Université de Montréal

MOOD: Un cadre d'applications pour le développement de décodeurs en traduction statistique

par
Alexandre Patry

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de Maître ès sciences (M.Sc.)
en informatique

Avril, 2006

Université de Montréal
Faculté des études supérieures

Ce mémoire intitulé:

MOOD: Un cadre d'applications pour le développement de décodeurs en traduction statistique

présenté par:

Alexandre Patry

a été évalué par un jury composé des personnes suivantes:

Guy Lapalme,	président-rapporteur
Philippe Langlais,	directeur de recherche
Jian-Yun Nie,	membre du jury

Mémoire accepté le: .....

RÉSUMÉ

Un système de traduction statistique traduit un document source par le document cible ayant la plus grande probabilité de le traduire. Un tel système est composé d'un modèle et d'un décodeur. Le modèle calcule la probabilité qu'un document en traduise un autre et le décodeur utilise le modèle pour trouver le document cible ayant la plus grande probabilité de traduire un document source.

Dans ce mémoire, je m'intéresse aux décodeurs et je présente MOOD, un cadre d'applications écrit en C++ que j'ai conçu pour les mettre en œuvre. À l'aide de MOOD, j'ai développé RAMSES, un décodeur basé sur les segments de phrases qui est similaire à PHARAOH (Koehn, 2004), le décodeur de référence dans la littérature. Afin d'encourager la recherche, MOOD et RAMSES sont distribués dans le domaine public accompagnés de leur code source.

La comparaison que j'ai effectuée entre RAMSES et PHARAOH m'a mené à la conclusion que les deux décodeurs produisent des traductions de qualité similaire. Pour comparer RAMSES à d'autres décodeurs, j'ai participé à la tâche partagée de traduction automatique organisée dans le cadre de l'atelier *Workshop on Machine Translation* s'étant tenu à New-York les 8 et 9 juin 2006. Il en est ressorti que RAMSES produit des traduction de qualité similaire aux systèmes suivant les règles de l'art.

Mots clés : décodeur, décodeur basé sur les segments de phrases, intelligence artificielle, traduction automatique, traduction statistique.

ABSTRACT

A statistical machine translation system translates a source document by the target document having the highest probability to translate it. Such a system is made up of a model and a decoder. The model computes the probability that one document translates another and the decoder uses the model to find the target document with the highest translation probability.

In this Master's Thesis, I am interested in decoders and I introduce MOOD, a C++ framework to implement such decoders. I then show how I used MOOD to implement RAMSES, a phrase-based decoder that is similar to PHARAOH (Koehn, 2004), a baseline system widely used in the community. To promote research, MOOD and RAMSES are published in the public domain accompanied with their source code.

The comparison that I have carried out between RAMSES and PHARAOH leads me to conclude that both decoders produce translations of similar quality. In order to compare RAMSES against other decoders, I have participated in a shared task that was organized in the *Workshop on Machine Translation* that was held in New-York on 8-9 June 2006. The results of this shared task show that the quality of the translations produced by RAMSES is similar to that produced by state-of-the-art systems.

Keywords: artificial intelligence, decoder, machine translation, phrase-based decoder, statistical machine translation.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ALGORITHMES

LISTE DES ANNEXES

À Cindy et Noah.

REMERCIEMENTS

Je tiens à remercier mon directeur de recherche, Philippe Langlais, pour ses critiques constructives, sa compréhension et le soutien qu'il m'a apporté tout au long de ma maîtrise. Je remercie aussi Fabrizio Gotti pour l'aide qu'il m'a apportée lors de l'évaluation automatique de RAMSES. Finalement, je veux exprimer ma reconnaissance à Elliott Macklovitch pour avoir accepté de participer à l'évaluation manuelle de RAMSES et pour les cafés sans lesquels je n'aurais sûrement pas encore terminé ce mémoire.

CHAPITRE 1

INTRODUCTION

Le but d'un système de traduction automatique est de traduire un document d'une langue naturelle (la langue source) vers une autre (la langue cible) sans intervention humaine. Ce problème est complexe et il occupe plusieurs chercheurs depuis le début de la deuxième moitié du vingtième siècle (Hutchins, 2001).

