Damien Levy

Rapport de stage

du 9 avril 2018 au 6 juin 2018

Au Laboratoire lorrain de recherche en informatique et ses applications

à Vandœuvre-lès-Nancy

Encadré par MM. Bruno Guillaume, Yves Lepage, Jean Lieber et Emmanuel Nauer
Tuteur de stage : M. Emmanuel Jeandel

Acquisition semi-automatique de cas de corrections de phrases en français

Licence Informatique
L3
De l'Université de Lorraine

Table des matières

Avant Propos

L'objectif de ce stage est d'acquérir des cas de corrections de phrases en français en exploitant le corpus WiCoPaCo. Ce corpus est issu d'historiques de modifications de pages sous Wikipédia. Certaines de ces modifications consistent en des corrections de phrases, ce qui permet de récupérer des cas. Ce stage sera composé principalement de deux étapes. La première consiste à récupérer le corpus WiCoPaCo sous forme XML et à en extraire des couples de phrases avant et après corrections. La deuxième étape consistera à distinguer les modifications qui correspondent à des corrections de phrases (par opposition aux modifications changeant la signification de la phrase). Cette étape n'est pas envisagée comme étant purement automatique, cependant, plusieurs critères seront envisagés afin de faire une présélection des corrections de phrase. Par exemple, un critère portant sur une distance d'édition entre les deux phrases sera envisagé : on fait l'hypothèse qu'une correction de phrases change relativement peu de caractères par rapport à une modification de fond. Cette hypothèse sera bien sûr évaluée. L'idée est ainsi d'obtenir des couples (phrases à corriger, phrase corrigée) permettant, après validation, d'enrichir la base de cas.[1]

Avant l'acquisition des cas de corrections de phrases en français, M. Lieber m'a demandé de créer une base de cas initiale reposant sur des recherches personnelles. La première étape énoncée dans la fiche de stage à été effectuée. La deuxième étape à été abordée, mais d'autres critères de sélection pourront être envisagés et ajoutés.

Le stage s'est déroulé correctement et fût très enrichissant. La base de cas exploitant le corpus WiCoPaCo à été créée.

1 Extrait de la fiche de stage

Remerciements

Je souhaite remercier Messieurs Bruno Guillaume, Yves Lepage, Jean Lieber et Emmanuel Nauer pour leur encadrement ainsi que pour leurs conseils lors de la durée du stage.

J’aimerais aussi remercier messieurs Nam Ly et André Giang pour leur aide, la bonne ambiance d’équipe durant ce projet commun et leurs retours lors de la préparation de la soutenance.

Introduction

Lorsque nous rédigeons un texte en français, nous souhaitons écrire de la manière la plus juste possible. Pour ce faire, il est possible d'utiliser des outils informatiques tels que les correcteurs orthographiques par exemple. On peut aussi avoir envie de proposer une phrase en français qui nous paraît incorrecte et vouloir sa correction. Pour pouvoir créer un programme de ce genre, on peut utiliser le raisonnement à partir de cas. Cela consiste à résoudre un problème en s'appuyant sur une base de cas contenant des couples problème/solution. Un modèle de ce processus classique comprend plusieurs étapes : d'abord une phase d'inférence puis d'une phase d'apprentissage. La phase d'inférence se compose d'une étape de remémoration et d'une étape d'adaptation. L'étape de remémoration consiste à sélectionner un cas source jugé similaire au problème cible. L'étape d'adaptation permet de modifier la solution du cas source en solution candidate résolvant le problème cible. Enfin, l'étape d'apprentissage présente à un expert le cas nouvellement formé, qui le corrige si besoin, et le valide. Dans l'hypothèse où ce nouveau cas est considéré comme utile, il est ajouté à la base de cas.

