E-Lexic : Extraction lexicale à partir de textes bilingues alignés

Florent Marié

E-Lexic : Extraction lexicale à partir de textes bilingues alignés

Florent
MARIÉ

Sous la direction de Olivier KRAIF

UFR LLASIC
Département Sciences du Langage

Mémoire de master 2 mention Sciences du Langage - 20 crédits

Parcours : Industrie de la Langue, orientation professionnelle

Année universitaire 2019-2020

Université Grenoble Alpes

E-Lexic : Extraction lexicale à partir de textes bilingues alignés

Florent MARIÉ

Sous la direction de Olivier KRAIF

UFR LLASIC
Département Sciences du Langage

Mémoire de master 2 mention Sciences du Langage - 20 crédits

Parcours : Industrie de la Langue, orientation professionnelle

Année universitaire 2019-2020

Remerciements

Je tiens tout d'abord à remercier Olivier Kraif, pour m'avoir épaulé pendant cette période de stage, pour m'avoir aidé à m'orienter vers un nouveau sujet de mémoire quand mon stage en entreprise à dû être écourté à cause de la crise sanitaire. Merci pour ces deux années de Master durant lesquels vos enseignements en algorithmique et en Python m'ont permis d'en arriver là.

Je remercie également Véronique Aubergé et Claude Ponton, premièrement pour leurs enseignements tout au long du Master, mais aussi pour avoir accepté de faire partie du jury de la soutenance de ce mémoire.

Je remercie ensuite l'entreprise ALTICA-Traductions de son accueil chaleureux, et particulièrement Annelise Jost qui m'a accordé sa confiance en me prenant en stage. Je remercie de même Clarisse Bayol qui a été ma tutrice de stage au sein de l'entreprise pendant cette courte période.

Un grand merci aussi à toute l'équipe des « jeudiens » : Happy pour sa bonne humeur, Naïs pour m'avoir rassuré sur l'écriture d'un mémoire, Quentin pour les balades avec les chiens. Pour tous les autres que je ne cite pas parce que ça fait beaucoup : merci à vous et à jeudi !

Encore un merci particulier à Mel, d'être comme une sœur pour moi et de bien vouloir écouter tous mes tracas quand j'en ai. Et prendre l'air en allant faire du skate les matins a été plus efficace que tout le café imaginable pour la rédaction du mémoire.

Merci à mes colocs Anaïs et Manon d'avoir partagé mon quotidien pendant plus 5 ans et pour avoir bien voulu relire mon mémoire. Vous êtes les meilleures !

Je remercie aussi Lucie, celle qui partage ma vie depuis maintenant 2 ans, qui compte plus que tout le reste, et qui est sans aucun doute la femme de ma vie.

Je dédie ce travail à celle qui a toujours cru en moi quand j'avais peur, quand j'hésitais, même quand j'avais la flemme. Celle qui a patienté tout le temps qu'il m'a fallu pour trouver ma voie sans jamais me presser. Tu savais que je pouvais réussir, tu as su que ça prendrait le temps qu'il faudrait. J'en suis là grâce à toi. Merci Maman.

Sommaire

Table des matières

Avertissement

Le travail présenté dans ce mémoire n'est pas un rapport de stage au sens habituel, dans la mesure où le stage commencé a été interrompu par la crise sanitaire et n'a pu reprendre par la suite.

En remplacement du mémoire de stage, il m'a été demandé de réaliser un travail de développement plus léger, en autonomie, sur un sujet de mon choix – puis de rédiger un rapport court.

Finalement j'ai choisi un sujet proposé par mon enseignant référent autour de la question de l'alignement bilingue.

L'essentiel du travail présenté est donc un travail effectué seul, sous la supervision de mon enseignant référent, mais toujours à distance.

Introduction

Ce mémoire est le fruit d'un travail réalisé dans le cadre d'un stage de fin d'études dans le domaine du traitement automatique des langues. Il découle d'une motivation renforcée au fil des années par un attrait pour les langues étrangères, et d'un intérêt particulier pour la traduction automatique. Cette dernière a pendant longtemps été considérée comme un outil peu efficace, engageant son utilisateur à devoir constamment rester sur ses gardes quant aux traductions obtenues par ce biais. Ce n'est que depuis quelques années, notamment grâce à l'évolution des technologies et des méthodes de traitement automatique des langues, qu'il nous a été donné de remarquer une avancée dans la fiabilité des outils de traduction assistée par ordinateur.

1. Le stage

À la suite d'un stage de fin d'études en entreprise entamé en mars, interrompu par les événements sanitaires qui ont eu lieu en 2020, j'ai commencé à prendre goût au domaine de la traduction, et particulièrement la traduction assistée par ordinateur. Durant ma recherche d'un nouveau sujet pouvant être abordé pour mon mémoire de fin d'études, il m'a été proposé par Olivier Kraif de travailler sur l'alignement de mots, outil souvent utilisé en traduction automatique. J'ai donc naturellement orienté mon travail vers ce sous-domaine de la traduction assistée par ordinateur : l'alignement automatique de textes bilingues[1].

2. Les objectifs du projet

L'objectif principal de ce projet était de faire usage d'un aligneur lexical automatique afin d'obtenir un alignement mot-à-mot rapidement et de pouvoir extraire ces alignements pour les récupérer dans un dictionnaire bilingue. L'intérêt de procéder de la sorte, avec ce type d'outils, est d'obtenir un dictionnaire bilingue précis, unique et spécifique à un texte bilingue. Dans le cadre de ce stage, tous les tests ont été effectués sur le couple de langues français-anglais. Les résultats obtenus sont donc exclusifs à ce couple de langues. J'envisage cependant sans difficulté la possibilité d'appliquer le processus contenu dans ces scripts à des langues différentes. J'ai en effet procédé de manière à ne jamais prendre en compte la langue des textes pour traiter ces derniers avec mon programme.

Une fois le dictionnaire bilingue créé, l'objectif était principalement d'en faire usage directement, soit par le biais d'un affichage complet du dictionnaire dans un premier temps, soit par une recherche par mot clé, donnant accès rapidement aux correspondances

1 Un couple de textes qui sont rédigés dans deux langues différentes, et qui sont une traduction l'un de l'autre.

disponibles dans une langue cible pour un mot recherché dans une langue source. Nous voulions avant tout faire de ce dictionnaire un outil lisible et facile d'utilisation.

Partie 1

-

Présentation du domaine

Chapitre 1. Qu'est-ce que la TAO ?

La Traduction Assistée par Ordinateur (TAO) est un domaine du traitement automatique des langues, dont le champ d'action touche principalement la traduction. Selon une définition du CNRTL[2], la traduction est le « fait de transposer un texte d'une langue dans une autre ».

La traduction assistée par ordinateur comporte un ensemble d'outils informatiques qui permettent à un traducteur humain de réaliser des traductions avec plus de simplicité. Ces outils permettent d'alléger la charge de travail du traducteur par l'automatisation de certaines tâches répétitives, soit par exemple en proposant à nouveau des productions réalisées antérieurement sur une chaîne de caractères, grâce à la mémoire de traduction, soit en alignant deux textes au niveau phrastique pour mieux se repérer pendant une phase de relecture. On peut également compter parmi cet ensemble d'outils des logiciels centrés essentiellement sur l'aide à la gestion de projet. Pour dresser une liste non-exhaustive de certains des logiciels les plus utilisés sur le marché de la traduction, je m'appuierai sur une veille concurrentielle[3] que j'avais réalisée en mars 2020 dans le cadre d'un stage en entreprise chez ALTICA-Traductions.

1. Aide à la gestion de projet

Un logiciel de TAO se compose généralement d'une suite[4] d'outils divers s'intégrant à un éventail très large de possibilités. Comme je l'ai explicité ci-dessus, ces outils peuvent aussi bien comprendre un simple tableau comparatif à deux entrées, aidant à visualiser le texte d'origine face à sa traduction en cours, qu'un logiciel complexe de gestion de projet. La gestion de projet est un sous-domaine large dans le milieu de la traduction professionnelle, il sera donc difficile d'en tisser un canevas regroupant tous ses aspects de manière exhaustive dans ce mémoire. J'aborderai néanmoins les points les plus importants seront abordés, à commencer par les outils de suivi de projet.

1.1. Le suivi de projet

Le suivi de projet est un élément incontournable des logiciels d'aide à la gestion de projet dans lequel on distingue des outils récurrents, tels que la création automatique d'un planning lors de la création d'un nouveau projet ou la création automatique de devis et de

2 Centre national de ressources textuelles et linguistiques. (2020)
3 Une veille concurrentielle est l'équivalent d'un état de l'art visant à répertorier les concurrents dans le champs d'action d'une entreprise. Voir Annexe 1. p47
4 Voir glossaire p.43

factures. Parfois, dans une équipe de plusieurs traducteurs, une sélection et une assignation automatisée de certains traducteurs à certains projets spécifiques en fonction de leurs spécialités respectives peuvent s'effectuer. On retrouve cette option avec le logiciel *XTM-cloud*[5] par exemple, qui permet l'assignation automatique de certains linguistes à des tâches différentes (traduction ou relecture) sur un même projet.

1.2. Automatisation des tâches

Durant la création d'un projet, l'automatisation de certaines tâches peut également avoir lieu. À l'aide de *templates* de projets[6], lors de la création d'un projet similaire à un ancien déjà mené à terme, certaines informations relatives à un client contenues dans une base de données peuvent être complétée automatiquement. Certains logiciels sont de plus équipés d'outils permettant le calcul de divers éléments, comme celui du budget nécessaire à la réalisation du projet, les calculs de KPI[7], les retours sur investissement relatifs au projet en cours... Tous ces éléments peuvent être estimés automatiquement afin d'offrir un gain de temps non négligeable aux traducteurs.

Tous les outils de gestion de projets en traduction n'offrent donc pas les mêmes services. La suite *XTM*, par exemple, ne propose pas l'automatisation des devis et des factures, bien que d'autres suites de logiciels comme *SDL Trados*[8] ou *LBS*[9] *suite* le fassent. De ce fait, lorsqu'une entreprise recherche un logiciel de gestion de projet, il est très important qu'elle prenne son temps. En effet, pour que l'outil choisi soit le plus optimal possible et corresponde réellement aux besoins de l'entreprise, il est nécessaire d'analyser les différents composants des suites proposées, ainsi que leurs performances, leur maniabilité et leur prise en main. De plus, certains logiciels d'aide à la gestion de projets orientés vers la traduction n'incluent pas systématiquement d'outils de TAO axés sur la traduction elle-même. Certaines entreprises préfèrent ainsi utiliser deux suites de logiciels différentes. D'une part un logiciel pour la gestion de projet, consacré exclusivement à l'automatisation des tâches administratives, comme le logiciel *XTRF*[10] par exemple, spécialisé dans la gestion de projet de traductions, mais qui ne possède aucun outil d'aide à la traduction. D'autre part, on sélectionnera une deuxième suite logicielle, qui propose des outils d'assistance à la traduction, comme *LBS*.