La traduction automatique s'est principalement développée autour de trois paradigmes. Un premier est la traduction symbolique. En traduction symbolique, un expert encode explicitement ses connaissances et un système les utilise ensuite pour traduire un document source.

Un deuxième paradigme, qui a gagné en popularité au cours des dernières années, est celui de la traduction statistique. Ce dernier se différentie du paradigme de la traduction symbolique principalement sur deux points. Premièrement, plutôt que de nécessiter qu'un expert encode ses connaissances, un système de traduction statistique modélisera automatiquement ses connaissances à partir d'un bitexte (un document en langue source accompagné de sa traduction en langue cible). Deuxièmement, plutôt que d'affirmer qu'un seul document cible peut traduire un document source, un système de traduction statistique affirme que tous les documents cibles traduisent un document source, mais avec différentes probabilités.

Un troisième paradigme est celui de la traduction basée sur les exemples. Tout comme la traduction statistique, la traduction basée sur les exemples apprend ses connaissances à partir d'un bitexte. Mais plutôt que d'utiliser ce bitexte pour créer un modèle statistique, ce paradigme l'utilise pour faire des analogies lors de la traduction d'un nouveau document source.

Certains systèmes de traduction automatique ont été utilisés avec succès, mais pour traduire des textes spécialisés. Un des systèmes les plus marquant de l'histoire de la traduction automatique est sans doute le système MÉTÉO, un système de traduction symbolique qui a été développé par le groupe de recherche TAUM de l'Université de Montréal dans les années 1970. Différentes versions de ce système ont été développées par la suite (Chandioux, 1988) et Environnement Canada utilise encore aujourd'hui un

système automatique pour traduire les bulletins météorologiques qu'il émet. Récemment, Langlais et al. (2005b) ont également mis en œuvre un système de traduction statistique produisant des traductions de qualité similaire. De leur côté, Way et Gough (2005) ont comparé un système de traduction statistique à un système de traduction basé sur les exemples et sont arrivés à la conclusion que les deux systèmes produisent des traductions de qualité similaire. Il n'a donc pas encore été prouvé qu'un paradigme soit supérieur aux autres.

Cependant, avec la disponibilité grandissante de différents bitextes, les paradigmes de la traduction statistique et de la traduction basée sur les exemples ont gagné en popularité. Parmi les bitextes populaires, nous comptons les débats parlementaires canadiens qui sont traduits en français et en anglais et les débats parlementaires européens qui sont traduits en 11 langues. Certains outils ont aussi été mis au point pour créer des bitextes à partir de sites bilingues publiés sur internet (Kraaij et al., 2003; Patry et Langlais, 2005).

Dans le cadre de ce mémoire, je me suis intéressé à la traduction statistique et plus précisément aux décodeurs. Ce chapitre décrit le rôle des décodeurs en traduction statistique et ensuite le cadre théorique dans lequel ils opèrent. Il continue en présentant un décodeur simple qui est ensuite utilisé pour établir une structure générale permettant de décrire un décodeur quelconque. Ce chapitre se conclut en résumant le contenu de ce mémoire.

1.1 La traduction statistique

En traduction statistique, traduire un document source se résume à trouver le document cible ayant la plus grande probabilité d'être sa traduction. Malgré la simplicité de cette formulation, le problème est d'envergure.

Il faut tout d'abord concevoir un modèle. Le modèle définit la forme des documents (séquence de mots, séquence de mots étiquetés morphosyntaxiquement, arbres, ...) et établit une fonction de densité qui calcule la probabilité qu'un document cible traduise un document source donné. Le grand défi lors de la conception d'un modèle est de créer une fonction de densité à partir d'un bitexte.

Une fois le modèle en main, il est utilisé par un décodeur. Le décodeur utilise le modèle pour générer le document cible ayant la plus grande probabilité de traduire un

document source donné. Cette tâche est complexe car l'espace de recherche ne peut être exploré en entier. Il faut donc choisir judicieusement le sous-espace qui le sera.