Dans le cadre de ce stage, l'objectif est de créer une base de cas pour une application à la correction de phrases. Le sujet de ce stage s'intègre dans un projet de création d'un programme correcteurs de langue. Deux autres personnes travaillent sur ce programme, messieurs André Giang et Nam Ly. M. Giang travaille sur un moteur d'inférences pour la correction de phrases en français. M. Ly travaille sur le développement de la base de données ainsi que sur une interface web servant d'interface entre utilisateur et le moteur d'inférences. Pour alimenter la base de cas, il sera utile de créer une base de cas minimale manuellement, afin que M. Giang puisse avancer dans la création du moteur d'inférences pour la correction de phrases en français. Il faudra aussi acquérir de manière semi-automatique des couples de phrases en mauvais français accompagnées de leurs solutions en français correct. Les cas seront extraits du corpus WiCoPaCo[2] (Wikipedia Correction and Paraphrase Corpus). Ce dernier est un historique des modifications de pages sous Wikipédia réalisé sous licence GNU Free Documentation License, contenant notamment des couples de phrases avant et après corrections. Et enfin, durant le processus de raisonnement à partir de cas, il faudra valider, puis ajouter les nouveaux cas créés par le programme, à la base de cas. La base de cas sera stockée dans une base de données créée par M. Ly.

Pour ce faire , je vais tout d'abord créer une base de cas minimale fondée sur mes expériences et recherches personnelles. Dans un premier temps, ces cas seront écrits dans un fichier texte, puis insérés dans une base de données. Ensuite, je vais récupérer le corpus WiCoPaCo sous forme XML, et en extraire des couples de phrases avant et après corrections. Pour finir, je vais isoler les modifications qui correspondent à des corrections de phrases. Je m'appuierai sur plusieurs critères afin de faire une présélection. Enfin, je validerai les couples obtenus dans le but d'enrichir la base de cas.

2 Aurélien Max and Guillaume Wisniewski, Mining Naturally-occurring Corrections and Paraphrases from Wikipedia's Revision History, LREC 2010.

Description de l’entreprise

Le Loria[3], Laboratoire lorrain de Recherche en Informatique et ses Applications à été créé en 1997. C’est une Unité Mixte de Recherche commune au CNRS, à l’Université de Lorraine et à Inria.

La mission du Loria est la recherche fondamentale et appliquée en sciences informatiques.

Le Loria est ordonné en 5 départements, composé de 28 équipes dont 15 communes avec Inria.

Le stage s’est déroulé dans le Département 4 : Traitement automatique des langues et des connaissances qui est sous la responsabilité de M.Bruno Guillaume. Ce département est divisé en 7 équipes : CELLO, MULTISPEECH, ORPAILLEUR, READ, SmarT, SEMAGRAMME et SYNALP. J’ai travaillé dans l’équipe Orpailleur. Le responsable de cette équipe est M.Amedeo Napoli. Les membres de cette équipe conçoivent des systèmes à base de connaissances pour raisonner et résoudre des problèmes dans des domaines d’application donnés.

3 #6.Références bibliographiques/Webographie|outline

Travail réalisé

Environnement de travail et choix du langage

N'ayant que très peu eu l'occasion d'extraire des données d'un fichier XML, j'ai fait quelques recherches sur l'Internet afin de choisir un langage de programmation adapté. Pour le choix du langage, je me suis appuyé sur plusieurs critères : la disponibilité de bibliothèques permettant l'exploitation de fichiers XML, la possibilité d'écrire facilement dans un fichier et les langages utilisés par messieurs Ly et Giang. De plus, monsieur Lieber m'avait orienté vers plusieurs langages, tels que le Java, le python ou le php. M.Giang utilisant python 2[4], je me suis donc tourné vers ce langage. J'ai ensuite approfondi mes recherches afin d'utiliser ce langage de manière adaptée. J'utilise la bibliothèque xml.etree.ElementTree[5]. Cette dernière permet de représenter le fichier XML sous forme d'arbre et d'extraire les informations voulues simplement.

Nous avons utilisé un gestionnaire de version[6] commun avec M. Giang et M. Ly afin de rassembler nos travaux respectif sur le logiciel de correction de phrase.

Pour écrire le programme, j'ai choisi de travailler avec Emacs. C'est un éditeur de texte que j'ai appris à utiliser durant la licence. Je le trouve très pratique d'utilisation pour développer en python grâce à l'indentation automatique et la coloration syntaxique. Je l'ai aussi utilisé pour lire les fichiers XML.