5 https://xtm.cloud
6 Modèle de projet enregistré comme base pour remplir automatiquement une nouvelle fiche projet, contenant des informations diverses sur le type de projet ou le client par exemple.
(Source :https://www.projetex.com/translation-management-project-management/features)
7 Key Performance Indices, ou indicateurs clés de performance en français. « Les KPI sont utilisés pour déterminer les facteurs pris en compte pour mesurer l'efficacité globale d'un dispositif commercial ou marketing » (Source : https://www.definitions-marketing.com/definition/kpi/).
8 https://www.sdltrados.com/fr/products/business-manager/
9 https://www.sdltrados.com/fr/
10 https://xtrf.eu/

Nous avons donc pu étudier ensemble que les projets de traduction, tout comme des projets d'autres types, pouvaient être allégés au niveau du nombres de tâches répétitives à effectuer, au niveau administratif par exemple. Je vais maintenant présenter des outils de TAO parmi lesquels certains permettent à un traducteur de produire ses traductions plus facilement et rapidement.

2. Mémoire de traduction

Améliorer dans le même temps la qualité et la quantité de traductions produites par un traducteur humain, c'est l'enjeu de la traduction assistée par ordinateur. Loin de l'idée de créer des logiciels capables de fournir des traductions sans supervision humaine, il n'en reste pas moins intéressant de s'appuyer sur des propositions faites par la machine pour gagner du temps. C'est en cela que la mémoire de traduction trouve sa première utilité.

Le fonctionnement est similaire pour la plupart des logiciels de traduction faisant appel à la mémoire de traduction. Le texte dans la langue originale et sa traduction sont affichés dans deux fenêtres côtes à côtes. Par un système d'alignement automatique, chaque traduction d'un bloc de texte effectuée est enregistrée et stockée sur une base de données, pour être proposée plus tard comme solution pour traduire un bloc de texte semblable au précédent. Le texte se complète alors automatiquement dans la fenêtre de traduction. Cette traduction n'est qu'une proposition et reste modifiable à tout moment. En effet, selon le contexte, les goûts du traducteur ou la nécessité de varier les formulations, il est nécessaire d'avoir la possibilité de changer cette proposition de traduction à tout moment. Cette traduction suggérée est donc une aide pratique facilitant grandement la tâche d'écriture en automatisant certaines parties. En n'ayant pas à chercher une traduction adéquate pour chaque segment de texte, le traducteur peut alors mieux se concentrer sur la relecture et la correction de sa production. Cela a pour effet de grandement améliorer la qualité de la traduction ainsi rédigée. L'auteur aura de ce fait plus de temps à consacrer à la recherche de figures de styles s'il le désire.

En plus d'une amélioration de la qualité de la traduction, on pourra également noter une augmentation de la quantité de textes pouvant être produits par un seul traducteur. En effet, un traducteur seul requiert parfois l'aide d'un collègue pour l'aider, ne serait-ce que dans une phase de relecture, pour s'assurer que son texte est bien écrit, compréhensible et sans fautes (s'il travaille sans outil de TAO, il prend le risque de rater des erreurs difficilement détectables, isolé face à sa production écrite). Or l'utilisation de la mémoire de traduction pour proposer une solution quand c'est possible permet au traducteur d'endosser le rôle de relecteur, lui permettant une vigilance accrue face aux erreurs ou aux problèmes de cohérence

liés au contexte et un gain de temps lui permettant de passer plus rapidement d'un projet de traduction à l'autre. Sa productivité est accrue sans avoir à sacrifier la qualité de son travail.

La mémoire de traduction est un outil de traitement automatique complexe. Elle peut donc mélanger des outils de reconnaissance automatique de textes écrits, des appels à une base de données contenant des couples de traductions enregistrés, mais également des alignements phrastiques générés automatiquement. L'alignement automatique, qui contribue à faire des outils de TAO ce qu'ils sont en automatisant l'alignement de phrases ou de mots entre deux textes de langues différentes, notamment en association avec la mémoire de traduction, est ce qui va nous intéresser dans ce mémoire.

Chapitre 2. L'alignement automatique en TAO

Comme mentionné précédemment, l'alignement automatique est un outil utilisé par de nombreux logiciels de TAO. Derrière un concept simple, celui d'aligner des phrases ou des mots de deux textes qui sont une traduction l'un de l'autre dans des langues différentes, se cache une réflexion complexe et de nombreux algorithmes de calculs statistiques.

L'alignement au niveau syntagmatique ou lexical est rendu possible par un ensemble de calculs de probabilités pour deux textes qu'un mot ou une expression d'un premier texte soit aligné avec un ou une autre dans l'autre texte. Brown & al. proposaient un algorithme d'estimation de la probabilité d'un mot en français pour la traduction de tel ou tel mot en anglais (1988 ;1990, cité par Brown, Della Pietra, Della Pietra & Mercer, 1993).

1. Modèles génératifs : les modèles IBM

Les premières études menées sur l'alignement automatique de mots remontent à 1949. On envisage alors pour la première fois l'utilisation de statistiques pour traiter la traduction assistée par ordinateur (Weaver, 1955, cité par Brown, Della Pietra, Della Pietra & Mercer, 1993). L'initiative est abandonnée pendant quelques années, notamment à cause du manque de moyens technologiques de l'époque. Plus tard, on reprend cette proposition d'utiliser des modèles statistiques pour traiter les alignements de mots automatiquement. Brown, Della Pietra, Della Pietra & Mercer présentent les 5 premiers modèles statistiques génératifs développés par IBM (1993).

Selon ces auteurs, contrairement à la traduction mentale et humaine, la TAO ne se fait pas en comprenant le sens de ce qui doit être traduit, mais en faisant le tour de toutes les possibilités et en procédant par élimination. Les modèles 1 et 2 développés par IBM posent les bases de l'alignement par modèle génératif. On pose le problème sous la forme d'un calcul de probabilité utilisant le théorème de Bayes :

$$\Pr(\mathbf{e}|\mathbf{f}) = \frac{\Pr(\mathbf{e})\Pr(\mathbf{f}|\mathbf{e})}{\Pr(\mathbf{f})}.$$

Figure 1: Théorème de Bayes.

On représente la probabilité *Pr* d'avoir par exemple un mot *f* en français (langue cible) en connaissant *e*, le mot en anglais (langue source). De la sorte, trouver la traduction la plus probable se résume à maximiser les probabilité selon deux modèles complémentaire : un

modèle de traduction Pr(f/e) qui enregistre la probabilité des équivalences entre textes sources et cibles, et un modèle de langage Pr(e) qui enregistre la probabilité d'un énoncé dans la langue cible (sa correction syntaxique et idiomatique) indépendamment de la langue source.

Les modèles IBM reposent sur un principe : tous les alignements sont acceptables pour une paire de chaînes de caractères, mais certains ont une probabilité moindre par rapport à d'autres d'être les meilleures traductions possibles. Pour aller plus loin, il n'est pas possible de se reposer uniquement sur les mots (leur ressemblance dans les deux langues ne suffit pas toujours). Les concepts ne sont pas représentés directement par les mots, mais plutôt par un intermédiaire formé de mots. Brown, Della Pietra, Della Pietra & Mercer nomment cet intermédiaire « cept » (1993, p.266). Ces « cepts » peuvent compter des nombres de mots différents selon les langues. Ainsi, même s'il est possible de trouver un alignement où chaque mot en français correspond précisément à un mot de l'anglais, il est aussi tout à fait possible de trouver des cas où un mot français correspond à plusieurs mots anglais et réciproquement. Il est également envisageable que certains mots dans une langue ne correspondent à aucun mot dans l'autre langue.

Les modèles HMM (dont nous allons parler dans la partie suivante) ainsi que les modèles IBM 1 et 3 fonctionnent selon une contrainte : un mot dans la langue cible ne peut être aligné qu'à un seul mot dans la langue source. Les syntagmes plus complexes ne sont donc pas toujours alignés correctement puisqu'ils ne peuvent pas être traités comme un tout depuis la langue cible. Ce problème n'est plus présent dans la version 2 du modèle IBM qui prend les bases du premier modèle en rajoutant la possibilité de prendre en compte le fait qu'un mot dans la langue cible a pu être généré par plusieurs mots dans la langue source.

2. Autres modèles : modèles heuristiques et modèles HMM

Parmi tous les modèles d'aligneurs, il en existe d'autres sortes que les modèles statistiques basés sur le théorème de Bayes. Ce sont ceux dits de type heuristiques, qui fonctionnent différemment des modèles statistiques dans la mesure où ils tendent à procéder par approches successives du problème à résoudre, à savoir quel mot dans la langue cible correspond à un alignement optimal pour un mot de la langue source. En s'appuyant sur des similitudes avec des alignements précédemment effectués, ce modèle élimine progressivement les alternatives les moins logiques pour finir par garder celle qui est considérée comme optimale parmi tous les choix possibles. Moore a par exemple développé en 2005 un modèle heuristique s'appuyant sur un algorithme glouton, ou qu'il décrit ainsi : « a series of greedy searches »(Moore, 2007, p.84). Il s'inspire de l'algorithme de liens compétitifs de Melamed (2000, cité par Moore, 2007).

Une comparaison de différents modèles d'aligneurs effectuée par Och & Ney (2003) montre cependant que la qualité de l'alignement proposée par un modèle statistique reste supérieure à celle d'un aligneur à modèle heuristique.

Un autre type de modèle existant que nous avons mentionné plus tôt comprend les modèles HMM, ou modèles de Markov cachés. Il s'agit à nouveau de modèles statistiques mais cette fois-ci basés sur l'idée qu'il est possible d'effectuer des calculs de probabilité d'alignement en ignorant certains paramètres. On peut ainsi ignorer le nombre d'alignements possédés par un mot dans la langue source, et produire des alignements en fonction de la probabilité et non du nombre de mots dans la phrase.

Selon Och & Ney : « *The HMM makes use of locality in the source language, whereas Model 4 makes use of locality in the target language* »[11](2003, p.28). Ces auteurs se sont concentrés sur cette idée pour chercher à développer un nouveau modèle capable de lier ces deux types de dépendances, afin d'augmenter la qualité de l'alignement. Ils présentent ce modèle comme le modèle 6 d'IBM.

Le développement de nouveaux aligneurs se poursuit au même rythme que les recherches dans le domaine des statistiques et de la théorie de la communication, les chercheurs proposant ainsi des algorithmes pour des aligneurs toujours plus performants.