1.2 Cadre théorique d'un décodeur

Le décodeur utilise la fonction de densité fournie par le modèle pour générer le document cible ayant la plus grande probabilité de traduire un document source donné. Cette tâche peut être accomplie en résolvant l'équation suivante :

$$t^\star = \underset{t \in \mathcal{T}}{\operatorname{argmax}} \Pr(t|s) \tag{1.1}$$

où s est le document source et $\mathcal{T}$ est l'ensemble des documents de la langue cible.

Telle que présentée, l'équation 1.1 est impraticable ; pour traduire un document source, elle demande l'évaluation de tous les documents de la langue cible. Cette formulation est aussi contre-intuitive. Traduire un document se fait habituellement étape par étape, en appliquant une séquence de transformations à une traduction initiale (par exemple, en débutant avec un document cible vide qui sera construit petit à petit en traduisant le document source un mot à la fois).

Ces transformations peuvent être introduites dans l'équation 1.1 à l'aide de la variable δ_1^n :

$$t^\star = \underset{t \in \mathcal{T}}{\operatorname{argmax}} \sum_{\delta_1^n \in \Delta(s,t)} \Pr(t, \delta_1^n|s) \tag{1.2}$$

où $\Delta(s,t)$ est l'ensemble des séquences de transformations pouvant traduire s par t et δ_1^n un membre de cet ensemble. Cette dernière équation est équivalente à l'équation 1.1, mais elle plus aisée à résoudre en pratique. L'ensemble des séquences de transformations qui s'appliquent à un document source est habituellement limité par une base de connaissances (par exemple, un dictionnaire bilingue), ce qui borne le nombre de documents cibles à évaluer.

Même si l'équation 1.2 limite le nombre de documents cibles à évaluer, elle demeure complexe à résoudre. Pour réduire cette complexité, les décodeurs résolvent plutôt l'équation suivante :

$$\hat{t} = \underset{t \in \mathcal{T}}{\operatorname{argmax}} \max_{\delta_1^n \in \Delta(s,t)} \Pr(t, \delta_1^n|s) \tag{1.3}$$

Il est important de noter que les équations 1.2 et 1.3 ne sont pas équivalentes. Alors que la première trouve le document cible le plus probable, la deuxième trouve la séquence de transformations la plus probable. Nous espérons bien entendu que $t^\star$ et $\hat{t}$ soient égaux dans la plupart des cas.

L'Algorithme 1.1 présente un algorithme général pour résoudre l'équation 1.3. L'idée derrière cet algorithme est de faire évoluer parallèlement un ensemble de traductions incomplètes jusqu'à ce qu'elles soient complètes ou mises de côté parce qu'elles sont trop peu probables.

Algorithme 1.1 Algorithme général utilisé par les décodeurs en traduction statistique.

1. initialiser l'ensemble des traductions candidates $\mathcal{H}$ avec une traduction initiale
2. choisir une traduction incomplète h de $\mathcal{H}$
3. pour chaque transformation δ qui s'applique à h
   (a) mémoriser le résultat de l'application de δ sur h dans h_δ
   (b) si h_δ est prometteuse, l'ajouter à $\mathcal{H}$
4. si $\mathcal{H}$ contient au moins une traduction incomplète, aller à 2
5. retourner la meilleure traduction candidate de $\mathcal{H}$

Pour identifier les traductions candidates prometteuses (ligne 3b), l'Algorithme 1.1 demande au modèle d'évaluer la probabilité d'un document cible en cours de traduction. Le modèle doit donc être conçu en conséquence.

Pour simplifier le processus de traduction, la plupart des modèles et des décodeurs posent l'hypothèse que les phrases d'un document sont indépendantes les unes des autres. Un document peut donc être traduit une phrase à la fois. Cette simplification oblige chaque phrase source à n'être traduite que par une et une seule phrase cible ; aucune fusion ou segmentation de phrase n'est permise. À partir de maintenant, pour simplifier la discussion, je traiterai de la traduction d'une phrase plutôt que la traduction d'un document.