4 Voir référence bibliographique
5 Voir référence bibliographique
6 https://github.com/

Création de la base de cas minimale

J’ai tout d’abord fait des recherches sur l’Internet[7] et grâce à mes connaissances, sur des erreurs courantes dans des phrases en français. Puis, j’ai répertorié une dizaine de couples de phrases dans un fichier texte contenant uniquement les couples de phrases contenant une erreur et une phrase corrigée. J’ai écrit ce fichier afin qu’il puisse être utilisé par monsieur Giang pour le développement du moteur de raisonnement à partir de cas. Par la suite, j’ai modifié le format du fichier en fichier CSV[8] et ajouté des informations pour les insérer dans la base de données créée par monsieur Ly. Un fichier CSV est un format informatique représentant des données sous forme de valeurs séparées par des indentations. Cette base de cas contient les informations suivantes :

- phrase contenant une erreur, par exemple : « J’aime pas les pommes. ». Dans cet exemple, l’erreur est l’absence du « n’ » exprimant la négation.
- phrase corrigée , par exemple : « Je n’aime pas les pommes. »
- statut :
  - ‘Correcte’
  - ‘Incorrecte’
  - ‘En attente’
- mot ou groupe de mots contenant la modification avant son application : « j’aime »
- mot ou groupe de mots contenant la modification après son application : « je n’aime »
- indice de lieu de modification : « 1 » correspond au lieu où se trouve la première modification
- indice de source :
  - 1 correspond aux cas extrait du corpus WiCoPaCo
  - 2 correspond aux entrées manuelles
  - 3 correspond aux cas créés correctement par le programme
- langue, par exemple « fr »

La figure 1 donne un exemple.

J‘ai par la suite inscrit de la même manière les informations extraites du corpus WiCoPaCo.

7 Voir références bibliographiques
8 Voir références bibliographiques

PHRASE FAUSSE	PHRASE CORRIGÉ	STATUT	ERREUR	CORRECTION	INDICE DE PREMIÈRE DIFFÉRENCE	INDICE DE SOURCE	LANGUE
J'aime pas les pommes.	Je n'aime pas les pommes.	True	j'aime	je n'aime	1		2fr
Je suis sur Nancy.	Je suis à Nancy.	True	sur	à	0		2fr
Au jour d'aujourd'hui.	Aujourd'hui.	True	Au jour d'		0		2fr
J'ai été au cinéma.	Je suis allé au cinéma.	True	J'ai été	Je suis allé	1		2fr
J'amènerai une bouteille.	J'apporterai une bouteille.	True	amènerai	apporterai	1		2fr
Elle apporte son enfant.	Elle amène son enfant.	True	apporte	amène	1		2fr
Ils croivent.	Ils croient.	True	croivent	croient	4		2fr
Tu es trop nul.	Tu es très nul.	True	trop	très	2		2fr
Nous avons été à Paris.	Nous sommes allés à Paris.	True	avons été	sommes allé	0		2fr
Après qu'il ait dîné.	Après qu'il a dîné.	True	ait	a	1		2fr
Après qu'il soit arrivé, il est allé chez sa mère.	Après qu'il est arrivé, il est allé chez sa mère.	True	soit	est	0		2fr
Lui aussi n'a pas compris pourquoi elle chante.	Lui non plus n'a pas compris pourquoi elle chante.	True	aussi	non plus	0		2fr
De façon à ce que.	De façon que.	True	à ce		0		2fr
J'ai manger.	J'ai mangé.	True	manger	mangé	4		2fr

Figure 1

Traitement du corpus WiCoPaCo

Étude du fichier

J'ai téléchargé le fichier XML et j'ai étudié sa forme. J'ai constaté qu'il était organisé en couples de phrases. Un couple est composé d'une phrase initiale et d'une phrase contenant une modification de la phrase initiale. La phrase initiale est placée dans une balise <*before*>. La phrase contenant la modification est placée dans une balise <*after*>. Ces deux balises sont encadrées par une balise <*modif*> contenant plusieurs attributs : un identifiant de modification unique, un identifiant de page caractérisant la page modifiée, un identifiant de révision avant et un autre après, un identifiant utilisateur correspondant à l'utilisateur qui a fait la modification et un commentaire. Toutes les balises <*modif*> sont incluses dans une balise <*modifs*>. Dans les balises <*before*> et <*after*>, on trouve une balise <*m*>. Cette dernière est accompagnée par un attribut exprimant le nombre de mots modifiés et contenant ce ou ces mots.