11 « Le modèle de Markov caché se sert de la place [du mot] dans la langue source, alors que le Modèle 4 se sert le la place [du mot] dans la langue cible ».(Notre traduction)

Partie 2

-

Démarche adoptée / travail réalisé

Chapitre 3. L'aligneur lexical

Pour ce projet, il m'a fallu sélectionner un aligneur lexical parmi un certain nombre d'aligneurs fonctionnant sur des bases différentes. J'ai décidé de tester deux d'entre eux : des aligneurs nécessitant un entraînement préalable sur un corpus de textes, et d'autres pouvant effectuer des alignements lexicaux sans que l'utilisateur n'ait à entraîner le programme en amont. Pour rappel, l'objectif principal de l'outil que je souhaitais créer était avant tout de pouvoir effectuer une extraction du lexique d'un texte bilingue à la suite d'un alignement lexical. Il était donc nécessaire que l'aligneur choisi soit aussi rapide que précis. J'ai testé un aligneur nécessitant un entraînement non-supervisé[12], *HMM-aligner*[13], et un autre dont l'entraînement n'est à faire par l'utilisateur, *fast_align*, pour voir lequel serait le plus efficace. J'ai finalement choisi *fast_align,* un aligneur basé sur le modèle IBM 2, qui était plus rapide et plus intuitif selon moi.

1. Présentation de fast_align

On peut obtenir *fast_align* directement depuis cette page *GitHub* : https://github.com/clab/fast_align.

Le logiciel *fast_align*, codé principalement dans le langage informatique *C++* et *Python*, fonctionne sur la base du modèle IBM 2, modèle statistique basé sur le théorème de Bayes vu dans le Chapitre 2. C'est un logiciel d'alignement à apprentissage non-supervisé, ce qui signifie que son entraînement se fait sur des données non-étiquetées[14], et peut donc s'entraîner à chaque nouveau texte sans intervention de la part de son utilisateur. Cela implique parfois une perte de précision dans certains genres littéraires s'il n'a jamais été exposé à de tels textes. En revanche, son utilisation est facilitée puisqu'il est possible de le lancer dès la première manipulation. Pour ce projet, dans un premier temps, il était nécessaire que je puisse travailler avec un aligneur dans de brefs délais afin de pouvoir développer rapidement mon programme. De plus, il me fallait un logiciel capable d'aligner des textes de genres variés sans préparation préalable.

12 En apprentissage machine, un entraînement supervisé se fait sur des données étiquetées, le non-supervisé sur des données non-étiquetées. (Voir notes de bas de page n°13, p.19).
13 GitHub de *HMM-Aligner* : https://github.com/sfu-natlang/HMM-Aligner
14 L'étiquetage de données peut par exemple dans le cas d'un texte comporter des indications sur le sens d'un mot que le programme peut analyser pour apprendre à reconnaître ce type de mot.

1.1. Fonctionnement

Pour commencer, il est nécessaire de posséder une machine qui utilise un système d'exploitation de la famille *Linux*, tel qu'*Ubuntu*, car *fast_align* n'est disponible que sur ce type de système. Il s'installe facilement en téléchargeant dans un premier temps le logiciel directement depuis la page *GitHub* de *fast_align*, puis en effectuant une courte série de commandes sur l'invite de commande d'*Ubuntu*, décrite dans le fichier *README.md* du dossier téléchargé.

On peut obtenir des résultats pour un alignement d'une première langue à une autre avec une simple ligne de commande[15].

Il est également possible d'avoir un alignement dans le sens inverse avec une commande incluant « -r » pour *reverse*. D'autres options sont également modifiables en fonctions des résultats recherchés, mais pour notre projet, la méthode de base alignant d'une langue vers l'autre nous suffit. Les résultats de l'alignement seront stockés dans le fichier *forward.align* enregistré dans le dossier *build* si le chemin d'accès du fichier de sortie n'a pas été modifié.

1.2. Résultats en sortie

Dans le fichier *forward.align*, on trouvera les résultats d'un alignement réalisé avec *fast_align* à partir d'une entrée qui suit la forme des deux exemples si dessous :

> Je suis le plus grand. ||| I am the tallest.
> Il aime le café. ||| He likes coffee.

Chaque ligne contient un alignement. Deux phrases alignées, une dans la langue source, ici le français, l'autre sa traduction dans la langue cible, l'anglais, doivent être placées sur la même ligne et séparées par trois barres verticales « ||| » qui seront reconnues par *fast_align* comme un séparateur entre chaque langue. Par ailleurs, il est important que les tokens à aligner (mots et ponctuation) soient séparés par un espace.

Le programme de *fast_align* va réaliser un alignement dont le format indique les correspondances entre numéro de tokens. Autrement dit, il ne transforme pas le texte de base, mais enregistre l'alignement proposé dans un fichier séparé avec un code précis. Les résultats seraient les suivants pour notre exemple ci-dessus :

15 Voir Annexe 2, p.48

0-0 1-1 2-2 3-3 4-3
0-0 1-1 2-2 3-2

Chaque ligne du fichier d'entrée correspond à une ligne dans le fichier de sortie. Pour un ensemble de nombres *x*, les couples « x_1-x_2 » correspondent à la place des mots dans les phrases des deux langues. Le premier nombre du couple correspond à la place d'un mot en français. Le deuxième nombre, séparé du premier par un tiret « - », correspond à la place occupée par le mot anglais, qui est une traduction du mot en français. Ainsi, l'alignement « 4-3 » se lit de cette manière : le mot d'indice 4 dans la langue source, « grand », est aligné avec le mot d'indice 3, « *tallest* », dans la langue cible.

On a également un autre couple, « 3-3 », faisant intervenir le mot d'indice 3 de la phrase anglaise. Cela signifie donc que « *tallest* » est aligné à la fois au mot « plus » et au mot « grand ». En effet, il est possible d'avoir plusieurs mots d'une langue alignés à un seul dans une autre.

2. Phase de tests

Lors de l'utilisation d'un logiciel extérieur, il est intéressant le tester avant de l'intégrer à son propre programme, premièrement pour s'assurer qu'il fonctionne correctement, et ensuite afin de connaître son fonctionnement, ses limites et ses capacités.

2.1. Méthodologie

Pour la phase de tests, le processus se déroule en sélectionnant tout d'abord dans un corpus bilingue quelques lignes alignées au niveau phrastique. L'objectif est de tester ces lignes dans deux contextes différents : d'un côté les lignes seules, donnant un corpus de test de petite taille, puis l'intégralité du corpus dont est extrait le petit corpus, offrant ainsi un contexte plus important à ces lignes testées. Je compare ensuite les alignements des deux blocs de lignes similaires des deux corpus, afin de repérer lequel des deux, s'il y a une différence, est le mieux aligné.

Après avoir essayé avec un corpus[16] de 343 339 lignes, il s'est avéré nécessaire de le raccourcir. En effet, *fast_align* n'a pas supporté un fichier d'une si grande envergure sur ma machine, probablement pour une raison liée à la mémoire vive de mon ordinateur. Le corpus devait donc être réduit à moins de 50 000 lignes, ce qui offre néanmoins déjà un écart suffisamment élevé entre les deux textes à tester. Dans ce nouveau corpus d'exactement

16 J'ai utilisé pour ce test le corpus *GlobalVoices* de textes français- anglais alignés phrastiquement. Source : http://opus.nlpl.eu/GlobalVoices.php

45786 lignes, soit 147 865 mots, je sélectionne 200 lignes exactement. J'ai choisi les lignes 30 000 à 30 199 du corpus de base pour les 200 lignes à tester. On lance alors le logiciel sur les deux textes, celui de 200 lignes et le corpus complet dont le premier est extrait. Le résultat de l'alignement du corpus intégral est sauvegardé dans un fichier nommé « forward-long.align », tandis que l'alignement de l'extrait est enregistré dans un fichier nommé « forward-reduit.align ».

L'observation des résultats de *fast_align* sur ces 200 lignes se fait en comparant les alignement lexicaux du corpus intégral et ceux du corpus réduit, en se concentrant seulement sur les 200 lignes identiques des 2 corpus. On effectuera donc une évaluation des deux résultats pour voir si la taille du corpus influe sur le résultat final pour un même groupe de lignes.

2.2. Évaluation

L'évaluation des résultats donne une indication sur la meilleure façon d'utiliser l'aligneur lexical *fast_align*. Pour vérifier rapidement si le traitement est différent entre les deux corpus, la méthode retenue a été d'observer leur distance à l'aide de l'algorithme de Levenshtein[17] pour *Python* (source : https://www.datacorner.fr/comparer-chaines/).

En lançant le script *levenshtein.py* (voir Figure 2), créé uniquement pour observer l'éloignement des deux résultats, on peut voir rapidement s'ils diffèrent. Plus la distance entre les deux fichiers est grande, plus les deux alignements[18] trouvés en résultats divergent. Si le ratio est de 1, cela signifie que les deux résultats sont identiques, et donc que la taille du corpus ne change rien au fonctionnement de *fast_align*. Plus le ratio se rapproche de 0, plus il y a de divergences entre les deux fichiers. Il suffit donc d'obtenir un ratio différent de 1 pour être certain qu'au moins une différence entre les deux alignements existe. Dans ce cas il faut alors regarder et vérifier manuellement les phrases des deux corpus comprises dans les 200 lignes testées.

17 Cet algorithme effectue un calcul de la distance entre deux chaînes de caractères. Source : https://fr.wikipedia.org/wiki/Distance_de_Levenshtein .

18 Pour rappel : les alignements entre deux mots faits par fast_align sont des paires de nombres. Si les alignements sont différents, on l'observe dans les suites de chiffres.

```python
# coding: utf-8
import Levenshtein as lev

def levCalclulate(str1, str2,n1,n2):
    Distance = lev.distance(str1, str2)
    Ratio = lev.ratio(str1, str2)
    print("Levenshtein entre {0} et {1} :".format(n1, n2))
    print("> Distance: {0}\n> Ratio: {1}\n".format(Distance, Ratio))

nom1="forward-reduit.align"
nom2="forward-long.align"
al1=open(nom1, "r", encoding="utf8")
al2=open(nom2, "r", encoding="utf8")
align1=""
align2=""
for line in al1:
    line=line.strip()
    align1+=line+"\n"
for line in al2:
    line=line.strip()
    align2+=line+"\n"


levCalclulate(align1, align2, nom1, nom2)

```

Figure 2. Script Levenshtein.py.

Le résultat de cette évaluation, pour rappel, n'influe pas sur le choix de garder ou non cet aligneur, mais nous permet d'obtenir l'information selon laquelle la longueur du corpus influe bel et bien sur les résultats obtenus par *fast_align* quant à l'alignement lexical des phrases.