1.3 Le décodeur du voyageur

Pour décrire un premier décodeur, nous allons supposer qu'un voyageur unilingue francophone, armé d'un dictionnaire bilingue, entre dans une boutique New-Yorkaise. En consultant la liste des prix, il remarque une ligne qui lui est incompréhensible :

Anglais	Français
A	Un, Une
sheet	drap, feuille
of	de, du
paper	papier

Figure 1.1 – Dictionnaire bilingue du voyageur pour traduire la phrase *A sheet of paper*.

```
A sheet of paper .. .. .. .. 0.25$
```

Cette section fait un premier lien entre le cadre théorique vu à la section 1.2 et la pratique en décrivant le processus que le voyageur utilisera pour décoder cette phrase qui lui est inconnue.

Le voyageur dispose de deux ressources pour l'aider à traduire la phrase anglaise : un dictionnaire bilingue (Figure 1.1) et sa connaissance du français. Avec ces ressources, il peut spécialiser l'Algorithme 1.1 et traduire une phrase anglaise en utilisant l'Algorithme 1.2.

Algorithme 1.2 Algorithme utilisé par le décodeur du voyageur.

1. initialiser l'ensemble des traductions candidates $\mathcal{H}$ avec une phrase française vide.
2. choisir la traduction incomplète h de $\mathcal{H}$ ayant le moins de mots anglais traduits
3. pour chaque mot français δ traduisant le prochain mot à traduire de h
   (a) concaténer δ à h et mémoriser la phrase française résultante dans h_δ
   (b) si h_δ est probable, l'ajouter à $\mathcal{H}$
4. si $\mathcal{H}$ contient au moins une traduction incomplète, aller à 2
5. retourner la phrase française la plus probable de $\mathcal{H}$

Le voyageur débute avec une phrase française vide qu'il complétera en traduisant la phrase anglaise un mot à la fois. Parce qu'un mot anglais peut avoir plusieurs traductions françaises (par exemple, *sheet* se traduit par *feuille* ou *drap*), le voyageur devra maintenir plusieurs traductions françaises en parallèle.

Quand une phrase candidate lui semble très peu probable (par exemple *Une drap*), le voyageur l'oublie. Cela lui évite de dépenser des efforts inutilement et lui permet de traduire la phrase plus rapidement.

Une fois l'ensemble des phrases candidates complètes en main, le voyageur utilise son intuition pour choisir celle qu'il retiendra. Dans cet exemple, il choisit *Une feuille de*

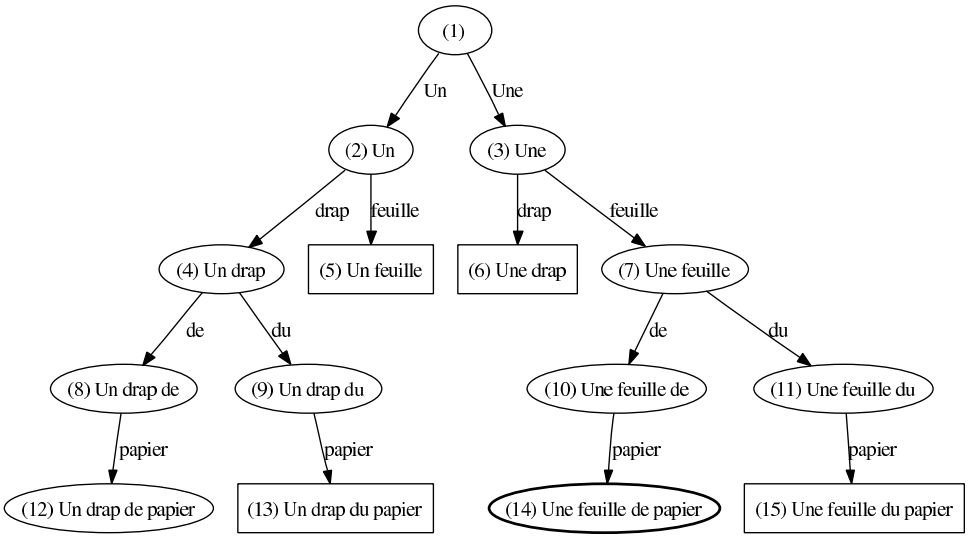

Figure 1.2 – Graphe de recherche pour la phrase *A sheet of paper*.

papier parce que cette phrase lui semble plus probable que la phrase *Un drap de papier*.

