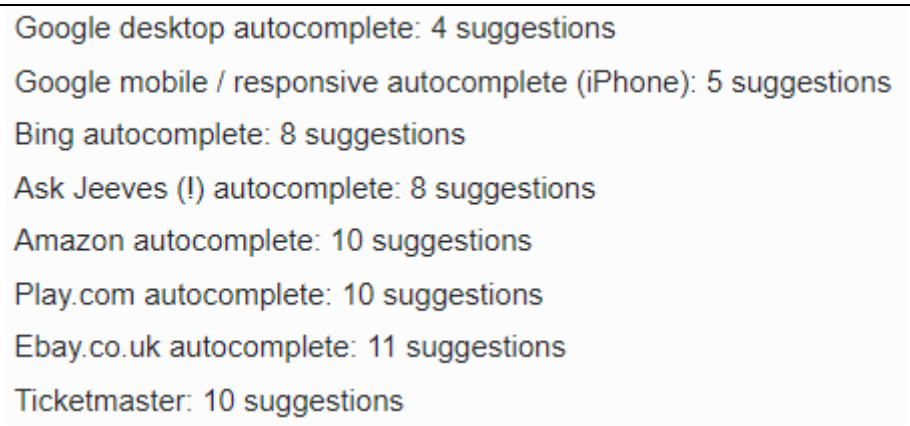

Figure 22 - Nombre de suggestions proposées par des moteurs de recherche (ux.stackexchange.com)

Le contexte de ces recherches étant légèrement différent, car celles-ci ne sont en général pas effectuées aussi rapidement que lors de la rédaction d'un texte par exemple, nous avons choisi de limiter nos suggestions dans la fourchette inférieure, à savoir entre **deux** et **cinq** propositions.

9.2.3.2 Nombre de derniers mots considérés

Comme nous l'avons décrit dans le chapitre occurrence des mots, notre web service est capable, pour prédire de façon adéquate, de considérer jusqu'aux quatre derniers mots d'une phrase. Il appartient à l'application qui l'appelle de définir la taille de l'expression qu'il considèrera pour la prédiction. Dans le cas de la phrase « je viens voir la », l'appel en utilisant « là », « voir la », « viens voir la », « je viens voir la », produira des résultats différents.

Dans notre processus de test, nous allons tester tous les découpages de phrase afin de pouvoir déterminer le plus efficace.

9.2.3.3 Paramètres pondérationnels

La dernière variable de ce processus ne représente pas directement un paramètre interne au workflow lui-même, il s'agit plutôt d'un état de la base de données sur lequel le web service est testé. Si nous nous référons à la section traitant du processus de mise à jour de la base de données, nous avons vu qu'il était possible de fixer la pondération d'un couple en fonction des valeurs à disposition.

La façon de déterminer cette pondération donne lieu à différentes instances de la base de données, qui répondent différemment aux requêtes de prédictions, les poids de leurs différents couples n'étant pas semblables.

Le web service ne pouvant toutefois se baser que sur un seul dictionnaire de manière simultanée pour garantir la compatibilité de la solution avec PhonoWriter, nous avons dû effectuer le processus de test pour chacune des pondérations que nous avons choisies, mais également pour l'état initial de la base de données, avant toute mise à jour.

Voici la liste des pondérations que nous avons utilisé :

- Pondération Normale : La pondération est fixée à l'ancienne + la nouvelle.
- Pondération X5 : La pondération est fixée à l'ancienne + 5* la nouvelle.
- Pondération X20 : La pondération est fixée à l'ancienne + 20* la nouvelle.
- Pondération X50 : La pondération est fixée à l'ancienne + 50* la nouvelle.
- Pondération 999999 : La pondération est fixée à l'ancienne + la nouvelle + 999999, cela permet d'avoir des valeurs excédant toutes celles déjà présentes dans la base, en gardant toutefois un sens de l'échelle entre les valeurs mise à jour. Nous pourrons en effet observer des valeurs à 1000000 et d'autres à 1000002 par exemple.

9.2.4 Structure

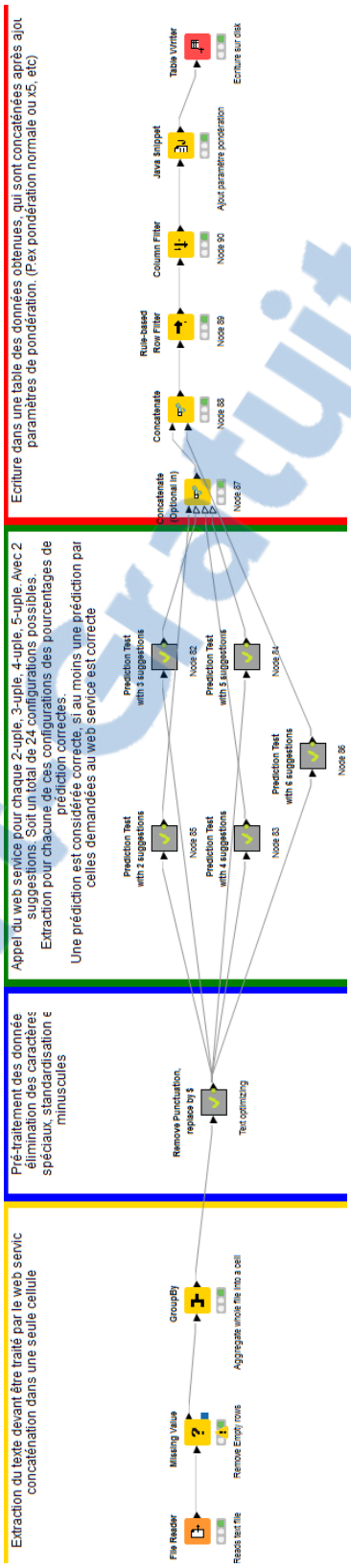

Figure 23 - WPS Accuracy Tester, structure

Dans une première phase, nous allons sélectionner le texte avec lequel nous souhaitons tester le web service. Celui-ci ne doit pas avoir été au préalable traité par le workflow de mise à jour de la base de données, ce qui engendrerait des résultats complètement faussés.