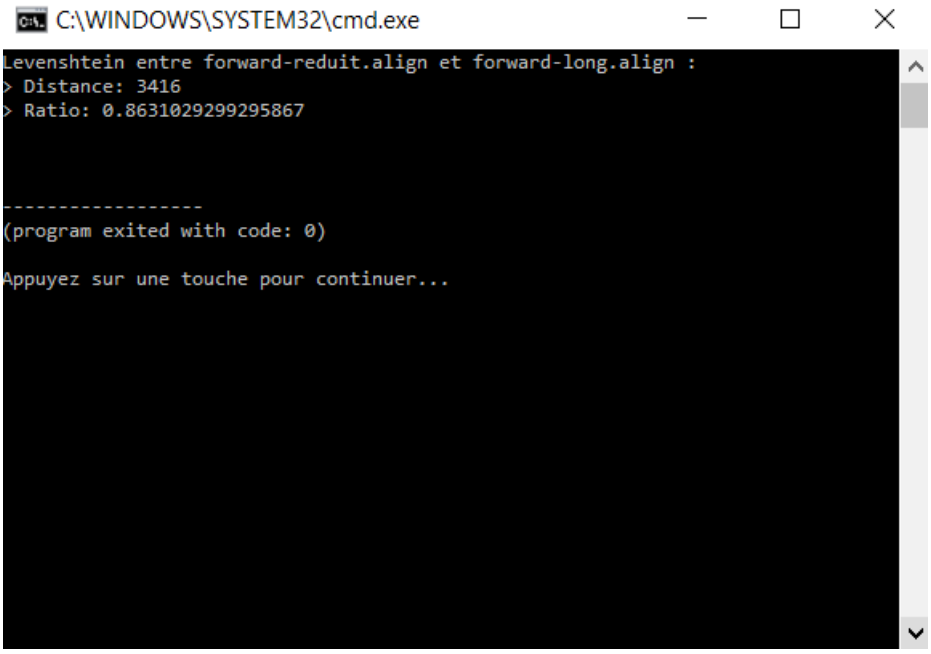


Figure 3. Résultats du script Levenshtein.py.

Après lancement du script, il apparaît qu'il y a bel et bien une différence entre les deux alignements (voir Figure 3). Après avoir observé les deux alignements proposés, celui réalisé sur un corpus long, et donc possédant plus de contexte que le corpus réduit, est bien mieux

réalisé et comporte moins d'erreurs d'alignement que pour le corpus réduit. Par exemple, sur ces 200 lignes, pour le mot « pays » en français, à partir du corpus long on aura 4 alignements : « *country* » deux fois, « *sector* » et « *countries* ». Ces alignements semblent corrects. En revanche, pour le même mot depuis le corpus réduit, on retrouve seulement 3 alignements : à nouveaux deux alignements avec « *country* », mais cette fois les deux autres alignements n'ont pas été faits, et à la place on en a un autre avec « *over* ». Il faut donc noter que des résultats de qualité moindre obtenus sur un corpus de textes bilingues peuvent éventuellement découler de la taille trop réduite de ce dernier. Pour mieux visualiser les décalages pouvant être observés Pour mieux visualiser les décalages pouvant être observés entre les deux alignements, voici deux exemples de phrases qui ont été alignées différemment selon que la longueur du texte duquel elles sont tirées :

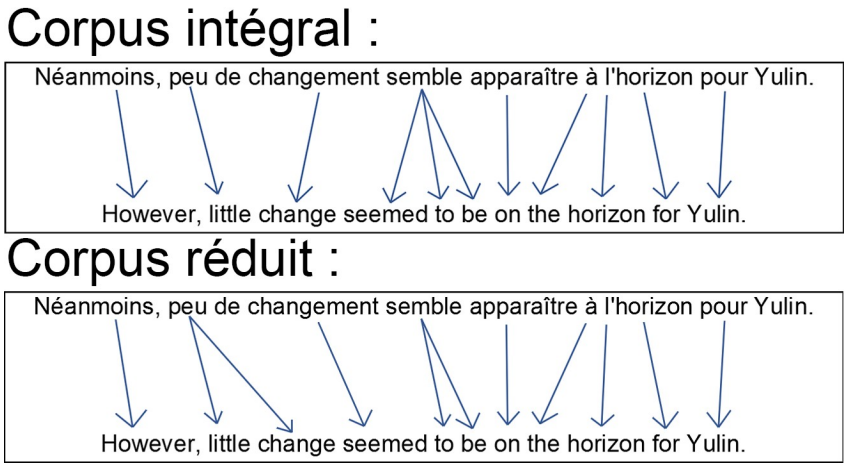


Figure 4: Ligne 198.

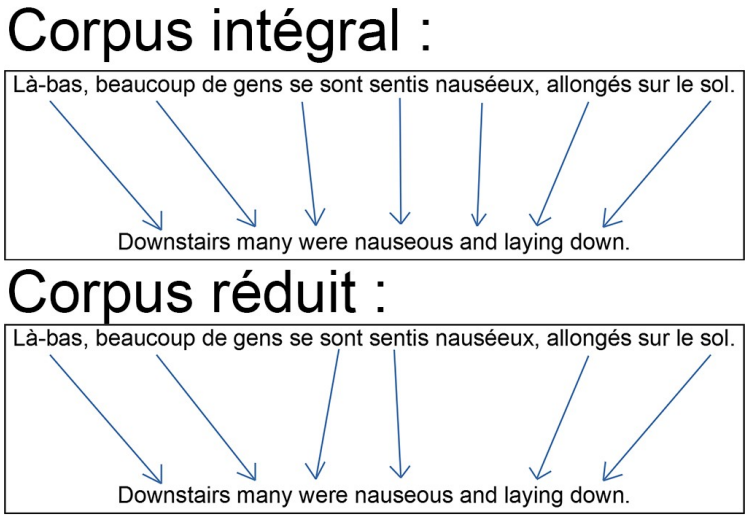


Figure 5: Ligne 64.

On observe des erreurs liées à une mauvaise reconnaissance des mots à aligner dans ces exemples dans le cas du corpus réduit qu'on ne retrouve pas pour le corpus intégral. Par exemple, sur la Figure 4, on observe que pour le corpus réduit « changement » est aligné avec « *seemed* », alors qu'il serait plus logique que ce mot soit aligné avec « *change* » comme c'est le cas avec le corpus intégral.

Cette évaluation n'influera pas sur ma décision d'intégrer *fast_align* à l'outil *E-Lexic* dans sa première version. Néanmoins, il est possible que ce choix soit altéré pour une prochaine version, en fonction des besoins à traiter dans les améliorations à apporter[19].

19 Voir Chapitre 6. Améliorations possibles, p.36.

Chapitre 4. Programmation d'un logiciel de récupération des alignements de mots

Les codes python créés dans le cadre de ce projet sont tous disponible sur cette page GitHub : https://github.com/Florent-Marie/E-Lexic.

L'objectif de mon outil était d'obtenir un visuel clair et lisible des résultats obtenus grâce à un aligneur lexical sur des textes bilingues alignés au niveau phrastique, et d'obtenir un aperçu des possibilités de traductions de chaque mot grâce à une méthode de recherche par mots clés dans un dictionnaire.

1. Fonctionnement

Le principe de mon programme est le suivant : l'utilisateur démarre avec deux fichiers, un texte en français et sa traduction en anglais. Tous deux doivent être alignés au niveau phrastique. Ces deux fichiers passent par une chaîne de traitement qui va rendre des résultats dans un fichier de sortie unique. Voyons plus en détail comment se déroule chaque opération de cette chaîne de traitement.

1.1. Formatage pour fast_align

Cette étape est nécessaire si le corpus utilisé n'est pas déjà formé de la bonne manière pour *fast_align*, à savoir, si ce corpus est contenu dans un seul fichier de texte avec la bonne mise en forme. Pour rappel, deux phrases alignées sont sur la même ligne et séparées par trois barres verticales « ||| ». Chaque couple de phrases est quant à lui séparé des autres par un retour à la ligne dans le texte. Si ce n'est pas le cas et que vous avez deux fichiers texte, il est nécessaire de suivre les étapes ci-après pour obtenir un fichier acceptable par *fast_align*.

Dans un premier temps, il faut s'assurer que le texte contenu dans chaque fichier est correctement formaté. Pour cela, un premier programme nommé *fast-align-format.py*, que j'ai codé dans le langage *Python*, permet de fusionner les deux textes du départ afin de n'avoir en sortie qu'un seul fichier contenant chaque couple bilingue de phrases alignées. La manière de lancer ce script est détaillée en annexe de ce mémoire (voir Annexe 6).

Au moyen d'une fonction nommée « *propre()* », on s'assure que le texte ne contient aucun caractère spécial qui ne serait pas reconnu par *fast_align*. Cette fonction effectue une série de remplacements d'éléments de ponctuations qui existent sous différentes formes, telles

que les apostrophes et les guillemets[20]. De plus, grâce à des expressions régulières[21], elle supprime les éléments de ponctuation isolés[22] qui sont parfois comptés comme des mots par *fast_align* quand ils sont mal reconnus par le logiciel.

Dans les cas où les textes utilisés sont relativement bien formés dès le départ, cette fonction ne sera utilisée que préventivement pour s'assurer du bon format du texte.

Le programme *fast-align-format.py* est indépendant du reste de la chaîne de traitement que j'ai créée afin d'éviter à l'utilisateur de s'en servir si ses textes bilingues sont déjà formés et alignés au niveau phrastique dans un seul fichier comme le requiert *fast_align*. La suite se trouve dans le script de *traitement_aligner.py*.

1.2. Traitement de l'alignement de fast_align

L'étape suivante, après s'être assuré que les phrases du corpus bilingue sont correctement alignées dans un seul fichier, est de procéder à l'alignement au niveau phrastique de chaque phrase. C'est à ce moment là qu'on pourra utiliser le logiciel *fast_align*. Une fois ce traitement fait, on récupère le fichier de texte utilisé en entrée ainsi que le résultat de l'alignement produit par *fast_align* en sortie. Ces deux fichiers seront ensuite utilisés en entrée du deuxième programme que j'ai conçu, nommé *traitement_aligner.py*. La marche à suivre pour lancer ce programme se trouve en annexe de ce document (voir Annexe 7).

Tout d'abord on ouvre les deux fichiers en mode lecture puis on enregistre le contenu de chaque fichier dans des variables. À partir de la variable contenant le texte bilingue, on répartit chaque paire de phrases alignées du corpus bilingue séparées par le symbole « ||| » dans deux listes dont chaque élément est une chaîne de caractères correspondant à une phrase. Un liste contient les phrases en français et l'autre celles en anglais.