La figure 2 donne un exemple.

```
 <modif id="31117" wp_page_id="7965" wp_before_rev_id="4952629" wp_after_rev_id="4952665" wp_user_id="0"
wp_user_num_modif="612674" wp_comment="Janvier">
    <before>** Vatican : Durant la messe du Nouvel An , consacrée au thème de la paix , célébrée dans la
basilique Saint-Pierre de Rome , le pape Benoît une 16 a appelé l' ONU à une conscience renouvelée de ses
responsabilités pour promouvoir la justice , la solidarité et <m num_words="2">la paix</m> dans le
monde .</before>
    <after>** Vatican : Durant la messe du Nouvel An , consacrée au thème de la paix , célébrée dans la
basilique Saint-Pierre de Rome , le pape Benoît une 16 a appelé l' ONU à une conscience renouvelée de ses
responsabilités pour promouvoir la justice , la solidarité et <m num_words="2">l' apéro</m> dans le
monde .</after>
  </modif>
```

Figure 2: exemple d'un couple en XML représentant une modification (extrait de WiCoPaCo)

Conception du programme

Afin de bien comprendre l'utilisation de la bibliothèque citée précédemment, j'ai créé un script de test que j'ai lancé sur une partie réduite du corpus WiCoPaCo. J'ai d'abord écrit un script qui extrayait uniquement les couples contenus dans la balise <*modif*>. J'ai eu quelques difficultés lors de cette étape. En effet, lors de l'utilisation des fonctions existantes pour extraire le contenu des balises <*before*> et <*after*>, seul le contenu précédant la balise <*m*> était extrait. J'ai additionné à ce bout de chaîne de caractères le contenu de la balise <*m*>, mais il me manquait toujours la fin de la phrase. J'ai constaté qu'il existait une fonction permettant d'extraire les balises et leur contenu. Je me suis aperçu qu'en utilisant cette fonction sur la balise <*m*> j'obtenais aussi la fin de la phrase. La chaîne de caractères obtenu avait la forme suivante :

<?xml version='1,0' encoding='utf8'?>

<m num_words= »2 »>la paix</m> dans le monde .

Seul les caractères présents après </*m*> m'étaient utiles. j'ai donc réfléchi à un algorithme permettant de garder uniquement les caractères suivant cette balise. Voici l'algorithme :

```
chaine = '<?xml version='1,0' encoding='utf8'?>\n<m num_words= »2 »>la paix</m> dans le monde .'

compteur = len(chaine) #retourne la longueur de la chaîne de caractères

fin_chaine =' ' #initialisation d'une chaîne qui contiendra la fin de la chaîne

while ( chaine[compteur] !=  '> ') :

        fin_chaine = chaine[compteur] + fin_chaine

        compteur = compteur-1
```

Après l'algorithme, la variable fin_chaine contiendra tous les caractères suivant la balise </*m*>.

Cela m'a permis de pouvoir ajouter les phrases complètes à la base de cas.

La figure 3 présente une exécution du programme de test.

```
damien@damien-pc-hp:~/Documents/STAGE/Test$ python test.py
arbre créé
-------

nom de l'element racine : <modifs>

-----

test de lecture du contenu de la première balise <before> :

** Vatican : Durant la messe du Nouvel An , consacrée au thème de la paix , célébrée dans
la basilique Saint-Pierre de Rome , le pape Benoît une 16 a appelé l' ONU à une conscience
 renouvelée de ses responsabilités pour promouvoir la justice , la solidarité et

-----
test de lecture de la balise <m> et son contenu :

<?xml version='1.0' encoding='utf8'?>
<m num_words="2">la paix</m> dans le monde .
chaine après suppression des caractères inutiles :
 dans le monde .
```

Figure 3: Exemple d'utilisation du programme de tests

Après discussion avec messieurs Bruno Guillaume, Yves Lepage, Jean Lieber et Emmanuel Nauer, nous avons convenu d'ajouter à la base de cas le ou les mots modifiés. Ils m'ont aussi demandé d'ajouter la source du cas, un indice indiquant le lieu de début de modification dans le ou les mots modifiés et une pertinence des cas qui pourra être utilisée par le moteur de recherche. Nous avons aussi convenu que le fichier créé serait au format CSV afin de faciliter l'insertion des couples dans la base de données.