Ce texte sera par la suite formaté de façon équivalente au processus d'extraction des couples dans le but d'extraire des n-uples, desquels les déchets auront été supprimés. (partie jaune et bleue du schéma ci-dessus).

Pour chaque n-uple (2,3,4,5) extrait du texte, nous allons contacter le WPS et exiger deux à six prédictions, et vérifier si l'une d'entre elles correspond au dernier mot du n-uple. (partie verte).

Comme expliqué dans la partie des paramètres prédictionnels, nous aurons donc 20 configurations possibles pour chaque instance pondérationnelle de la base de données.

Enfin, nous allons concaténer les résultats de ces appels et le nom de l'état pondérationnel de la base de données dans une table (partie rouge).

Suite à l'exécution de ce processus pour chaque base de données, nous aurons à notre disposition six tables contenant les résultats pour pouvoir en extraire des conclusions.

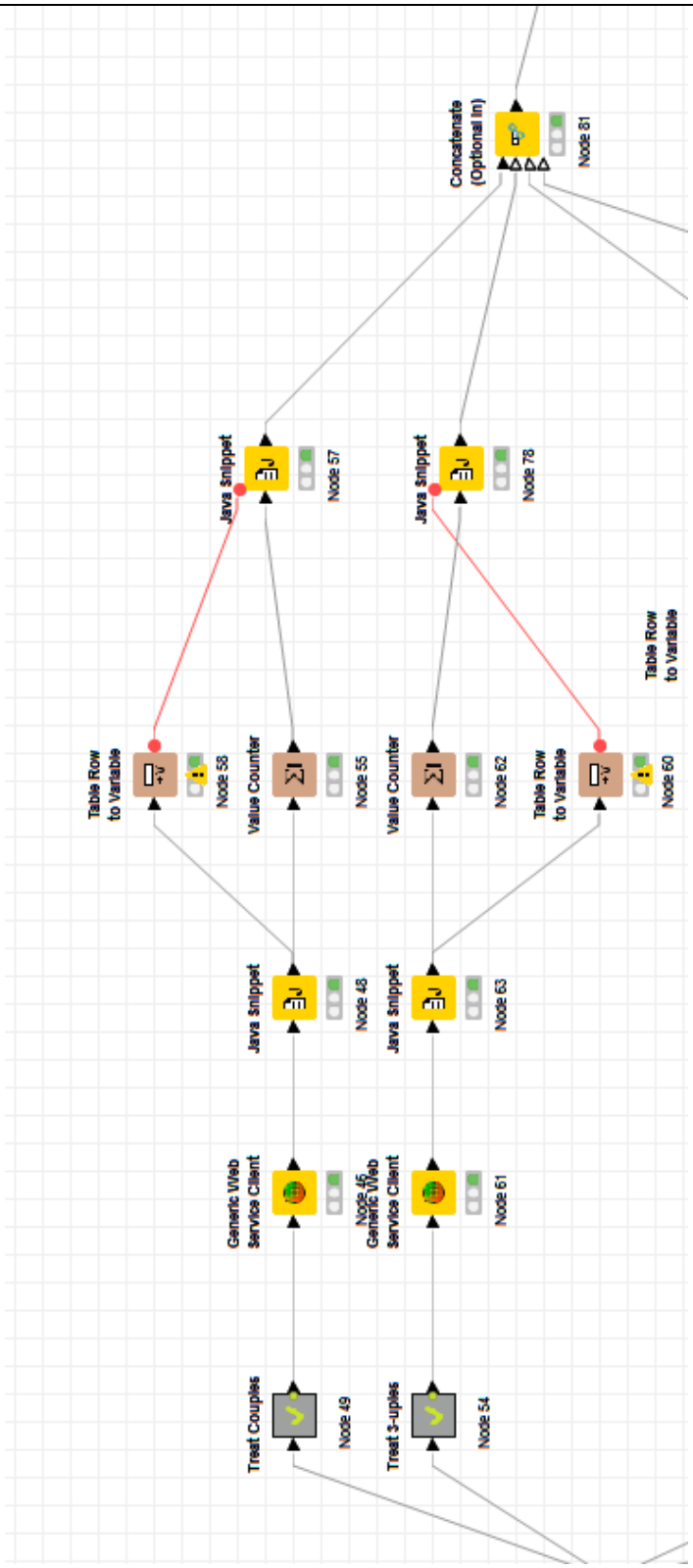

Figure 24 - Détail du meta-noeud "Prediction Test with 4 suggestions", nous pouvons observer le décompte des prédictions correctes pour les couples et les 3-uples

10 Analyse des résultats

Avant de commenter directement les résultats, nous allons détailler sous quelle forme ceux-ci seront présentés.

10.1 Workflow

10.1.1 Emplacement

L'implémentation du workflow de comparaison des résultats est disponible sur le support physique fourni en annexe de ce document, dans le répertoire « *Worfklows et Applications /1.Workflows/4.WPS Tester Comparator*»

Ce répertoire contient notamment un fichier *readme.txt* détaillant les prérequis pour pouvoir exécuter le processus.

Parallèlement, le sous-dossier *Code Source* contient le fichier .**knfw** nécessaire à l'import dans KNIME du workflow.

10.1.2 Préconfiguration

Ce workflow ne nécessite aucune configuration particulière, si ce n'est une instance de KNIME (les détails du versioning se trouvent dans le fichier readme.txt), ainsi que le changement des chemins d'accès aux tables d'entrée. (celles-ci sont fournies également dans le répertoire Exemples d'input).