Ensuite, pour chaque phrase, on fait défiler les mots pour les enregistrer dans un dictionnaire en face du mot qui est aligné avec celui-ci. Les paires de nombres présents dans le fichier de sortie de *fast_align* nous permettent de savoir quel(s) mot(s) en langue source correspond(ent) à quel(s) mot(s) en langue cible. Chaque nombre correspond à l'indice d'un mot dans la phrase. On s'en sert donc pour ranger les mots dans un dictionnaire. Le premier indice sert à trouver le mot qui servira de clé du dictionnaire, le deuxième indice correspond au mot qui devient alors la valeur de cette clé correspondante. Pour chaque itération d'un mot en clé, on peut soit en ajouter un nouveau en valeur accompagné d'un compteur initialisé à 1,

20 Il existe différents styles d'apostrophes dont les codages Unicode sont différents, comme par exemple « ' » et « ’ ». Source : https://fr.wikipedia.org/wiki/Apostrophe_(typographie) (Septembre 2020)
21 Les expressions régulières sont des chaînes de caractères qui, rédigées dans un code spécifique, correspondent plusieurs chaînes de caractères lisibles dans un langage formel.
22 La plupart du temps, ce sont des erreurs d'espacement ou des ponctuations accessoires telles que les guillemets autour d'une citation.

soit ajouter un point au compteur d'un mot traduit déjà présent. On aura ainsi pour chaque clé une valeur dont le type sera un tuple[23].

Les éléments de ce tuple sont les différents mots trouvés dans le texte aligné au mot en clé du dictionnaire. Ce tuple est lui-même composé de tuples de deux éléments : le mot, et le nombre d'alignement que ce mot partage avec le mot en clé dans ce texte. Ce nombre fonctionne comme un compteur durant le traitement. Il est instancié à 1 à la création du tuple, et on lui ajoute 1 à chaque nouvelle occurrence du mot valeur dans un alignement avec le mot en clé. Voici une représentation de ce à quoi ressemble la structure interne d'un dictionnaire pour le mot « chat » en français :

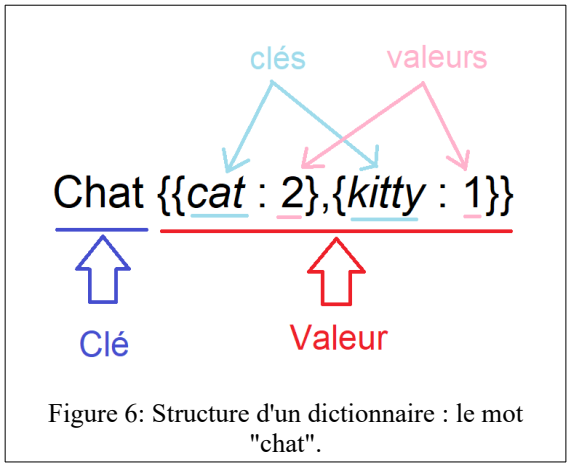


Figure 6: Structure d'un dictionnaire : le mot "chat".

Une fois le traitement terminé, il est désormais possible d'accéder au dictionnaire. L'application est pour l'instant dépourvue d'une interface utilisateur, néanmoins la recherche de mots clés est déjà possible.

2. *Structure interne du programme*

Comme énoncé précédemment, ce programme est une chaîne de traitements complexe, composée de différentes fonctions et différentes étapes de traitement qui seront explorées plus en profondeur dans les points suivants.

2.1. fast-align-format.py

Ce programme, qui a pour but de prendre deux textes qui sont des traductions l'un de l'autre, alignés au niveau phrastique, et de les fusionner en un seul nouveau fichier de texte au format accepté par *fast_align*, comporte différentes étapes de traitement.

La première étape est la lecture des deux fichiers textes à fusionner. Les chemins d'accès à ceux-ci sont donnés en arguments au moment du lancement de ce programme. Un

23 Il s'agit d'une variable contenant plusieurs éléments.

quatrième argument facultatif est le chemin d'accès du fichier qu'on obtient en sortie. Si cet argument n'est pas donné, le nom par défaut du fichier de sortie est *Corpus_fast-align.txt*. Le programme lit les deux fichiers de textes et les transfère dans deux tableaux : *srctab*, dont chaque élément est une chaîne de caractères correspondant à une phrase en français, et *tgttab* qui contiendra les phrases en anglais. Pour ce faire, il fait appel à la fonction *tab()* qui récupère le contenu d'un fichier dans un tableau dont les éléments sont les lignes du fichier lu.

Ensuite, comme précédemment énoncé, on passe à la phase de mise en forme des deux textes adaptée à *fast_align* des deux textes. Pour cela, on crée une boucle qui va passer en revue chaque élément des tableaux *srctab* et *tgttab*, qui sont donc les lignes des textes, ce qui permettra d'effectuer la mise en forme sur chaque phrase indépendamment.

```
def propre(s):
    """
    Fonction qui épure le texte de caractères de ponctuations non-conformes,
    comme les apostrophes différentes et les signes de ponctuation gênants
    qui seraient encore présents dans le texte.
    """
    regex= re.compile(r"\s(\W|·)\s")
    new=regex.sub(" ",s)
    new=new.replace("’","'")
    new=new.replace("’ ","'")
    new=new.replace("' ","'")
    new=new.replace("’ ","'")
    new=new.replace("’","'")

    new=new.replace("“","\"")
    new=new.replace("”","\"")
    regex= re.compile(r"^\W+",re.M)
    new=regex.sub("",new)
    return new
```

Figure 7. Fonction propre() du fichier fast-align-format.py.

Il s'agit dans un premier temps de "nettoyer" le texte de toute impureté grâce à la fonction *propre()* (voir Figure 7), qu'on va appliquer à chacune des phrases contenues dans les tableaux *srctab* et *tgttab*. Cette fonction effectue une série de substitutions et de remplacements dans le texte, en partie à l'aide d'expressions régulières, notamment au niveau des marques de ponctuations. On en trouve certaines mal placées dans certains textes, telles que des virgules non-collées à la fin des mots comme cela devrait être le cas. On met également en place une uniformisation de la ponctuation. Cela concernera davantage les apostrophes ou les guillemets, qui existent sous diverses formes, et qui peuvent parfois apparaître sous différentes formes au sein du même texte.

Les phrases "nettoyées" sont alors directement fusionnées dans la chaîne de caractères temporaire *ligne*. Cette dernière subit à nouveau des changements, comme par exemple le remplacement de deux espaces par un seul afin de ne pas avoir de problèmes de mots vides

par la suite. Chaque itération de la boucle se termine par l'ajout du contenu de *ligne* à la variable *corpus*, qui accumule les lignes fusionnées.

C'est le contenu de cette variable *corpus* qu'on va inscrire dans un fichier texte qui est nommé par défaut *Corpus_fast-align.txt* qui sera le fichier d'entrée de *fast_align* possédant le bon format pour être aligné.

2.2. traitement_aligner.py

De même que pour le premier programme *fast-align-format.py*, destiné à donner le bon format au texte pour l'aligneur lexical, ce programme est composé de plusieurs étapes de traitement, faisant appel à différentes fonctions.

Tout d'abord, étudions la partie menant à la création des dictionnaires spécifiques. Le programme commence simplement par ouvrir les fichiers que l'utilisateur a donné en arguments au moment de l'appel au programme. Le premier argument est le fichier contenant le texte bilingue au format de *fast_align*, le second est le fichier contenant les alignements codés sous formes de couples de nombres.

Une fois les fichiers lus par le programme, le premier à être sauvegardé dans une variable sera celui du fichier de texte. On enregistre chaque ligne du fichier dans un tableau nommé *lineTab* en faisant appel à la fonction *tab()* (voir Figure 8) qui récupère le contenu d'un fichier dans un tableau dont les éléments sont les lignes du fichier lu.

```
def tab(fichier):
    """
    Fonction qui permet après lecture d'un fichier d'ajouter chacune des
    lignes de ce fichier comme élément d'une liste.
    """
    table=[]
    for line in fichier:
        line=line.strip()
        table.append(line)
    return table
```

Figure 8. Fonction tab() du fichier traitement-aligner.py.

On sépare ensuite chaque couple de phrases dans deux tableaux, l'un contenant les phrases en français, *srctab*, l'autre les phrases en anglais, *tgttab*. Ce sont ces deux tableaux qui vont êtres parcourus mot par mot en fonction des indices donnés par l'aligneur. Ces indices sont récupérés dans un tableau *lineAl*, à nouveau grâce à la fonction *tab()*. Les tableaux *lineAl*, *srctab* et *tgttab* sont censé tous les trois de taille identique.

À ce stade,les phrases alignées sont séparées, il ne reste plus qu'à les comparer les unes aux autre avec l'alignement contenu dans *lineAL*. On fait alors une boucle pour traiter chaque ligne des tableaux dans l'ordre. Chaque mot de chaque phrase est alors aligné selon le

```
def coordonnees(phrase):
    """
    Fonction qui repère les coordonées type latitudes/longitudes et
    remplace l'espace qui les sépare par [SPACE]
    """
    phrase=re.sub(r"(\(?(-?\d+\d*([$€£%'°]))+)(\s)((\d+\d*([$€£%'°]))+\))",r"\g<1>[SPACE]\g<5>",phrase)
    return phrase
```

Figure 9. Fonction coordonees() du fichier traitement-aligner.py.

traitement de *fast_align*. Grâce à des expressions régulières, on s'assure de ne récupérer qu'un mot à la fois, en excluant les ponctuations comme les virgules, les points ou les guillemets qui sont collés aux mots. Une exception ayant été décelée concernant les coordonnées géographiques pour lesquelles parenthèses et points doivent être collés aux nombres, ces éléments ne peuvent pas être retirés dans ce cas. On applique alors également la fonction *coordonees()* au mot actuel (voir Figure 9). Il n'est pas exclu que d'autres exceptions existent, mais elles seront traitées dans une autre version du programme, ce problème n'étant pas bloquant à ce niveau de la conception.

Une fois l'alignement réalisé, on enregistre le résultat sous forme de dictionnaire dont la clé est le mot *m* étudié, et sa valeur un autre dictionnaire dont les clés sont les mots *w* qui ont été trouvés comme étant alignés avec le mot *m*, et dont les valeurs sont leur nombre d'occurrences en tant qu'alignement avec le mot *m*. De cette manière, on est sûrs qu'un mot *m* est une clé unique, et que chaque mot *w* auquel il a été aligné est enregistré en valeur de cette clé.

On fait la même chose dans le sens inverse pour l'autre langue : chaque mot *w* est une clé d'un dictionnaire, avec pour valeur un dictionnaire qui a pour couple « clé / valeur » un couple mot *m* / nombre d'occurrences.

C'est ainsi que nous obtenons deux dictionnaires, un pour chacune des langues, et dans lesquels on peut naviguer grâce au système de recherche par mot clé.