La figure 4 montre une exécution du programme sans utilisation de filtre.

```
damien@damien-pc-hp:~/Documents/STAGE/Test/traitement_données$ python traitement
_wikopaco_1.py
nom du fichier à traiter? wicopaco_v2.xml
created tree
root element : modifs
comment voulez vous nommer le fichier de sortie ?
le fichier sera automatiquement suivi de 'traite.csv'
sans_filtre_
file sans_filtre_traite.csv open
file sans_filtre_traite.csv close
-------
nombre d'element traités = 408816
damien@damien-pc-hp:~/Documents/STAGE/Test/traitement_données$
```

Figure 4: Utilisation du programme d'extraction de cas à partir de WiCoPaCo

Filtrage des données

Lors de l'étude du corpus, j'ai constaté qu'il existait un certain nombre de cas ne traitant pas de correction de français. Afin de supprimer les cas inutiles pour le logiciel de correction de phrases et d'avoir un maximum de cas pertinents, j'ai approfondi l'étude du corpus afin d'avoir une idée plus précise des cas n'étant pas adaptés. Je me suis d'abord demandé s'il n'était pas possible d'utiliser les commentaires présents dans les attributs des balises *<modif>*. Les figures 5, 6, 7 sont des exemples de commentaires extrait du corpus.

```
  <modif id="2" wp_page_id="3" wp_before_rev_id="4529726" wp_after_rev_id="4847377" wp_user_id="7343"
wp_user_num_modif="319" wp_comment="ortho">
```

Figure 5: Exemple 1

```
  <modif id="1" wp_page_id="3" wp_before_rev_id="1350936" wp_after_rev_id="1409789" wp_user_id="3763"
wp_user_num_modif="15342" wp_comment="correction de numéraux">
```

Figure 6: Exemple 2

```
  <modif id="4" wp_page_id="3" wp_before_rev_id="18398158" wp_after_rev_id="19601285" wp_user_id="0"
wp_user_num_modif="612471" wp_comment="">
```

Figure 7: Exemple 3

Ces commentaires n'étant pas normalisés, comme nous pouvons le constater sur les figures 5, 6 et 7, cela rend leur utilisation, pour créer un filtre, compliquée. De plus, beaucoup de commentaires sont vides. Je me suis donc mis à la recherche d'autres filtres possibles. Ce corpus répertoriant tous les types de modifications faites, il contient aussi les modifications successives revenant au cas initial.

La figure 8 en est un exemple.

```
<modif id="15" wp_page_id="7" wp_before_rev_id="1599401" wp_after_rev_id="1599406" wp_user_id="0"
wp_user_num_modif="612471" wp_comment="Quelques théorèmes">
    <before>Un théorème intéressant à l' époque des mémoires d' ordinateurs de <m num_words="1">petite</m>
taille était qu ' on pouvait travailler séparément sur des sous-ensembles (« blocs ») d' une matrice en
les combinant ensuite par les mêmes règles qu ' on utilise pour combiner des scalaires dans les
matrices . Avec les mémoires actuelles de plusieurs Go , cette question a perdu un peu de son intérêt
pratique , mais reste très prisée en théorie des nombres , pour la décomposition en produit de facteurs
premiers avec le crible général de corps de nombres ( GNFS ) ( méthode Lanczos par blocs )</before>
    <after>Un théorème intéressant à l' époque des mémoires d' ordinateurs de <m num_words="1">grd</m>
taille était qu ' on pouvait travailler séparément sur des sous-ensembles (« blocs ») d' une matrice en
les combinant ensuite par les mêmes règles qu ' on utilise pour combiner des scalaires dans les
matrices . Avec les mémoires actuelles de plusieurs Go , cette question a perdu un peu de son intérêt
pratique , mais reste très prisée en théorie des nombres , pour la décomposition en produit de facteurs
premiers avec le crible général de corps de nombres ( GNFS ) ( méthode Lanczos par blocs )</after>
  </modif>
  <modif id="16" wp_page_id="7" wp_before_rev_id="1599406" wp_after_rev_id="1648108" wp_user_id="0"
wp_user_num_modif="612471" wp_comment="Quelques théorèmes">
    <before>Un théorème intéressant à l' époque des mémoires d' ordinateurs de <m num_words="1">grd</m>
taille était qu ' on pouvait travailler séparément sur des sous-ensembles (« blocs ») d' une matrice en
les combinant ensuite par les mêmes règles qu ' on utilise pour combiner des scalaires dans les
matrices . Avec les mémoires actuelles de plusieurs Go , cette question a perdu un peu de son intérêt
pratique , mais reste très prisée en théorie des nombres , pour la décomposition en produit de facteurs
premiers avec le crible général de corps de nombres ( GNFS ) ( méthode Lanczos par blocs )</before>
    <after>Un théorème intéressant à l' époque des mémoires d' ordinateurs de <m num_words="1">petite</m>
taille était qu ' on pouvait travailler séparément sur des sous-ensembles (« blocs ») d' une matrice en
les combinant ensuite par les mêmes règles qu ' on utilise pour combiner des scalaires dans les
matrices . Avec les mémoires actuelles de plusieurs Go , cette question a perdu un peu de son intérêt
pratique , mais reste très prisée en théorie des nombres , pour la décomposition en produit de facteurs
premiers avec le crible général de corps de nombres ( GNFS ) ( méthode Lanczos par blocs )</after>
  </modif>
```