10.1.3 Structure

Ce workflow met en exergue le résultat obtenu après le scénario suivant :

Mise à jour de la base de données avec 10 textes, puis test d'efficacité avec un 11ème texte, puis test d'efficacité de ce même texte après la mise à jour de la base de données avec 5 nouveaux textes, écrits par le même auteur que le 11ème.

Cette dernière étape sera effectuée pour chaque valeur de pondération.

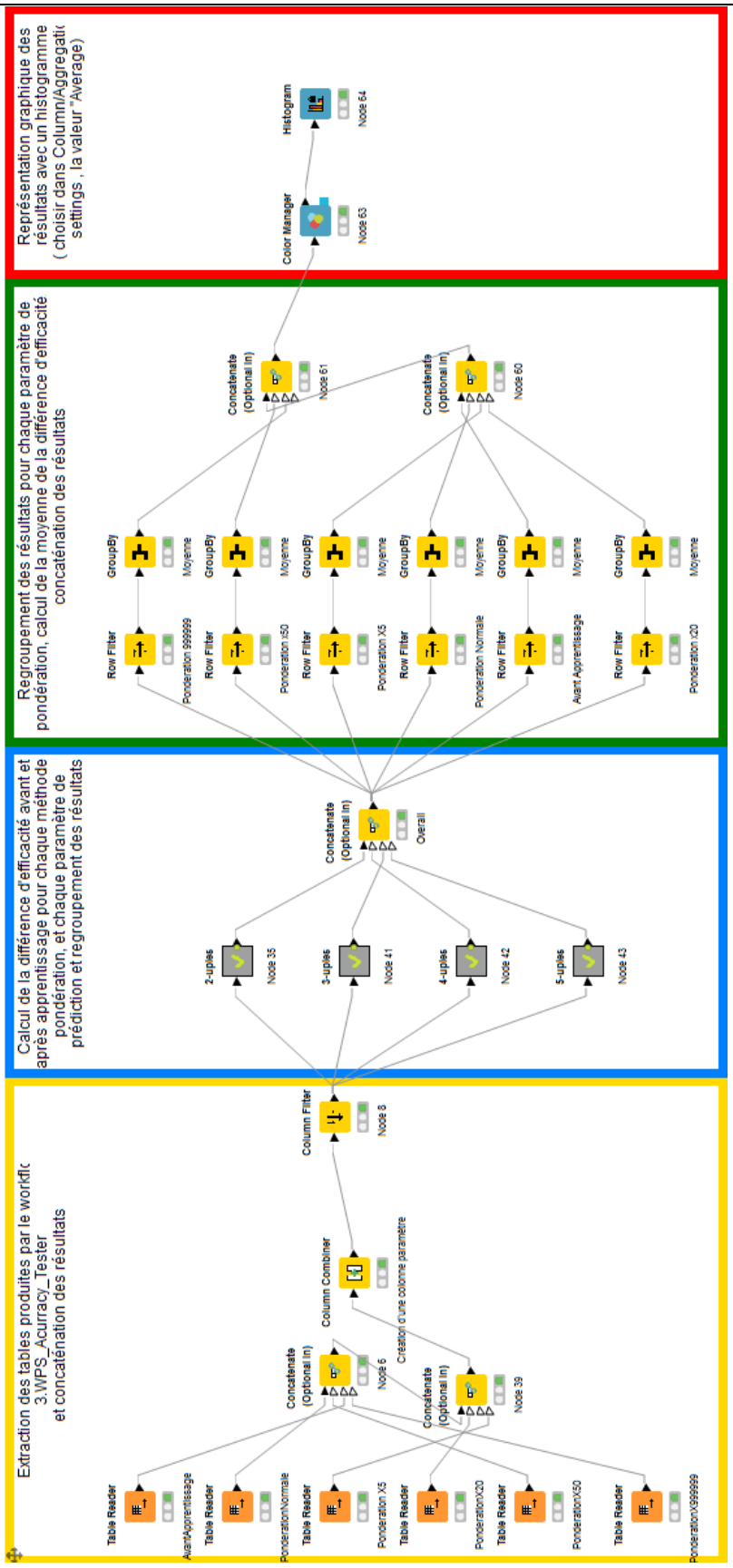

Figure 25 - Structure du workflow de comparaison de résultats

Cette étape générera les six tables de la partie jaune du workflow ci-dessus.
La partie bleue sera elle responsable de calculer la différence d'efficacité avant et après apprentissage, puis la partie générera la moyenne pour chaque paramètre de pondération.
La partie rouge sera quant à elle un résumé sous forme visuelle des résultats obtenus.

10.2 Estimation des résultats

Avant de découvrir et de commenter les résultats obtenus, nous allons effectuer une estimation, basée sur les tendances que nous avons observées et notre intuition, de ce que les résultats devraient être.

10.2.1 Meilleur paramètre pondérationnel

Premièrement, aucun paramètre de pondération ne devrait entraîner une baisse d'efficacité par rapport aux valeurs avant mise à jour / apprentissage.

La pondération **999999** devrait produire les meilleurs résultats, car grâce à elle, les couples utilisés par l'auteur des textes de mise à jour ainsi que celui de test vont se voir attribuer des valeurs de pondération supérieures au reste de la base de données.

Le texte de test étant en théorie rempli de ces n-uples car ils appartiennent au même vocabulaire, la logique voudrait qu'ils soient prédits de façon plus régulière, car leur pondération sera de toute façon supérieure à n'importe quel autre couple.

