

André GIANG

Rapport de stage

Un moteur d'inférences pour la correction à partir de cas de phrases en français



Du 9 avril au 6 juin 2018

Responsable de stage :
Bruno GUILLAUME
Yves LEPAGE
Jean LIEBER
Emmanuel NAUER

Tuteur universitaire :
Mathieu CONSTANT

Licence 3 informatique

Table des matières

Avant propos.....	1
Remerciements.....	2
Introduction.....	3
I - Description de l'entreprise.....	5
II - Choix et analyse.....	6
III - Travail réalisé.....	8
3.1.1 - Distance LCS.....	9
3.1.2 - Niveau d'indexation.....	11
3.1.3 - Comparatif.....	12
3.1.4 - Récupération des données.....	12
3.2.1 - Raisonnement.....	13
3.2.2 - Algorithme de l'adaptation:.....	14
IV - Travaux futurs et améliorations.....	15
V - Conclusion.....	16
Bibliographie / Sitographie.....	17

Avant propos

La première phase de ce stage consistera à définir une première version de la remémoration et de l'adaptation, s'appuyant sur des techniques connues :

— La remémoration consistera à appliquer une distance d'édition classique sur les chaînes de caractères : le cas remémoré ($srce$, $sol(srce)$) sera celui qui minimisera la distance de $srce$ à cible (avec un choix arbitraire en cas d'ex æquo).

— L'adaptation s'appuiera sur un système déposé en ligne par Yves Lepage et ses collaborateurs, qui fait des analogies entre chaînes de caractères. Le langage de programmation, imposé par l'utilisation de la bibliothèque de Yves Lepage, sera Python. Une fois ce travail effectué, il devra être testé sur un ensemble d'exemples. La troisième phase de ce stage, plus exploratoire, consistera à trouver des améliorations, en particulier sur la phase de remémoration :

— Implanter d'autres distances et tester la différence des résultats.

— Implanter une approche s'appuyant sur une indexation des cas sources dans l'optique d'accélérer le processus de remémoration (si le nombre de cas devient trop important).

Les étapes décrites ont été généralement respectées, cependant le dernier point qui consiste dans l'indexation des cas afin d'accélérer la recherche a été abordé mais n'a pas été développé.

Au final le stage s'est plutôt bien déroulé dans l'ensemble et le moteur de correction est fonctionnel et utilisable.

Remerciements

Je tiens à remercier Monsieur Jean LIEBER qui m'a encadré et conseillé durant toute la durée du stage.

Ainsi que Monsieur Yves LEPAGE qui m'a beaucoup aidé notamment grâce au partage de son travail.

Messieurs Bruno GUILLAUME et Emmanuel NAUER qui m'ont aidé à la rédaction du rapport et la présentation.

Mathieu CONSTANT pour ses enseignements durant la licence et sa tutelle pendant le stage.

Mes partenaires durant le stage Hue Nam LY et Damien LEVY pour leur aide.

Introduction

De nos jours, nous sommes aidés et entourés de beaucoup d'outils qui nous permettent de modifier et corriger nos fautes d'orthographe. Cependant aujourd'hui et de tout temps nous avons toujours eu de nombreuses déformations et « raccourcis » grammaticaux sans même s'en rendre compte. Le but de ce stage est d'arriver à corriger une phrase.

Par exemple « *Je suis sur Nancy* » devrait être « *Je suis à Nancy* ».

Pour ce faire nous allons utiliser le raisonnement à partir de cas (RàPC) qui consiste à résoudre un problème à partir d'expériences déjà réalisés, ici nous utiliserons une base de cas qui contient un ensemble de couples (problème, solution du problème).

Voici un exemple :

- Notre phrase à corriger
J'aime pas les araignées.
- Un couple qui servira à la correction du problème
J'aime pas nager. Je n'aime pas nager.
- Notre correction à partir de cas donnée :
J'aime pas les araignées. Je n'aime pas les araignées.

Le RàPC se déroule en quatre étapes :

- Remémoration : Choix d'un cas dans la base jugé ressemblant au problème demandé.
- Adaptation : Correction et modification du problème demandé en une correction à partir du cas choisi.
- Révision : Une fois qu'une correction est générée par le moteur, elle doit être vérifiée et validée. Cette étape est externe au moteur et sera réalisée manuellement par les administrateurs de notre site.
- L'apprentissage : Une fois l'étape de révision faite, il nous faut ajouter le nouveau cas dans la base avec les informations importantes permettant ainsi de nouvelles ou meilleures corrections.