Figure 8: Exemple de modification non pertinente.

Dans la première balise <*modif*>, on peut voir que la modification concerne le mot *petite* qui est modifié en *grd*. Dans la seconde balise <*modif*>, nous voyons que le mot *grd* est à nouveau modifié en *petite*. Ces deux couples ne sont donc pas utiles. J'ai donc continué la lecture du corpus, afin de constater que ce n'était pas un cas isolé, mais un cas récurant. J'ai cherché d'autres similitudes dans ces cas afin de faciliter leurs suppressions. C'est à ce moment là que j'ai constaté que *wp_after_rev_id* de la première modification était égale à *wp_before_rev_id* de la seconde modification. J'ai aussi observé que ces cas n'était pas forcément successifs dans le corpus. J'ai donc écrit une fonction comparant les attributs *wp_after_rev_id* et *wp_before_rev_id* dans les cas suivants. Cette fonction modifie directement le fichier XML. L'utilisation de cette fonction est très longue sur le corpus WiCoPaCo (approximativement 8h sur mon ordinateur personnel ayant un processeur intel core i5-6300HQ 2,30GHz*4). Elle a cependant permis de retirer un certain nombre de cas inutiles.

La figure 9 est un affichage du résultat du programme avec le filtre décrit précédemment.

```
damien@damien-pc-hp:~/Documents/STAGE/Test/traitement_données$ python traitement
_wicopaco.py
nom du fichier à traiter? wicopaco_v2.xml
created tree
root element : modifs
start suppr
54177 couple suppr
comment voulez vous nommer le fichier de sortie?
le fichier sera automatiquement suivi de 'traite.csv'
wicopaco_v2_
file wicopaco_v2_traite.csv open
file wicopaco_v2_traite.csv close
-------
nombre d'element traité = 300462
```

Figure 9: Utilisation du programme avec suppression de cas.

État du travail

À l'issu du stage, une base de cas initiale a été créée puis insérée dans la base de données développée par M. Ly. Un programme d'extraction de l'ensemble des cas présents dans le corpus de WiCoPaCo a été écrit. Un algorithme supprimant les cas contenant des modifications successives revenant au cas initial a aussi été développé. Le programme développé et les fichiers CSV sont disponibles sur Github[9].

Améliorations futures possibles

Avec plus de temps, plusieurs améliorations sont possibles. On pourra optimiser l'algorithme filtrant déjà mis en place, afin d'améliorer ses temps de calcul. Des filtres supplémentaires pourront aussi être ajoutés afin d'affiner la pertinence des couples de cas présents dans la base de données. Il sera possible de détecter la modification de dates par exemple. La figure 10 est un exemple de modification de dates qui n'apporte pas d'utilité dans le programme de correction de phrase.

```
  <modif id="201" wp_page_id="16" wp_before_rev_id="13009940" wp_after_rev_id="13019866"
wp_user_id="190750" wp_user_num_modif="52" wp_comment="Correction faute de frappe énorme dans une
date :-/">
    <before>L' Arc de triomphe de l' Étoile est inauguré en <m num_words="1">1936</m> par Louis-Philippe
Ier .</before>
    <after>L' Arc de triomphe de l' Étoile est inauguré en <m num_words="1">1836</m> par Louis-Philippe
Ier .</after>
  </modif>
```

Figure 10: Exemple de modification de dates.