10.2.2 Meilleur nombre de prédictions

Dans ce domaine précis, nous ne pourrons qu'observer le fait que le nombre maximum de prédictions utilisées donnera le meilleur résultat. Dans notre cas, il s'agit de **6** prédictions. Comme nous l'avons détaillé dans la partie concernant les paramètres prédictionnels du chapitre sur le processus de test du web service, il n'est en effet pas mathématiquement possible qu'un nombre

inférieur de prédictions donne un résultat meilleur. Ce dernier doit être au maximum identique au résultat d'un nombre de prédiction supérieur.

10.2.3 Meilleur nombre de derniers mots considérés

Selon le bon sens, plus le nombre de derniers mots considérés est grand, plus le contexte de la phrase sera précis, et en conséquence la prédiction sera plus juste. Toutefois la taille et la qualité de la base de données doivent être prises en considération : bien qu'une séquence de cinq mots offre un contexte plus grand, il est également moins probable que la base de données connaisse déjà cette séquence en particulier, et soit donc en mesure d'effectuer une prédiction correcte.

Par rapport à notre expérience et nos observations au cours du développement de cette solution, nous estimons que le pic d'efficacité en termes de nombre de derniers mots considérés devrait se situer entre **3** et **4** mots**.**

Ceci car nous estimons la taille de la base de données suffisante pour connaître l'essentiel des 3-uples/4-uples rencontrés dans le texte.

10.3 Résultats obtenus

10.3.1 Meilleur paramètre pondérationnel

Pour pouvoir déterminer le meilleur paramètre pondérationnel, nous avons utilisé comme indicateur la moyenne du gain d'efficacité pour chacun des 20 paramètres prédictionnels correspondants au paramètre pondérationnel évalué. Nous avons extrait les observations suivantes :

Paramètre Pondérationnel	Moyenne du gain d'efficacité
Avant Apprentissage	Valeur de référence
Pondération Normale	**+ 0.491 %**
Pondération X5	**+ 0.412 %**
Pondération X20	**+ 0.162 %**
Pondération X50	**+ 0.118 %**
Pondération 999999	**+ 0.302 %**

Figure 26 - Meilleur paramètre pondérationnel

Conformément à notre première assomption, nous constatons en effet qu'aucun paramètre pondérationnel ne constitue de manière globale une baisse d'efficacité de la prédiction, bien que cela ne soit pas vrai pour certains paramètres prédictionnels que nous décrirons ci-après.

En second lieu, nous observons que notre prédiction de la pondération 999999 comme étant la plus efficace est erronée. Ce sont en effet les pondérations bouleversant le moins l'équilibre de la base de données (normale et X5) qui engendrent les meilleurs résultats, bien que l'extrême opposé avec une pondération extrêmement élevée (999999) voit une augmentation sensible par rapport aux valeurs médianes X20 et X50.

Ces résultats peuvent être expliqués par le fait qu'une augmentation normale de la pondération conserve le ratio de pondération entre les différents types de mots : de manière générale, les conjonctions et autres déterminants qui possèdent intrinsèquement des valeurs élevées car ils apparaissent souvent gardent toujours une pondération plus importante que des simples mots, même si ceux-ci font parties d'un vocabulaire typique où ils apparaissent régulièrement.

Inversément, attribuer une pondération anormalement élevée X20-X50-999999 détruit cet équilibre et des mots rares peuvent dépasser en pondération des séquences extrêmement courantes qui ne se seraient toutefois pas trouvées dans le texte d'apprentissage.

Nous spécifions que ces observations sont étroitement liées à la taille de la base de données et surtout au nombre de textes utilisés pour la remplir.

Pour illustrer ce cas de figure, prenons par exemples les couples « chien aboie », « chien d'aveugle », « chien d'exercice » ayant des pondérations respectives de 100, 5 et 2.

Le couple « chien aboie » est clairement celui qui apparait le plus dans notre base de données et a en conséquence la pondération la plus haute.

Imaginons maintenant que nous mettons la base de données à jour avec un texte ou apparaissent les termes « chien d'exercice » trois fois, et « chien d'aveugle » une fois.

Si l'on applique une pondération élevée X50 à cet apprentissage, nous retrouverions les valeurs suivantes :

- Chien d'exercice 152
- Chien aboie 100
- Chien d'aveugle 55

Chaque fois que le service se verra demander un couple pour le mot chien, celui-ci suggérera « d'exercice » bien que ce couple-ci soit beaucoup plus rare que « aboie » entrainant ainsi de nombreuses prédictions erronées.

Inversément, si nous appliquons une faible pondération X2 à cet apprentissage, nous retrouverions les valeurs suivantes.

- Chien aboie 100
- Chien d'exercice 8
- Chien d'aveugle 7

Avec ce paramètre pondérationel, nous aurions été en mesure d'effectuer une distinction entre le vocabulaire spécifique « d'exercice » et « d'aveugle », sans pour autant interférer sur le ratio du couple bien plus fréquent. En conclusion, nous pouvons affirmer que le facteur de pondération utilisé doit être déterminé en fonction du ratio entre les expressions courantes et spécifiques.

10.3.2 Meilleur nombre de prédictions

Comme nous l'avons prédit dans notre phase d'estimation, il n'y a aucune surprise quant au résultat obtenu. C'est en effet la requête de six prédictions qui obtient les meilleurs rendements.

Nombre de prédictions	% de prédictions correctes
2	**15.687**
3	**17.572**
4	**19.235**
5	**21.535**
6	**23.088**

Figure 27 - Efficacité prédictionnelle pour les prédictions de 2-uples, avec une pondération normale

La différence d'efficacité entre deux et six prédictions observée est de **7.4%** et demeure constante au pourcent près pour toutes les pondérations testées des prédictions de 2-uples.

Bien que l'écart soit moins important pour les autres n-uples, une augmentation du nombre de prédictions entraine toujours une augmentation du taux de prédiction correctes.

10.3.3 Meilleur nombre de derniers mots considérés

Dans l'objectif de déterminer ce nombre, nous nous sommes basés sur les meilleurs résultats obtenus pour les paramètres pondérationnels ainsi que le nombre de prédictions, à savoir la pondération normale et six prédictions respectivement.