C'est un outil qui peut donc 'apprendre' très vite si beaucoup de cas lui sont donnés, dans notre groupe ce sera Monsieur Levy qui se chargera d'alimenter la base de cas, manuellement au départ puis en automatisant avec un script permettant la récupération de cas trouvé sur le Web et notre trinôme Monsieur Ly se chargera de créer la base de données ainsi qu'un site vitrine donnant l'accès au moteur. Mon rôle à moi est donc le développement du moteur permettant la correction de phrase en français nous verrons plus particulièrement les deux premières étapes du RàPC la remémoration et l'adaptation.

Dans le cadre de ce stage nous avons reçu une maquette des travaux d'Yves Lepage et de ses collaborateurs. En effet ceux-ci ont développés un outil qui utilise le raisonnement à partir de cas qui permet de traduire une phrase d'une langue à une autre.

Nous allons nous servir de ces travaux afin de développer en Python un outil capable de proposer une correction d'une phrase cible en français, en s'appuyant sur le principe de raisonnement à partir de cas.

Je vais vous présenter le déroulement du développement du moteur de correction de phrases en plusieurs parties :

- L'explication et le choix des outils utilisés ;
- La phase d'analyse et de compréhension des travaux d'Yves Lepage.

Les parties suivantes seront les différentes étapes du développement du moteur :

- L'adaptation ;
- Remémoration.

Pour finir sera présenté les difficultés rencontrées et apports éventuels du moteur :

- Les améliorations éventuelles et futures ;
- Conclusion.

I - Description de l'entreprise

Le stage s'est déroulé au Loria, Laboratoire lorrain de recherche en informatique et ses applications situé à Nancy.

Ce laboratoire est une Unité Mixte de Recherche (UMR), il est commun à plusieurs établissements le CNRS, l'université de Lorraine et Inria.

Créé depuis 1997 son domaine est la recherche fondamentale et appliquée en sciences informatiques.

Leurs travaux scientifiques sont menés au sein de 28 équipes structurées en 5 départements, dont 15 sont communes avec Inria, représentant un total de plus de 400 personnes. Le Loria est un des plus grands laboratoires de la région lorraine.

Source Loria.

II - Choix et analyse

Dans le cadre du stage j'ai développé le moteur avec le langage Python dans sa version 2.

Ce choix s'est fait assez naturellement, en effet les travaux d'Yves Lepage étaient en Python 2.7 et dans un souci de cohérence nous avons décidé de garder le même langage de programmation.

N'ayant encore jamais développé sous Python il a fallu que j'apprenne ce nouveau langage.

Une fois cela décidé et fait j'ai récupéré et analysé le code de la maquette qu'Yves Lepage a développé qui utilise des analogies.

Une analogie est une association d'idée entre deux ou plusieurs objets de nature différente.

Voici une analogie :

$a : b :: c : d$

qui se lit « a est à b ce que c est à d »

Quelques analogies avec des mots en français :

Avion : voler :: Voiture : rouler

Faim : manger :: Soif : boire

manger : manges :: ranger : ranges

J'aime pas les marshmallow. : Je n'aime pas les marshmallow. ::

J'aime pas nager dans le lac du connemara. : Je n'aime pas nager dans le lac du connemara.

Un exemple du fonctionnement de la maquette d'Yves Lepage :

Phrase à traduire

They love apples.

- Remémoration

Cas dans la base

They love pears. Ils aiment les poires.

- Adaptation

Mot à traduire

pear | apple poire | pomme

Remplacement de poires par pommes

Ils aiment les poires. Ils aiment les pommes.

Résultat

They love apples. Ils aiment les pommes.

Un exemple d'équation analogique mathématique :

$6 / 9 :: 1 / x$

On cherche à déterminer x

En connaissant le rapport des 2 premiers termes on arrive à trouver le rapport entre le 3ème et 4ème.

si $6 / 9 = 8 / x$ alors $x = 12$.

Mon but sera donc de résoudre ces équations analogiques sur des chaînes de caractères. À partir d'un problème cible et d'un cas choisi dans la base je dois retourner une solution au problème cible.