3. Résultats et recherche par mots-clés

Il est possible d'afficher les résultats de différentes manières. L'utilisateur du programme peut choisir de consulter l'intégralité des dictionnaires générés par le programme (voir Annexe 3, p.49). Cela a pour effet d'afficher chaque mot ainsi que toutes ses traductions trouvées par alignement grâce à *fast_align*. Cette solution offre la possibilité d'avoir une vue d'ensemble des résultats. Il est également possible d'opter pour une recherche par mots clés d'un mot afin d'afficher chacune des traductions possibles de ce mot dans le texte bilingue (voir Annexe 4, p.50). La dernière option est celle de pouvoir obtenir un unique mot, le plus utilisé pour traduire un certain mot clé (voir Annexe 5, p.50). En effectuant une recherche d'un mot contenu dans les clés de l'un des dictionnaires, on va pouvoir voir tous les mots qui correspondent à une traduction du mot recherché selon l'aligneur, ainsi que son nombre d'apparitions.

La recherche par mot clé avec obtention d'un seul mot le plus utilisé dans le texte comme traduction du mot recherché est rendue possible grâce à la fonction *best()* qui va

chercher au sein du dictionnaire contenu en valeur du mot recherché, le mot clé qui possédera la valeur la plus haute parmi les nombres d'occurrences de chaque mot (voir Figure 10).

```
def best(h, clé):
    """
    Fonction récupérant le mot le plus utilisé pour traduire le mot clé
    recherché en inversant les couples mot/nombre d'occurences parmi les
    valeurs du dictionnaire
    """
    invMax={}
    for mot in h[clé]:
        invMax[h[clé][mot]]=mot
    return(invMax[max(invMax.keys())])
```

Figure 10. Fonction best() du fichier traitement-aligner.py.

Partie 3

-

Applications et perspectives

Chapitre 5. Applications de E-Lexic

La réalisation de ce projet dans le cadre d'un stage de fin d'étude ne m'a pas empêché de réfléchir à plus long terme sur l'utilité d'un programme tel qu'*E-Lexic* dans le monde de la traduction assistée par ordinateur. Dans une première version encore modeste, cet outil sait déjà répondre à une demande : extraire un lexique spécifique et le conserver dans un dictionnaire bilingue afin de réutiliser ces résultats. Il existe des domaines, notamment en TAO, qui pourraient bénéficier de l'application d'un outil tel que celui-ci.

1. Utilisation en mémoire de traduction

Ce programme a pour objectif actuel l'obtention d'un dictionnaire bilingue spécifique à un texte bilingue proposé en entrée. De nombreuses utilisations d'un tel programme peuvent être intéressantes, notamment dans le domaine de la traduction. En TAO par exemple, j'ai énoncé précédemment de concept de mémoire de traduction. L'extracteur *E-Lexic* pourrait devenir un outil exploitable par des logiciels de TAO offrant des mémoires de traduction. La force de ce programme étant, en effet, de créer des dictionnaires spécifiques pour des textes bilingues, la mémoire de traduction pourrait, avec son aide, devenir beaucoup plus importante dans le travail de traduction. En effet, elle ne se contenterait pas seulement de retenir des blocs de textes entiers, mais également des mots, des syntagmes, afin de proposer des traductions plus précises et contenant possiblement des variations de styles, via la comparaison des différentes possibilités de traductions contenues dans un dictionnaire pour en sélectionner la plus adaptée selon un contexte donné.

2. Application à des outils de traduction en ligne

Il existe en ligne des sites internet proposant des services de traduction automatique. Ces derniers proposent des fonctionnalités telles que la mise en contexte des mots recherchés et de leurs traductions dans des corpus de textes bilingues. Parmi les sites offrant de telles fonctionnalités on retrouve par exemple le site *linguee.com*. Ce site affiche des alignements phrastiques mais aussi lexicaux, sous la forme de sélection grisée légèrement dégradée. Il est facilement imaginable que l'utilisation d'un extracteur de lexique tel qu'*E-Lexic* pourrait aider à améliorer l'outil utilisé pour cette fonction de moteur de recherche dans des corpus bilingues, en permettant par exemple une vérification des alignements des phrases, afin de limiter l'apparition de phrases qui n'ont pas de rapport avec les mots recherchés, ce qui arrive parfois.

On pourrait également, par une recherche inversée dans un dictionnaire bilingue, proposer des phrases en contexte qui auraient comme mot dans la langue source un synonyme de celui recherché en mot clé. Par exemple, pour un utilisateur recherchant le mot « chat », utiliser ce type de programme pourrait permettre de proposer automatiquement à l'utilisateur d'effectuer une recherche sur le mot « félin ». En effet, une recherche par mot clé du mot « chat » dans certains dictionnaires donnerait le mot « cat » en anglais comme étant le mot le plus trouvé dans des alignements avec « chat ». En effectuant une recherche du mot « cat » dans un dictionnaire français-anglais cependant, un programme du type E-lexic pourrait mettre en avant que le mot le plus utilisé pour traduire « cat » était « félin » en français, et donc le proposer. En effet, les équivalences traductionnelles permettent de trouver des synonymes. Des réseaux sémantiques tels que *WOLF*[24] ont utilisé des corpus parallèles alignés pour effectuer ce genre d'aller-retour.

L'utilisation de ce logiciel en soutien à des applications déjà existantes pourrait donc les améliorer grandement, et permettre à leurs utilisateurs d'avoir un outil toujours plus complet et en accord avec leurs besoins. Bien entendu, la version actuelle d'*E-Lexic* ne suffira sûrement pas pour réussir à s'imposer dans le marché de la traduction. Il me faut alors réfléchir aux améliorations à apporter à ce programme.

En revanche, une idée originale d'application qui n'existe pas à l'heure actuelle serait celle d'un site web permettant d'aligner ses propres textes à la demande, pour en extraire à la fois un lexique bilingue et des concordances. *E-Lexic* est précisément capable d'effectuer une telle extraction, et pourrait être la première pierre de cet édifice. Un utilisateur possédant deux textes alignés au niveau phrastique pourrait les importer directement sur ce site web et obtenir rapidement l'affichage actuellement proposé par *E-Lexic*, puis dans un second temps récupérer ces résultats ainsi que les concordances des mots de ce lexique dans un fichier qu'il pourra télécharger.

24 Wordnet Libre du Français. Source : http://pauillac.inria.fr/~sagot/index.html#wolf

Chapitre 6. Améliorations possibles

E-Lexic est un logiciel qui est en phase d'optimisation pour le moment. Le principe de cet outil restera néanmoins toujours le même, un extracteur lexical agissant sur des alignements de mots dans des textes bilingues. Les changements qui pourraient s'opérer à l'avenir concerneront principalement les éléments périphériques à son fonctionnement, comme l'optimisation du traitement du résultat produit par l'aligneur lexical.

1. Optimisation du traitement de l'alignement

Le traitement d'un alignement lexical par mon programme fonctionne actuellement de manière optimale sur des alignements réalisés avec *fast_align*. Cependant, après quelques modifications au niveau de la lecture des fichiers d'entrée, il sera envisageable de l'adapter à n'importe quel aligneur lexical. Comme l'ai présenté, le logiciel *fast_align* est un aligneur efficace et rapide. Néanmoins, il existe ou existera peut-être un jour un autre aligneur lexical plus puissant, plus rapide et plus précis que ce dernier. Il serait alors bon de faire une étude comparative plus élaborée afin de sélectionner l'aligneur le plus à même de remplir la fonction requise par le système *E-Lexic*, à savoir un alignement lexical rapide, permettant par la même occasion à notre programme une extraction de lexique tout aussi rapide. Actuellement, *fast_align* commet encore quelques erreurs d'alignement, particulièrement sur des textes courts, ce qui lui offre moins de contexte. On observe parfois des décalages d'un mot sur toute une phrase, alignant des noms en français avec les épithètes de leurs traductions anglaises au lieu du nom en anglais lui-même par exemple.

Toujours à propos de *fast_align*, il faudrait trouver un moyen d'automatiser son lancement depuis le code *Python* afin d'éviter des manipulations supplémentaires. L'accent a pour le moment été mis sur le développement de l'extracteur lexical à partir de l'alignement, mais un de mes futurs objectifs serait d'intégrer l'aligneur directement à *E-Lexic*, afin de pouvoir ainsi lancer le programme avec pour seul fichier d'entrée le fichier texte contenant le corpus bilingue à traiter.

2. Traitement des mots grammaticaux et de syntagmes

Dans sa version actuelle, *E-Lexic* ne distingue pas les mots lexicaux des mots grammaticaux[25]. Il serait intéressant de pouvoir séparer ces deux types de mots afin d'éviter

25 Les mots lexicaux, ou mots pleins, sont des mots possédant un poids sémantique, à l'inverse des mots grammaticaux qui eux sont simplement nécessaire à la construction de la phrase d'un point de vue syntaxique.

d'être mis en difficulté par le bruit[26] généré par des articles quand on veut extraire les traductions d'un nom. On retrouve de nombreuses occurrences de ce genre, comme par exemple dans la phrase « Il aime le café. ». L'article défini « le » en français sera aligné avec le nom « coffee » dans la phrase anglaise, au même titre que le nom « café » en français. L'alignement qui nous intéresse dans ce cas est bien « coffee:café », mais on va obtenir au même titre l'alignement « coffee:le » qui n'est pas nécessairement intéressant. La présence de ce bruit est due au fait que parfois l'utilisation de déterminants sera différent d'une langue à l'autre : parfois en français on aura un déterminant, là où en anglais on en mettra pas. Pour pallier à ce problème, il faudrait trier les résultats en sortie. On pourrait par exemple filtrer les mots en fonction de leur partie du discours, en enlevant les mots grammaticaux alignés à des mots lexicaux par exemple. De plus, imaginer un filtrage en fonction de la fréquence des correspondances entre les mots serait intéressant dans la mesure où, dans bien des cas, le bruit correspond à des correspondances de basse fréquence. C'est le cas lorsque la correspondance détectée est due à une erreur d'alignement.

Parallèlement à ce problème, il serait intéressant de réfléchir à une solution qui traiterait des syntagmes complexes, des expressions polylexicales, au lieu de se cantonner aux alignements mot-à-mot. Dans un cas comme l'exemple proposé dans le paragraphe précédent, le programme pourrait très bien aligner le mot « coffee » à l'intégralité du syntagme nominal « le café », en ne séparant pas le déterminant et le nom. Il en serait de même pour les expressions idiomatiques qui ne sont pas du tout traitées dans cette version de l'extracteur lexical : l'expression « il pleut des cordes » en français diffère bien de sa version la plus proche en anglais « *it's raining cats and dog*s » qui donne traduite mot-à-mot « il pleut des chats et des chiens ». Dans un tel cas, l'alignement, n'a pas d'intérêt s'il se fait mot-à-mot et non syntagme par syntagme. Pour traiter les expressions polylexicales, il existe différentes stratégies. L'une d'entre elles serait d'identifier ces expressions en amont du traitement, pendant la phase de tokenisation, afin de les traiter comme un seul bloc plutôt que comme des mots séparés comme c'est actuellement le cas. Il faudrait pour cela s'assurer que l'aligneur est capable de son côté d'identifier ces expressions polylexicales.