Nous obtenons les données suivantes :

Nbre de derniers mots considérés	% de prédiction correctes
2	**23.088**
3	**11.351**
4	**3.43**
5	**0.894**

Figure 28 - Efficacité prédictionnelle par nombre de derniers mots considérés pour 6 prédictions avec pondération normale

C'est ici que survient la plus grande surprise. Non seulement, les prédictions 3-4-5 sont au moins à moitié moins bonnes que la prédiction par couples, mais leurs valeurs sont extrêmement basses. Nous avions prédit un pic d'efficacité compris entre des prédictions pour trois et quatre derniers mots considérés, mais nous observons en fait ce pic pour deux derniers mots considérés.

Cela s'explique par le fait que le contexte ne joue pratiquement aucun rôle d'améliorateur de la qualité de prédiction, car une très grosse majorité des n-uples pour lesquels une prédiction est demandée sont tout simplement inconnus de la base de données, et ne pouvaient donc en toute logique pas être prédits.

Lors de nos essais nous avons constaté que la plus grosse proportion de fausses prédictions ne venait non pas d'une mauvaise prédiction au détriment d'un autre mot, mais que le n-uple n'existait simplement pas dans la base de données.

Le pourcentage de n-uples inconnus lors de la phase d'apprentissage représente environ 55% pour les 2-uples, 82 % pour les 3-uples, 95 % pour les 4-uples et 99% pour les 5-uples.

Tant que la base de données ne sera pas suffisamment enrichie, ces valeurs continueront d'être hautes, mais baisserons au fur et à mesure, de façon logarithmique par rapport aux nombres de mots considérés avec pour conséquence directe de déplacer le pic d'efficacité prédictionnelle vers un nombre plus élevé de derniers mots.

Toutefois, il est probable qu'a un certain niveau, rajouter des couples ne produise que du bruit supplémentaire, baissant ainsi la qualité des données.

Pour illustrer ce phénomène, nous avons réalisé la simulation suivante :

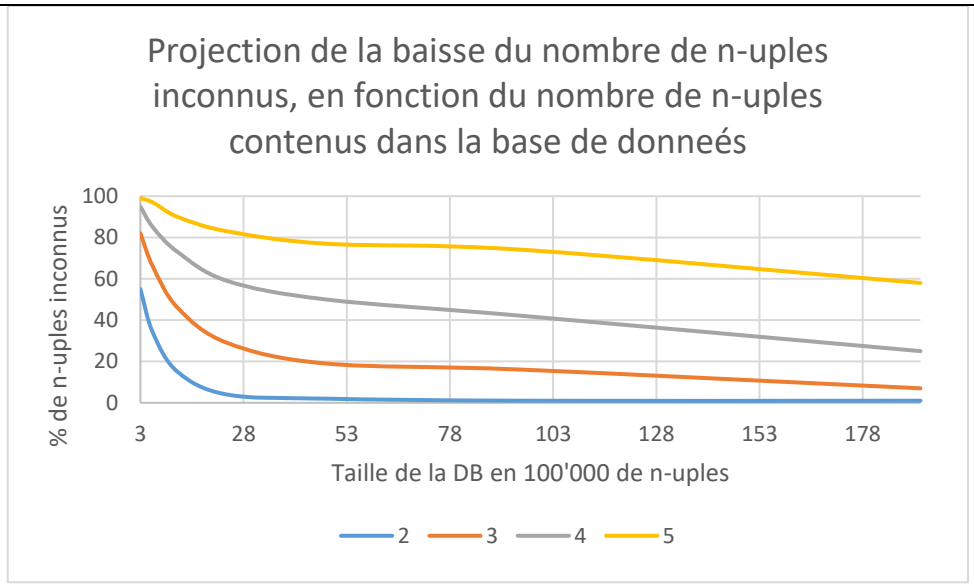

Figure 29 - Projection de la baisse du nombre de n-uples inconnus, en fonction du nombre de n-uples contenus dans la base de données

Dans cette simulation, seules les valeurs initiales ont été calculées par l'auteur, les suivantes ont été fixées de manière à démonter une tendance générale. Nous pouvons observer que le gain d'efficacité dépend directement du nombre de n-uples inconnus et de la taille de la base de données. Les n-uples plus grands nécessitent plus de données pour fournir de l'efficacité.

10.3.4 Meilleurs paramètres globaux

Pour synthétiser cette phase d'analyse des résultats, nous sommes en mesure d'affirmer que le trio, **6 prédictions** pour une prédiction se basant sur une **pondération normale** en considérant uniquement le **dernier mot** est celui qui procure les résultats les plus précis.

En l'absence d'une base de données plus conséquente, nous recommandons donc ces trois paramètres pour l'utilisation de notre solution de prédictions de mots.

11 Conclusion

Dans cet ultime chapitre, nous disséquerons de façon factuelle les fruits de notre travail. Nous exprimerons ensuite nos recommandations envers la FST, si celle-ci souhaitait intégrer notre développement à sa solution. Nous terminerons par donner quelques ouvertures pour optimiser ou améliorer ce que nous avons mis en place.

11.1 Résultat final et bilan

A la conclusion de ce travail, nous pouvons présenter un bilan positif. Nous avons été en mesure, à travers notre implémentation, de proposer une solution concrète et applicable pour pouvoir rajouter la fonctionnalité de prédiction de phrases par apprentissage au logiciel PhonoWriter.

Nous avons notamment pu prouver, au moyen d'une analyse détaillée des résultats, qu'il était possible d'obtenir un taux de prédictions correctes d'environ un quart, et qu'en plus de ça, ce taux d'efficacité pouvait être augmenté au travers de notre processus de mise à jour de la base de données.