Problème source : solution source :: problème cible : x

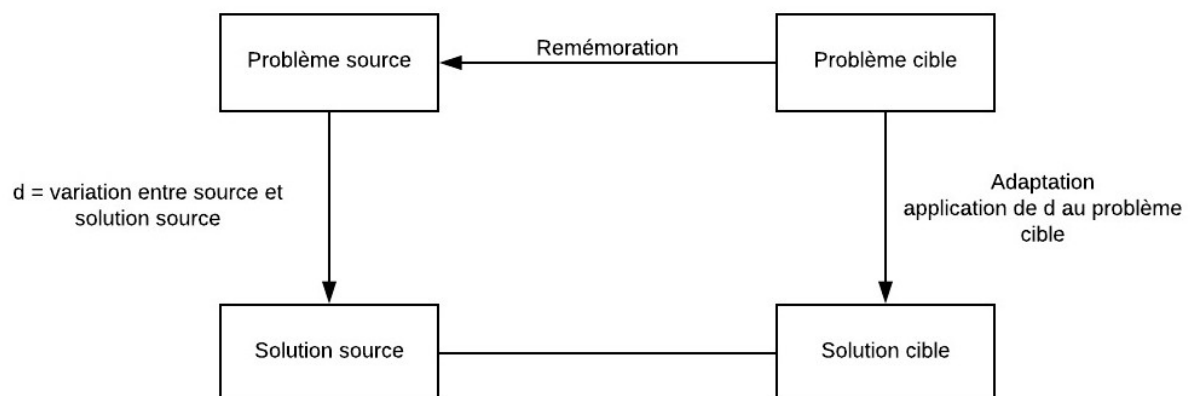
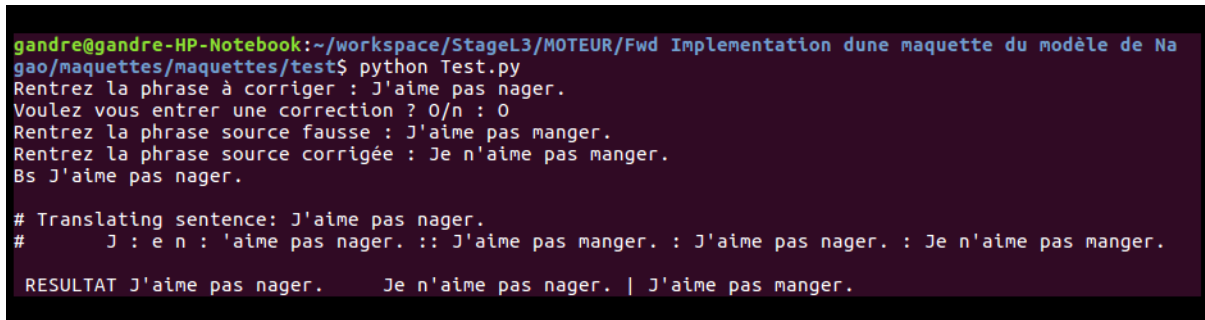


Figure 1 : carré analogique

III - Travail réalisé

Je me suis appuyé sur un existant qui est le code de la maquette d'Yves Lepage et l'ai modifié puis adapté selon nos besoins et fonctionnement.

Une interface en ligne de commande a été ajoutée afin d'être plus facile à comprendre et mieux indiquer les marches à suivre pour tester le moteur.



```
gandre@gandre-HP-Notebook:~/workspace/Stagel3/MOTEUR/Fwd Implementation dune maquette du modèle de Na
gao/maquettes/maquettes/test$ python Test.py
Rentrez la phrase à corriger : J'aime pas nager.
Voulez vous entrer une correction ? 0/n : 0
Rentrez la phrase source fausse : J'aime pas manger.
Rentrez la phrase source corrigée : Je n'aime pas manger.
Bs J'aime pas nager.

# Translating sentence: J'aime pas nager.
#      J : e n : 'aime pas nager. :: J'aime pas manger. : J'aime pas nager. : Je n'aime pas manger.

RESULTAT J'aime pas nager.      Je n'aime pas nager. | J'aime pas manger.
```

Figure 2 : Capture d'écran du test du moteur

Je vais expliquer en détails les deux premières étapes du raisonnement à partir de cas la remémoration et l'adaptation qui sont celles que j'ai développées :

3.1 - Remémoration

La remémoration est l'étape qui permet de choisir le/les meilleures cas pour la résolution d'un problème cible.