J'ai constaté que des modifications changeaient le sens des phrases, un filtre basé sur l'étude sémantique des cas peut être imaginé. La figure 11 illustre ce type de cas.

```
  <modif id="295" wp_page_id="23" wp_before_rev_id="2333007" wp_after_rev_id="2334483" wp_user_id="0"
wp_user_num_modif="612472" wp_comment="Histoire">
    <before>En 2004 , 2 ans après l' <m num_words="1">invasion</m> internationale , l' Afghanistan est
redevenu le premier pays producteur mondial de pavot ( papaver somniferum ) dont le latex est utilisé
pour produire l' opium et l' héroïne .</before>
    <after>En 2004 , 2 ans après l' <m num_words="1">intervention</m> internationale , l' Afghanistan est
redevenu le premier pays producteur mondial de pavot ( papaver somniferum ) dont le latex est utilisé
pour produire l' opium et l' héroïne .</after>
  </modif>
```

Figure 11: Exemple de modification de sens.

Dans la figure 11, le mot *invasion* est remplacé par le mot *intervention*. Ces deux mots ne sont pas synonymes. Ce type de cas est à supprimer de la base de cas.

9 https://github.com/Sle3pingForest/StageL3/tree/master/BASECASE

Conclusion

À la fin de ce stage, une base de cas initiale s'appuyant sur mes recherches à été créée. Un programme d'extraction des couples de phrases avant/après correction du corpus WiCoPaCo à été écrit. J'ai rencontré des difficultés dans l'écriture de ce programme car je n'arrivais pas à extraire l'intégralité de chaque phrase contenue dans chaque cas du corpus. J'ai aussi implémenté un algorithme qui permet de supprimer des couples de phrases qui n'ont pas d'intérêt dans la correction de langue. J'ai aussi rencontré des difficultés à ce moment là car je n'ai pas réussi à faire un algorithme optimisé.

Le fait d'avoir beaucoup travaillé en groupe durant la licence m'a permis de m'adapter rapidement au travail d'équipe avec messieurs Giang et Ly. La manière dont nous avons appris à programmer indépendamment des langages durant la licence m'a été bénéfique car j'ai pu m'adapter facilement à l'utilisation de langages que j'avais peu utilisés avant ce stage.

Ce stage m'a apporté une première expérience dans un laboratoire de recherche en informatique. Cela m'a permis de faire des recherches sur le français et d'en apprendre un peu plus sur ma langue maternelle et ses erreurs courantes.

Références bibliographiques/Webographie

Loria : http://www.loria.fr/fr/presentation/

Dernière consultation : juin 2018

WiCoPaCo Corpus : https://wicopaco.limsi.fr/

Dernière consultation : juin 2018

LCI : https://www.lci.fr/livre/ces-treize-grosses-fautes-de-francais-que-vous-commettez-sans-vous-en-apercevoir-1505867.html

Dernière consultation : mai 2018

Le Figaro . Fr : http://www.lefigaro.fr/langue-francaise/expressions-francaises/2017/04/16/37003-20170416ARTFIG00003-j-ai-ete-a-paris-ne-faites-plus-la-faute.php

Dernière consultation : mai 2018

OSEZ ÉCRIRE VOTRE ROMAN : http://www.osez-ecrire-votre-roman.com/20-fautes-de-syntaxe-barbarismes-et-autres-erreurs-de-francais-a-bannir/

Dernière consultation : mai 2018

le web pédagogique : http://lewebpedagogique.com/ressources-fle/les-fautes-de-francais-a-eviter/

Dernière consultation : mai 2018

L'OBS : http://la-conjugaison.nouvelobs.com/

Dernière consultation : mai 2018

Français Facile : https://www.francaisfacile.com/exercices/exercice-francais-2/exercice-francais-9307.php

Dernière consultation : juin 2018

Wikipédia : https://fr.wikipedia.org/wiki/Comma-separated_values

Dernière consultation : mai 2018

Documentation python : https://docs.python.org/fr/2.7/library/xml.etree.elementtree.html

Dernière consultation : mai 2018

Github : https://github.com/Sle3pingForest/StageL3

Dernière consultation : juin 2018