26 Données inexploitables parmi celles qu'on cherche à étudier.

3. Affichage des résultats

Enfin, à la liste des améliorations pouvant être apportées à cet outil d'extraction, je peux aussi ajouter la question du nombre de mots à afficher comme réponse à la recherche par mot clé de l'utilisateur. Il est déjà possible d'afficher toutes les traductions possibles, ou simplement le mot qui a le nombre d'occurrences le plus élevé. Or, dans le cas où le mot aurait été retrouvé sous de très nombreuses formes dans sa version traduite, il faudrait ajouter une fonction d'affichage de quelques-uns des mots les plus souvent utilisés et se servir d'une option particulière pour afficher tous les résultats. Cela permettrait peut-être une lecture plus lisible et plus pertinente pour l'utilisateur, car sans quoi il risquerait de perdre des informations. Une étude approfondie basée cette fois sur l'expérience utilisateur me permettrait sans doute d'obtenir une réponse.

Conclusion

1. Résumé

E-Lexic est un logiciel, programmé en Python, conçu dans le but d'extraire des mots de textes bilingues préalablement alignés au niveau phrastique et lexical afin de les enregistrer dans un dictionnaire spécifique au corpus traité. L'outil est fonctionnel, et permet après le traitement d'effectuer une recherche par mot clé dans les dictionnaires ainsi créés.

Il s'agit bien entendu d'une première version qui reste perfectible, mais qui possède déjà de bonnes bases pour la suite. Du fait d'une durée du stage relativement courte[27] pour faire plus de recherches, j'ai dans un premier temps tout mis en œuvre pour écrire un script fonctionnel pour cette application. Il était important pour moi de terminer au moins cette partie du développement et d'arriver jusqu'à la création du dictionnaire bilingue à partir d'un alignement lexical du texte bilingue. Je souhaitais en effet aboutir à un logiciel qui répondrait à une requête utilisateur. Il faut néanmoins envisager de prendre plus de temps pour se pencher sur le développement de l'application afin de corriger certains points qui restent à améliorer, et surtout pour prendre le temps de faire plus de tests sur d'autres aligneurs afin de les comparer et de trouver s'il en existe un plus performant que *fast_align*.

2. Bilan personnel

Durant la période où j'ai effectué ce stage, j'ai appris à être autonome. Il a fallu que je m'organise seul : ainsi j'ai préparé et respecté un emploi du temps que je m'étais fixé pour me permettre de travailler efficacement à la réalisation de ce projet.

J'ai appris à travailler sur un domaine que je ne maîtrisais pas encore, celui de la traduction automatique. J'ai ainsi pu développer mes connaissances dans ce domaine de la traduction, j'ai aussi dépassé mes limites dans le langage Python afin d'obtenir les résultats que j'attendais. J'ai également appris à effectuer une série de tests sur un programme afin d'évaluer ses possibilités et savoir en tirer le meilleur.

Enfin j'ai connu la satisfaction de démarrer un projet relativement complexe, de le mener à terme dans le temps qui m'était imparti. Si le doute a pu s'emparer de moi à certains moments, j'ai néanmoins relevé ce défi, et j'ai fini par me prouver à moi-même que j'étais capable d'accomplir des tâches diverses : de la réflexion, de la conception, des tests, ou encore du développement.

27 Le stage a démarré mi-mai, nous laissant environ trois mois pour repartir sur un tout nouveau projet.

Toutefois, s'il m'était donné la possibilité de changer une chose dans ma manière de travailler durant ce stage, je sais que je voudrais trouver un lieu consacré au travail, en dehors de chez moi. Il m'a en effet été bien plus difficile qu'escompté de travailler seul chez moi. Durant cette période de crise sanitaire, les bibliothèques et autres lieux propices à la concentration que j'aime fréquenter pour travailler habituellement étaient pour la plupart fermés, ou à éviter pour ne pas prendre de risques. C'est un confort qui m'a manqué parfois, sans pour autant directement impacter mon travail.

Pour conclure, j'ai abordé ce travail de mémoire comme je l'espérais, et je pense avoir respecté le quatrième accord toltèque énoncé par Don Miguel Ruiz « Faites toujours de votre mieux. » (1997). Ce travail a été d'une intensité différentes des autres exercices auxquels j'ai participé durant mon parcours scolaire et universitaire, mais pouvoir le mener à bien fut quelque chose de gratifiant.

Bibliographie

Brown, P., Della Pietra, S., Della Pietra, V. & Mercer, R. (1993). The Mathematics of Statistical Machine Translation : Parameter Estimation. *Computational Linguistics*, 19(2), 263-311.

Och, F. & Ney, H. (2003). A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1) , 19-51.

Moore, R. (2005). A Discriminative Framework for Bilingual Word Alignment. *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, 81-88.

Dyer, C., Chahuneau, V. & Smith, N. (2013). A Simple, Fast, and Effective Reparameterization of IBM Model 2. *Proc. of NAACL*.

Sitographie

Centre National de Ressources Textuelles et Lexicales. (Août 2020). Repéré à l'adresse : https://www.cnrtl.fr/

Bathelot, B. (Août 2020). *KPI*. Repéré à l'adresse : https://www.definitions-marketing.com/definition/kpi/

XTM International. (Mars 2020). Repéré à l'adresse : https://XTM.cloud

SDL Trados. (Mars 2020). Repéré à l'adresse : https://www.sdltrados.com/fr/

LBS suite. (Mars 2020). Repéré à l'adresse : http://lbs-software.com/lbs-software.com/

XTRF. (Août 2020). Repéré à l'adresse : https://xtrf.eu/

Projetex. (Mars 2020). Repéré à l'adresse : https://www.projetex.com/

GitHub de fast_align(Août 2020) : https://github.com/clab/fast_align.

GitHub du projet E-Lexic (Septembre 2020) : https://github.com/Florent-Marie/E-Lexic

GitHub de HMM-Aligner (Septembre 2020) : https://github.com/sfu-natlang/HMM-Aligner

Distance de Levenshtein (Septembre 2020). Repéré à l'adresse : https://fr.wikipedia.org/wiki/Distance_de_Levenshtein

Apostrophe (typographie). (Septembre 2020). Source : https://fr.wikipedia.org/wiki/Apostrophe_(typographie)

Glossaire

Traduction : Transposition d’un texte d’une langue à une autre.

Textes bilingues : Paires de textes rédigés dans deux langues différentes, et qui sont une traduction l’un de l’autre.

Alignement : Deux items mis en correspondance, soit par un lien visible, soit selon une suite de paires de nombres dans le cas du logiciel fast_align présenté dans ce mémoire par exemple.

Corpus : Ensemble de données. Exemple : un corpus de textes est un rassemblement de plusieurs textes.

Suite (logicielle) : collection de logiciels au fonctionnalités liées, utilisant la plupart du temps une même interface graphique pour tous les logiciels.

Token : entité lexicale (mot ou ponctuation).

Sigles et abréviations utilisés

CNRTL : Centre National de Ressources Textuelles et Linguistiques.

KPI : Key Performance Indices, ou Indicateur de Performances Clés.

HMM :Hidden Markov Model, ou Modèle de Markov Caché.

TAO : Traduction Assistée par Ordinateur.

Table des illustrations

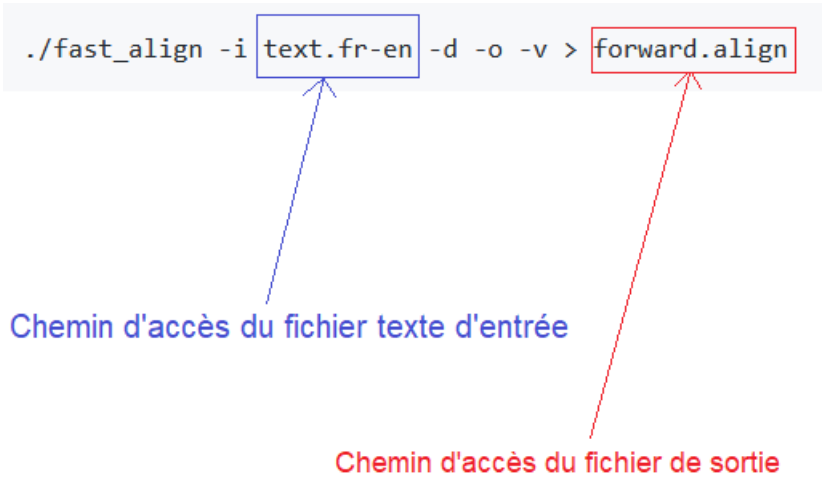
Table des annexes

Table des annexes

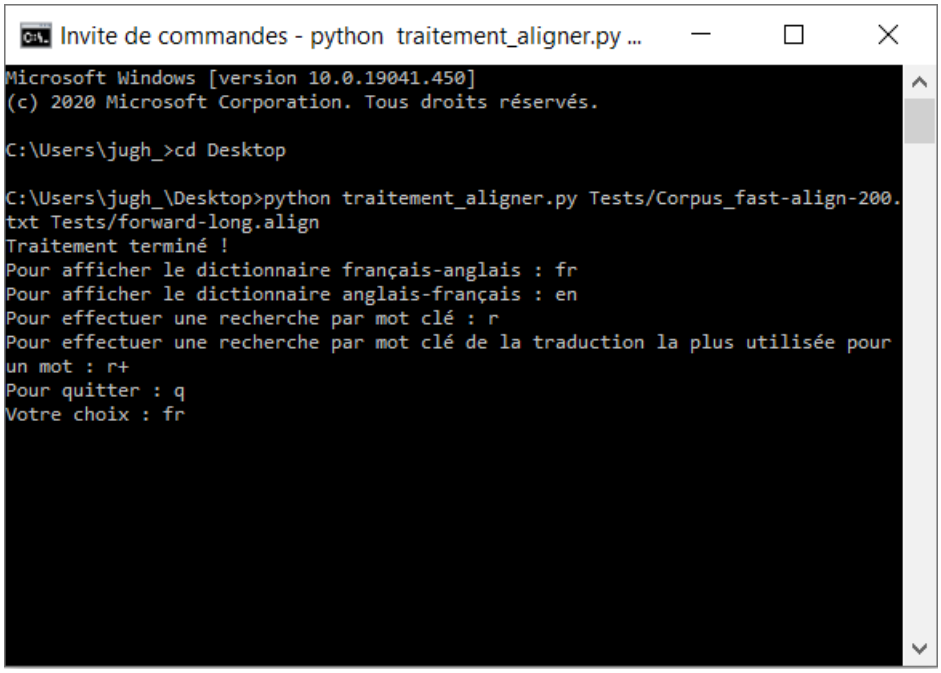
Annexe 1
Veille concurrentielle : logiciels de gestions de projets de traduction