Pour un exemple comme celui-ci :

Salut je manges.

Comment retrouver le bon cas pour avoir la bonne correction ?

On pourrait naturellement se dire, on détecte l'utilisation du pronom personnel 'je' + verbe et rechercher un cas similaire dans la base.

Ce n'est pas le cas pour notre moteur en effet celui-ci a été conçu avec le principe le plus simple possible juste en travaillant sur les chaînes de caractères sans analyse sémantique, lexicale.

3.1.1 - Distance LCS

Du coup sans analyse du contexte de la phrase notre première piste a été d'utiliser la distance LCS afin de classifier les phrases par ordre de ressemblance.

Longest common subsequence (plus longue sous-séquence commune) :

C'est donc la plus longue chaîne de caractère commune entre deux phrases dans notre cas.

La spécificité est que les caractères n'ont pas besoins d'être consécutif à l'instar d'une autre distance LCS (longest common substring) où les caractères doivent se suivre.

Par exemple pour les deux chaînes de caractères suivantes :

« bonjour »

« tondre »

la plus longue sous-séquence commune est « onr »

La distance LCS que j'utilise quand à elle calcule le nombre de caractère manquants et à rajouter entre une chaîne de référence et une deuxième chaîne à comparer.

$$\text{Dist LCS} = \sum |S1|_a - |S2|_a|, \dots, |S1|_z - |S2|_z|$$

La notation $|S1|_a$ est le nombre d'occurences de la lettre a dans la chaîne S1.

Après quelques tests je me suis rendu compte que la distance LCS n'était pas la bonne méthode pour la remémoration.

Voici un exemple :

Problème cible « Tu peux appelé ? »

Cas1(« Tu peux demandé.», « Tu peux demand~~er~~.»)

Cas2(« Il peut apeler.», « Il peut app~~e~~ler. »)

distance_cas1(Tu peux appelé? , Tu peux demandé.) = 11

distance_cas2(Tu peux appelé?, Il peut apeler.) = 10

ou plus flagrant :

Problème cible « Tu peux appelé ? »

Cas1(« J'aime les Litchys.», « J'aime les Litchis.»)

Cas2(« Salut j'ai un problème tu peux appelé chez moi.», « Salut j'ai un problème tu peux appelle^r chez moi . »)

distance_cas1(Tu peux appelé? , J'aime les Litchys.) = 27

distance_cas2(Tu peux appelé?, Salut j'ai un problème tu peux appelé chez moi.) = 39

On voit qu'il y a beaucoup de caractères 'parasites' qu'on ne voudrait pas prendre en compte. Cela est d'autant plus problématique car la base dont les cas proviennent de WiCoPaCo peuvent contenir plusieurs phrases dans un problème. Il a donc fallu réfléchir à un autre moyen de choisir les cas dans la base.

3.1.2 - Niveau d'indexation

Nous nous sommes rendus compte que prendre en compte la totalité des phrases n'était pas efficace et que ce qui importait en plus de la chaîne à substituer était aussi ce qu'il y avait autour, le contexte de la phrase.

Par exemple pour la négation *Je veux pas.* → *Je ne veux pas.* Le « pas » est important.

Pour une conjugaison au participe passé *Elle est allé.* → *Elle est allée.* L'utilisation d'un pronom et plus particulièrement *Elle*.

On a donc décidé de récupérer les mots adjacents à gauche et à droite de la chaîne à substituer.

A l'indice 0 `index_0` on récupère la chaîne qui va être substituer.

A l'indice 1 on récupère la chaîne qui va être substituer et 1 mot à gauche et à droite de la chaîne.

A l'indice 2 on récupère 2 mots à gauche et à droite de la chaîne.

Et ainsi de suite.

Pour l'instant on a jugé qu'un indice maximum de niveau 2 était suffisant.

Cible « Je suis sur Paris. »

Couple1 (Je suis sur Nancy., Je suis à Nancy.)

Modification droite et gauche : (**sur**, **à**)

Index_0 « **sur** »

Index_1 « suis **sur** Nancy. »

Index_2 « Je suis **sur** Nancy. »

Couple2 (Le monde est triste., Le monde est triste.)