Notre étude nous a également permis de mettre en exergue les limites de la prédiction de mot suivant de notre solution ainsi que celles de la structure de données mise en place pour PhonoWriter. Nous avons relevé qu'en effet, la taille de la base de données actuelle (environ 300'000 n-uples), n'était pas adéquate pour pouvoir offrir des prédictions prenant en compte plus qu'un seul mot. De surcroît, la structure actuelle de la base de données et surtout son format, rendent des opérations de mises à jour de grande ampleur longues et coûteuses en ressources.

Il n'en demeure pas moins vrai que dans l'état actuel des choses, l'infrastructure mise en place correspond à une augmentation opérationnelle des capabilités du logiciel PhonoWriter.

11.2 Recommandations

Nous recommandons aux personnes qui intégreront notre solution dans une éventuelle version de PhonoWriter d'utiliser dans une première phase une prédiction basée sur six suggestions, en ne considérant que le dernier mot entré, puis de passer sur les deux derniers mots une fois que les 3-uples donneront des résultats prédictionnels supérieurs à 25% et au minimum égaux à ceux des 2-uples.

Nous suggérons également de n'utiliser que la pondération de mise à jour normale, l'équilibre des pondérations étant trop complexe à préserver avec d'autres méthodes.

Bien que le format de la base de données ait été choisi pour être facilement portable, nous recommandons la migration de la base de données vers une solution plus robuste basée sur du SQL Server, du moins lors de la phase d'enrichissement de la base de données.

Nous conseillons en conséquence de ne pas déployer directement la version actuelle de notre solution dans une version opérationnelle de PhonoWriter, mais d'enrichir à l'aide des entrées utilisateurs la base de données, pour construire une somme de connaissances qui soit assez étendue pour connaître une grande majorité des n-uples instruits.

Enfin, nous recommandons de ne pas considérer plus de derniers mots que 3, car nous estimons la phase d'apprentissage trop importante et longue pour pouvoir fournir un jour des résultats opérationnels.

11.3 Future works (Evolution et améliorations envisageables)

Pour finaliser ce document, nous donnerons quelques lignes directrices sur les éléments qui pourraient être mis en place pour augmenter l'efficacité de notre solution. Nous tenons à signaler que si nous n'avons pas directement implémenté ces aspects, c'est parce qu'ils auraient nécessité dans une certaine mesure la restructuration de la base de données PhonoWriter, ou parce que l'ampleur de travail qu'ils représentaient était telle, qu'ils auraient pu être des sujets de travaux à part entière.

11.3.1 Stemming

Le stemming, ou racinisation en français est un procédé de transformation des flexions en leur radical ou racine. *(Wikipedia, 2017)*

L'idée de ce processus est de réduire les mots rencontrés à la base sémantique qui les constituent. Des mots comme « chercher,chercheur,cherchent,cherchées,cherchera » seront donc tous réduit à leur radical « cherch », qui représente l'idée sémantique de la recherche.

Appliqué à notre solution, cela signifierait qu'au lieu d'associer des couples de mots, nous pourrions associer des couples de racines. Par exemple l'idée de chercher serait associée à celle de solution, ou de trésor par exemple. De plus, aucune différenciation ne serait faite en fonction des déclinaisons ou de la conjugaison. En d'autres termes les n-uples « chercher un trésor , chercherait un trésor, cherchera la trésorerie » seraient tous considérés comme des n-uples identiques.

Il n'est toutefois pas possible d'insérer ces radicaux directement dans la base de données, car ceci entrainerait des incohérences au niveau de la correction orthographique ; « cherch » n'étant en effet pas un mot correct.

Pour donner une idée au lecteur, nous avons implémenté un exemple d'extraction situé sur le support physique en annexe de ce document, dans le dossier « *Workflow et Applications / 1.Workflows / 5.WordRankingStemmingTextProcessing »,* celui-ci extrait, pour une liste donnée de documents les quatres racines les plus fréquentes de chaque document, puis les représente sous forme de nuage de mots, ou la taille correspond à la fréquence d'apparition.

Figure 30 - Exemple de nuage de mots crée par un workflow de stemming réalisé par l'auteur

11.3.2 Part of speech

Ce processus correspond à extraire la catégorie syntaxique d'un mot, correspondant à sa fonction dans la phrase. Les catégories les plus communes sont par exemples les noms les pronoms, les déterminants ou encore les verbes.

Extraire cette information est bien plus complexe qu'il n'y paraît, car cela nécessite de traiter le texte d'entrée de façon intelligente, en extrayant beaucoup plus d'informations du contexte. Il ne s'agit plus là de repérer simplement des mots qui se suivent.

D'un autre côté, ajoutée à notre processus prédictionnel, cette information pourrait améliorer la qualité des suggestions. Imaginez par exemple que la phrase que vous êtes en train d'écrire ne comporte pas de verbe, à chaque nouveau mot entré, la pondération d'une prédiction contenant un verbe serait augmentée, améliorant ainsi la probabilité qu'un verbe soit suggéré.

Au même titre que le stemming, le stockage de cette information entrainerait une refactorisation de la base de données actuelle.

11.3.3 Correction orthographique et prédiction de mot

Pour améliorer la qualité d'une prédiction, il serait possible de conjuguer la capabilité de prédiction orthographique déjà produite par PhonoWriter avec la prédiction de mot suivant que nous avons implémenté. Cela consisterait à utiliser les premières lettres tapées d'un second mot, puis d'extraire tous les couples contenants le premier mot, et les mots contenant les lettres entrées.

Ceci ne serait en revanche plus directement de la prédiction de mot suivant, mais constituerait plutôt un complément d'efficacité à l'autocomplétion.

On pourrait imaginer ajouter aux prédictions FUZZY, PHONETIC, CLASSIC, une prédiction COUPLE, qui implémenterait ce principe.