	XTM	Memsource	SDL Business Ma	XTRF	SmartCat	Eazylang	TPBox	LBS	Projetex
Gestion de projet	4	1	4	4	4	3	5	4	4
Devis /factures	0	0	1	1	1	1	1	1	1
Suivi de projet	1	1	1	1	1	0	1	1	1
Tableau de bord personnalisable	1	0	1	1	0	0	1	0	1
Notifications et e-mails	1	0	0	0	1	1	1	1	0
Transferts de fichiers	1	0	1	1	1	1	1	1	1
TAO (Traduction Assistée par Ordinateur)	3	5	3	2	3	3	0	5	0
TAO intégrée	3	3	3	0	3	3	0	3	0
Mémoire de traduction	1	1	1	0	1	1	0	1	0
Traduction Machine	1	1	1	0	1	1	0	1	0
Editeur de texte	1	1	1	0	1	1	0	1	0
Ajout d'une TAO externe à l'outil	0	2	0	2	0	0	0	2	0
Interface	2	3	3,5	2,5	2,5	4	3,5	4	3,5
Ergonomie	0	1	1	1	1	1	1	1	1
Portails	0	0	1,5	1,5	1,5	2	1,5	2	1,5
Accessible client / fournisseur (/1,5)	0	0	1,5	1,5	1,5	1,5	1,5	1,5	1,5
Interface personnalisable (/0,5)	0	0	0	0	0	0,5	0	0,5	0
Intuitivité	1	1	1	0	0	1	1	1	1
Application mobile	1	1	0	0	0	0	0	0	0
Stockage des données	2,5	2,5	5	2,5	2,5	2,5	2,5	2,5	2,5
En ligne/Cloud	2,5	2,5	2,5	2,5	2,5	2,5	2,5	0	0
Local	0	0	2,5	0	0	0	0	2,5	2,5
Communication	4	5	4	3	5	4	4	1	3
Nombre de langues sur site (/2)	1	2	2	0	2	1	1	1	0
Avis(/2)	2	2	1	2	2	2	2	0	2
Affichage des prix	1	1	1	1	1	1	1	0	1
Automatisation	5	4	1	5	3	1	2	0	1
Total points (/30)	20,5	20,5	20,5	19	20	17,5	17	16,5	14
Prix (à partir de 10 employés)	538€/mois	150€/mois	2495 € (licence) + Business manager (7500)+1650 d'installation+Contrat d'assistance : 20% du prix des licences couvertes (annuel)	8€/jour	Service gratuit avec les fonctionnalités de base puis des prix allant jusqu'à 949$/mois (offre smart)	Version marketplace : Prix libre pour le traducteur	Formule sur abonnement à partir de 59 €/mois + hébergement (49€/mois HT ou hébergement dédié à partir de 84€/mois HT) possibilité d'hébergement sur site avec devis	5000€/ licence à vie + prix des modules	Licence à vie : 1995€ (pour 1 poste de travail); pour 10 : 12450€
			revient à environ 200 euros/mois	240euros/ mois					

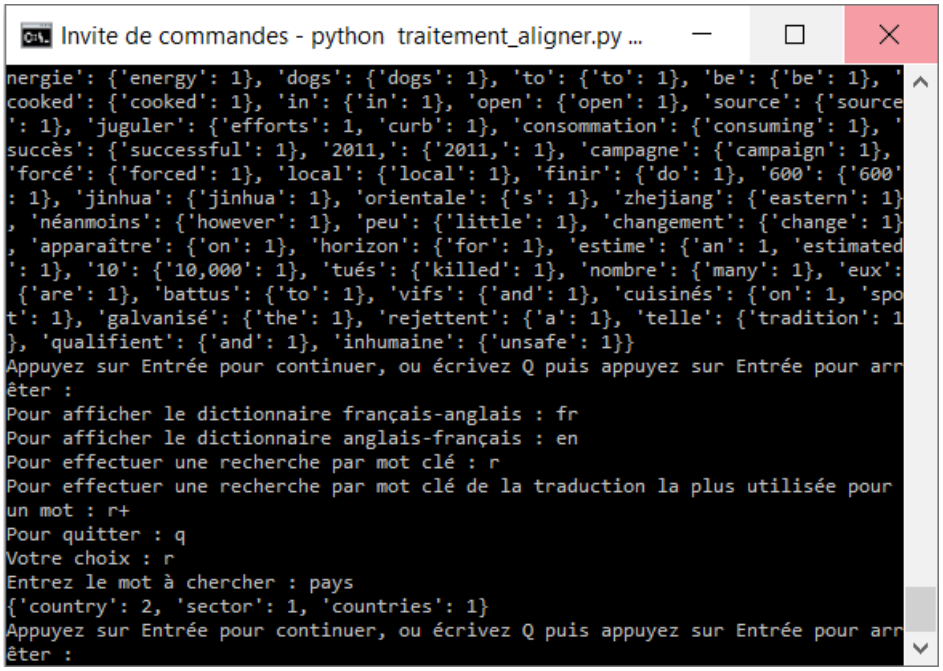
Annexe 2
Schéma explicatif de la ligne de commande de lancement de *fast_align*



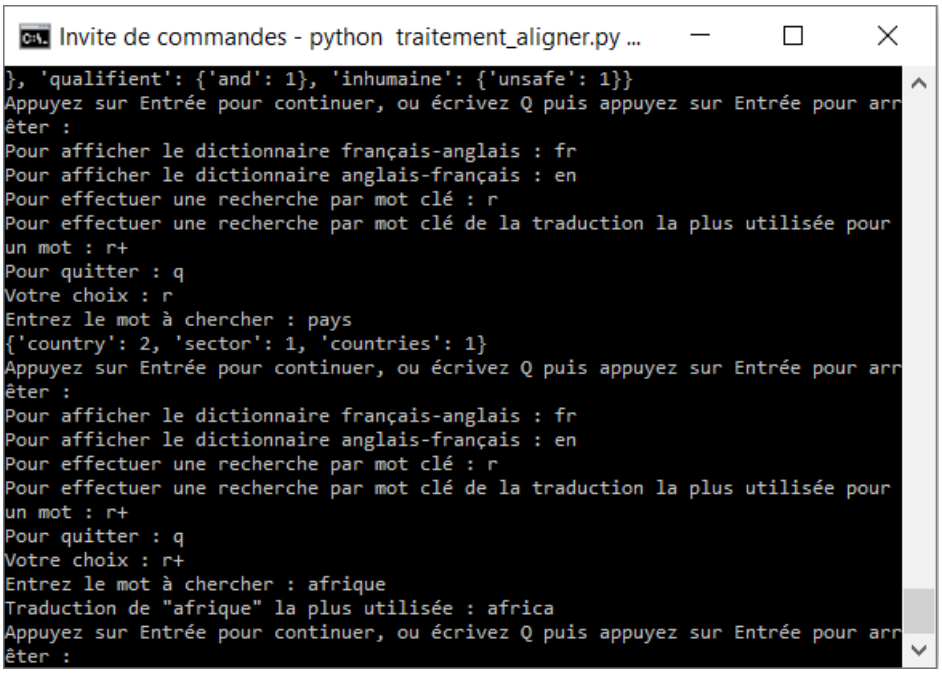
Annexe 3
Affichage d'un dictionnaire bilingue spécifique avec *E-Lexic*



Annexe 4
Affichage d'une recherche par mot clé avec *E-lexic*



Annexe 5
Résultat de recherche de la meilleure traduction par mot clé avec *E-lexic*



Annexe 6
Indications d'utilisation de *fast-align-format.py*

```
"""
    Programme qui fusionne deux textes dans un unique fichier texte
    après les avoir nettoyés afin que leur format soit accepté par
    fast_align.

    Usage :
        python fast-align-format.py texte1 texte2

        ou :

        python fast-align-format.py texte1 texte2 nomSortie

    Arguments:

        texte1 -- Chemin d'accès au fichier contenant le texte
        dans la langue source

        texte2 -- Chemin d'accès au fichier contenant le texte
        dans la langue cible

        nomSortie -- Chemin d'accès du fichier de sortie contenant la
        fusion des deux textes. En cas d'absence de cet argument, un chemin
        d'accès par défaut sera donné : Corpus_fast-align.txt.
"""
```

Annexe 7
Indications d'utilisation de *traitement_aligner.py*

```
"""
    Programme qui produit l'extraction lexicale selon l'alignement de chaque
    mot et qui enregistre chaque mot du texte et tous ses alignements
    dans deux dictionnaires, l'un de la langue source à la langue cible,
    l'autre dans le sens inverse.
    Les deux dictionnaires sont ensuite consultables par l'utilisateur.

    Usage :
        python traitement_aligner.py nomFichierTexte nomFichierAlignement

    Arguments:

        nomFichierTexte -- Chemin d'accès au fichier contenant le texte
        bilingue utilisé par l'aligneur

        nomFichierTexte -- Chemin d'accès au fichier contenant l'alignement
        réalisé par Fast-align
"""
```

Table des matières

Table des matières

MOTS-CLÉS : traduction, alignement lexical, extracteur de mots, dictionnaire

RÉSUMÉ

On a vu naître, avec l'émergence de nombreux outils de traitement automatique des langues, de nombreux domaines en linguistique computationnelle. Parmi eux on retrouve la traduction assistée par ordinateur. Dans ce sous-domaine de la traduction, on a là encore observé l'apparition de différents outils numériques comme la mémoire de traduction ou ce qui nous intéressera dans ce mémoire, les aligneurs syntagmatiques et lexicaux. Ce mémoire rend compte d'un travail fourni lors du développement d'un extracteur de mots basé sur l'alignement lexical de textes bilingues. L'objectif étant la création grâce à cette extraction d'un dictionnaire bilingue spécifique au texte traité consultable par un utilisateur.

KEYWORDS : translation, word alignment, word extractor, dictionnary

ABSTRACT

With the emergence of numerous automatic language processing tools, many fields in computational linguistics have been brought to light. Among them we can find computer-assisted translation. In this sub-domain of translation, we have again observed the appearance of various digital tools such as translation memory or, what will interest us in this thesis, syntactic and lexical aligners. This dissertation reports on work done in the development of a word extractor based on the lexical alignment of bilingual texts. The objective being the creation, thanks to this extraction, of a bilingual dictionary. This dictionary will be specific to the text that is being processed and can be consulted by a user.