Modification droite et gauche : (, **s**)

Index_0 « »

Index_1 « et »

Index_2 « monde et triste »

Une fois ces index récupéré le moteur va compter le nombre de mots différents entre les index de même niveau et le problème cible. Le moteur choisira le minimum s'ils ont le même nombre de mots différents il prend les index de niveau supérieur et recommence.

3.1.3 - Comparatif

Capture d'écran et remarques à venir

3.1.4 - Récupération des données

Au départ Monsieur Levy a créé manuellement une dizaine de cas d'erreur courante en français dans un fichier au format CSV, c'est ce dont je me suis servi pendant le stage pour tester le moteur et les différentes étapes. Une base de données contenant les cas récupéré automatiquement est prévue. La base ayant été développée sous Mysql j'ai utilisé une librairie capable d'exécuter des requêtes sur cette base.

3.2 - Adaptation

3.2.1 - Raisonnement

Après avoir choisi un cas ressemblant on va appliquer le même rapport à la solution cible, dans notre moteur on va substituer des chaînes de caractères.

Adaptation((problème source, solution source), problème cible) = solution cible.

Problème cible «J'aime pas nager.»

Cas «J'aime pas les serviettes.», «Je n'aime pas les serviettes.»

Pour détecter la chaîne à remplacer et sa position dans la phrase j'utilise des algorithmes permettant de récupérer les préfixes et suffixes communs entre deux chaînes de caractères.

Préfixe commun «J»

suffixe commun «'aime pas les serviettes»

Différence dans le problème source «>>

Différence dans la solution source « e n »

Position du changement à l'indice 1 car on commence 0.

3.2.2 - Algorithme de l'adaptation:

Calcul des préfixe suffixe commun (problème cible , problème source)

si oui (ressemblance dans la fin et/ou le début)

Calcul préfixe suffixe commun problème source, solution source

récupération de la sous chaîne à remplacer et sa position dans le problème source

Découpage dans problème cible

Constitution d'un triplet prefixe, sous chaine, suffixe

sinon

Utilisation de LCS (détection d'une sous chaîne consécutive ressemblante dans la chaîne)

si oui

Calcul préfixe suffixe commun problème source, solution source

récupération de la sous chaîne à remplacer et sa position dans le problème source

Découpage dans problème cible

Constitution d'un triplet prefixe, sous chaine, suffixe

sinon

retourne le problème cible

retourne le triplet

IV - Travaux futurs et améliorations

Tester les moteurs de manière plus complète et approfondie avec la base de cas filtrée ou non et dans d'autres langues, afin de réaliser des statistiques pour comparer les différentes méthodes de remémoration et chercher aussi d'autres méthodes plus efficaces.

Classer et catégoriser les cas sous forme d'arbre permettant une recherche moins coûteuse et plus rapide dans la base.

Modifier ou chercher une approche différente de l'adaptation qui utilise des préfixes et suffixes communs mais toujours en travaillant sur les chaînes de caractères.

Chercher d'autres approches d'adaptation comme par exemple utiliser une analyse plus précise et poussée en analysant la structure sémantique et lexicale des phrases.

V - Conclusion

Le stage s'est globalement bien déroulé le moteur est utilisable et les points principaux du stage ont été respectés. Une des plus grosses difficultés a été l'apprentissage d'un nouveau langage de programmation, la reprise d'un code existant a apporté son lot d'avantages et d'inconvénients :

- D'un côté l'analyse et la compréhension d'un autre code est toujours plus ou moins fastidieux dans mon cas le code était clair et bien commenté, il contenait toutefois quand même des notions plus poussées que mes connaissances.
- L'existence d'une base déjà développée a permis d'éviter de commencer de 0.

Les divers projets développés en groupe et seul durant la licence m'ont permis une rapide mise en route au sein du projet.

Le développement d'un projet concret et utilisable, le travail en groupe a été très motivant. La fonctionnalité du moteur dans une première version peut être la base d'un projet plus poussé car il y a de nombreuses voies d'améliorations.

Bibliographie / Sitographie

- Analogy for Natural Language Processing and Machine Translation – Yves Lepage
- Wikipédia.org
- <https://openclassrooms.com/courses/apprenez-a-programmer-en-python>
- <https://docs.python.org/3/tutorial/datastructures.html>
- <https://stackoverflow.com/questions/1187970/how-to-exit-from-python-without-raise>
- <http://www.loria.fr/fr/presentation/>