11.3.4 Restructuration phonétique

Un des problèmes rencontrés par notre solution, est la présence de nombreux n-uples qui peuvent être réduits à une ou deux syllabes.

Par exemple, dans le langage courant, des expressions comme « je suis la » sont simplement prononcées rapidement comme « chuila » et ne sont pas directement interprétées par nos interlocuteurs comme une séquence de trois mots, mais comme un mot à part entière.

Il serait envisageable dans notre solution d'extraire ces séquences puis de les stocker au niveau phonétique. Le problème toutefois serait identique à celui rencontré pour le stemming et générerait des incohérences au niveau de la correction orthographique.

12 Références

Apfssu. (2013, Mars). Consulté le Juin 7, 2017, sur afpssu.com: http://www.afpssu.com/wp-content/uploads/2013/07/logiciels_dyslexie.pdf

Cimis. (s.d.). *Logiciel de prédiction d'anticipations de mots Skippy*. Récupéré sur Cimis: http://www.cimis.fr/logiciel-de-prediction-d-anticipation-de-mots-skippy.html

Dr Berthold, M., & Dr Thiel, K. (2012). *The KNIME Text Processing Feature : An Introduction.* Récupéré sur tech.knime.org: https://tech.knime.org/files/knime_text_processing_introduction_technical_report_120515.pdf

Esposito, D. (1999, Octobre 31). *OLE DB or ODBC?* Récupéré sur SQL ServerPRO: http://sqlmag.com/sql-server/ole-db-or-odbc

FST. (2016). *PhonoWriter Windows*. Récupéré sur FSTLAB: http://www.fstlab.ch/site/index.php/produits/phonowriter

HACAVIE. (2011, Janvier 24). *Logiciel de prédiction de mots PHRASE EXPRESS*. Consulté le Mai 13, 2017, sur Hacavie: http://www.hacavie.com/aides-techniques/articles/logiciel-de-prediction-de-mots-phrase-express/

Harris, K. (2014, 7 24). *10 English Words that mean something else in other languages*. Récupéré sur Ink Tank: http://inktank.fi/10-english-words-mean-something-else-languages/

Kadav, A. (2015, Août 17). *How does TF-IDF Work*. Consulté le Mai 30, 2017, sur Quora: https://www.quora.com/How-does-TF-IDF-work

Kooij, J. S. (2013). *Adult ADHD: Diagnostic Assessment and Treatment.* Lopndon: Springer.

M, S. (2013, 3 21). *Using LINQ in .NET*. Récupéré sur C SharpCorner: http://www.c-sharpcorner.com/UploadFile/c5c6e2/using-linq-in-net/

Mahood, A. (2011, Novembre 21). *SQL Server Data Tools Team Blog*. Récupéré sur blogs.msdn.microsoft.com: https://blogs.msdn.microsoft.com/ssdt/2011/11/21/what-is-sql-server-data-tools-ssdt/

rpauldesign. (2014, Avril 22). *How many search suggestions should i display*. Récupéré sur ux.stackexchange: https://ux.stackexchange.com/questions/56126/how-many-search-suggestion-should-i-display

Rpubs. (s.d.). *PredictNextWord*. Consulté le Mai 13, 2017, sur Rpbus: http://www.rpubs.com/scenthr/PredictNextWord

Tyranus, S. (2014, Mars 4). *What are the major differences and benefits of Porter and Lancaster Stemming alorithms?* Consulté le Mai 30, 2017, sur Stack Overflow: https://stackoverflow.com/questions/10554052/what-are-the-major-differences-and-benefits-of-porter-and-lancaster-stemming-alg

Washington Post. (2012, Juillet 31). *How much time you really spend emailing at work*. Récupéré sur The Washington Post: https://www.washingtonpost.com/blogs/post-leadership/post/how-much-time-you-really-spend-emailing-at-work/2012/07/31/gJQAI50sMX_blog.html?utm_term=.b6ad7d4d031b

Wikipedia. (2017, Mars 19). *Racinisation*. Récupéré sur Wikipedia: https://fr.wikipedia.org/wiki/Racinisation

Wikipedia. (2017, Mars 24). *SQL Server Integration Services*. Récupéré sur Wikipedia: https://en.wikipedia.org/wiki/SQL_Server_Integration_Services

Annexe I Contenu du support physique

Le présent support physique fourni en annexe de ce dossier contient les éléments suivants :

- Le présent document en format PDF.
- Un dossier contenant l'intégralité du développement effectué, ainsi que les fichiers d'installation des logiciels testés lors de l'état de l'art.
- Des fichiers readme.txt contenant les instructions et les recommandations d'exécution pour chaque workflow et application développé.
- Le poster de présentation de cette thèse en format .pptx.

Annexe II Journal de travail

Date	Catégorie	Tache	Heures
30.01.2017	Dev - Knime	Word raking Stemming avec les nœuds KNIME Labs	8
30.01.2017	Redaction - Analyse	Analyse DB et code source phonoWriter et rédaction	2
30.01.2017	Web Service	Tentative de déploiement vers vm distante	4
30.01.2017	Dev-Knime	Enrichissement des sources tb notamment	3
30.01.2017	Web Service	Setup Remote vm	4
30.01.2017	Dev - Knime	Adaptation de l'extracteur de couples pour gérer les 3-uples	4
01.05.2017	Administratif	Séance Hebdomadaire	1
03.05.2017	Administratif	Création de la structure du répertoire dropbox	1
03.05.2017	Anaylse Préliminaire	Prise en main de PhonoWriter	2
03.05.2017	Anaylse Préliminaire - Rédaction	Fonctionnement - PhonoWriter	2
07.05.2017	Analyse Préliminaire	Fonctionnement des logs PhonoWriter	2
08.05.2017	Administratif	Séance Hebdomadaire	2
10.05.2017	Dev - Knime	Optimisation du compteur d'occurrence de sequences pour plusieurs fichiers	5
13.05.2017	Logistique	Mise en place de l'environnement de dev (visual studio, github)	4
13.05.2017	Analyse Préliminaire - Rédaction	Fonctionnement - Phrase Express	3
14.05.2017	Analyse Préliminaire - Rédaction	Fonctionnement Skippy	3
14.05.2017	Analyse Préliminaire	Compréhension Code source PhonoWriter (PredictionRelationship.cs)	3
14.05.2017	Logistique	Mise en place environnement de dev (Knime, sql server)	4
14.05.2017	Dev - Knime	Creation du compteur d'occurrence de mots dans un texte	7
16.05.2017	Administratif	Séance Hebdomadaire	2
18.05.2017	Dev - Knime	Creation du compteur d'occurrence de sequences dans un texte	5
19.05.2017	Redaction	Compteur d'occurrence knime	2
22.05.2017	Aide	Séance Knime Jérôme	2
22.05.2017	Dev - Knime	Mise en place des premiers models	7
22.05.2017	Rédaction	Choix des sources (language, documents)	3
22.05.2017	Dev - Knime	Adaptation de l'extracteur de couples pour gérer des 5-uples	4
23.05.2017	Administratif	Séance Hebdomadaire	1
30.05.2017	Rédaction	Choix des sources (accents)	1
30.05.2017	Dev - Knime	Adaptation de l'extracteur 5-uples pour forcer les 4	2

		premiers mots en 1 colonne	
30.05.2017	Rédaction	Occurrence des 5-uples de mots dans une phrase	2
31.05.2017	VM	Création environnement de dev dans la VM	6
01.06.2017	VM	Premier Web Service de base disponible remotely	5
01.06.2017	Administratif	Séance Hebdomadaire	1
05.06.2017	Web Service	Initialisation d'entity framework - faire marcher sqlite comme datasource pour EF	6
06.06.2017	Web Service	Faire marcher la méthode get all words avec EF	9
06.06.2017	Rédaction - Web service	Préconfiguration	3
06.06.2017	Knime	Creation du workflow pour update la database a partir du fichier d'occurrences de couples	3
06.06.2017	VS integration	Update la table Mots avec les nouveaux mots issus du fichier d'occurrences (SSIS) (2uples)	9
08.06.2017	Dev - SSIS	Update table mots avec fichier d'occurrence SSIS FINI	9
08.06.2017	Web Service	Refaire Entity Framework code first	6
12.06.2017	Web Service	Création de requête rendant les couples avec la plus haute pondération	4
12.06.2017	App Test	Création d'une application de test du web service	4
13.06.2017	Administratif	Séance Hebdomadaire	1
13.06.2017	Rédaction	LINQ	5
13.06.2017	Dev - SSIS	Adaptation workflow pour 3-uples	2
14.06.2017	Dev - Tester	Adaptation application de test pour 3 uples	6
16.06.2017	Rédaction	Avant Propos, Résumé, Remeciements, table illustrations	4
17.06.2017	Rédaction	Introduction, Contexte, Objectifs	4
19.06.2017	Dev - Knime SSIS	Adaption pour gérer "c'est" au lieu de "c est"	7
20.06.2017	Administratif	Séance Hebdomadaire	1
21.06.2017	Dev-Knime	Creation Worlfklow test du web service avec un texte que j'ai écrit	6
27.06.2017	Administratif	Séance Hebdomadaire	1
27.06.2017	Dev Web Service	Adaptation web service 2-3-4-5 uples	5
27.06.2017	Dev Knime	Creation workflow test avec 2-3-4-5-6 suggestions pour 2-3-4-5 uples	8
28.06.2017	Knime SSIS	Exécution et test avec pondération normale, x5, x20	15
29.06.2017	Knime SSIS	Exécution et test avec x50, 99999	13
30.06.2017	Administratif	Séance Hebdomadaire	1
30.06.2017	Knime	Creation des résultats, moyenne par catégorie	6
03.07.2017	Rédaction	Processus de test du web service, paramètres prédictions	3.5
03.07.2017	Administratif	Séance Hebdomadaire	1
05.07.2017	Dev SSIS	Corrections de divers bugs lors de la mise à jour	8
10.07.2017	Rendu	Préparation des packages Logiciels à rendre sur CD (	8

		Workflows et applications)	
10.07.2017	Rendu	Embelissement des workflow et code source à travers la disposition et les commentaires	6
10.07.2017	Administratif	Séance Hebdomadaire	1
12.07.2017	Rédaction	Sequence occurrence extractor, ETL	15
14.07.2017	Rédaction	Workflows	14
15.07.2017	Rédaction	Analyse des résultats	15
17.07.2017	Rédaction	Finalisation du rendu	14
20.07.2017	Rédaction	Relecture et corrections	8
21.07.2017	Rédaction	Relecture et corrections	8
22.07.2017	Rédaction	Relecture et corrections	8
28.07.2017	Rédaction	Relecture et corrections	5
		TOTAL	**359.5**

Déclaration sur l'honneur

« Je déclare, par ce document, que j'ai effectué le travail de Bachelor ci-annexé seul, sans autre aide que celles dûment signalées dans les références, et que je n'ai utilisé que les sources expressément mentionnées. Je ne donnerai aucune copie de ce rapport à un tiers sans l'autorisation conjointe du Responsable de Filière et du professeur chargé du suivi du travail de Bachelor, y compris au partenaire de recherche appliquée avec lequel j'ai collaboré, à l'exception des personnes qui m'ont fourni les principales informations nécessaires à la rédaction de ce travail et que je cite ci-après : M. Dominique Genoud, M. Jérôme Treboux . »

Sierre, le 2 août 2017

Pierre Baran