Le graphe de recherche exploré pour traduire la phrase *A sheet of paper* est présenté à la Figure 1.2. Chaque ellipse représente une traduction candidate (complète ou incomplète) et chaque boite rectangulaire une traduction improbable qui a été abandonnée. Les arcs représentent les transformations nécessaires pour passer d'une phrase candidate à une autre.

Malgré la simplicité de ce décodeur, sa complexité est quand même exponentielle. Si N est le nombre de mots dans la phrase source et b le nombre maximal de traductions permises pour un mot source donné, le nombre de traductions possibles est de l'ordre de $O(b^N)$ et le nombre de sommets dans le graphe de recherche de l'ordre de $O(\sum_{i=1}^{N} b^i)$. Si nous bornons b à cinq, une phrase de 10 mots peut être traduite par plus de 9 millions de phrases cibles et son graphe de recherche peut être composé de plus de 12 millions de sommets.

1.3.1 Un voyageur moins naïf

Si le voyageur devait traduire la phrase *the red car* à l'aide du dictionnaire présenté à la Figure 1.3, il arriverait sûrement à la phrase française *la rouge voiture* alors que la phrase *la voiture rouge*, même si elle est moins poétique, serait plus appropriée.

Anglais	Français
the	la, le
red	rouge
car	voiture

Figure 1.3 – Dictionnaire bilingue du voyageur pour traduire la phrase *the red car*.

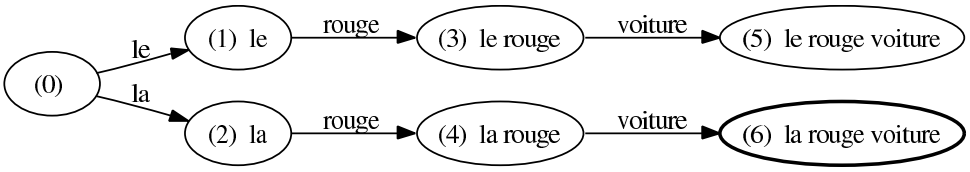

Figure 1.4 – Graphe de recherche pour une traduction monotone de la phrase *the red car*.

Il semble donc qu'il faille permettre au décodeur de traduire les mots dans un ordre arbitraire. Cette extension est nécessaire, mais elle rend le décodage plus complexe. Le graphe de recherche non élagué passe de 7 sommets (Figure 1.4) à 27 sommets (Figure 1.5). En permettant le réordonnancement des mots, le nombre de traductions complètes passe à $O(N!b^N)$ et le nombre de sommets passe à $O(\sum_{i=1}^{N} \binom{N}{i} i!b^i)$. Si nous bornons b à cinq, une phrase de 10 mots peut avoir plus de 35 000 milliards de traductions et son graphe de recherche peut être composé de plus de 43 000 milliards de sommets.

Créer un décodeur qui produise des traductions de qualité demande des modèles expressifs, ce qui engendre beaucoup de sommets dans le graphe de recherche. Cependant, plus le nombre de sommets est élevé, plus le temps nécessaire pour traduire une phrase augmente. La tâche du décodeur est donc de concilier la qualité des traductions produites et le temps nécessaire pour les générer en choisissant intelligemment la partie du graphe de recherche qui sera explorée.

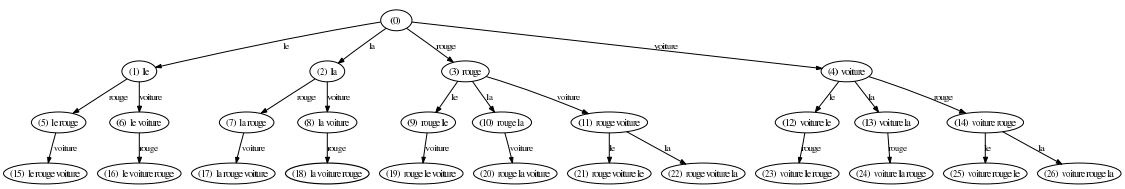

Figure 1.5 – Graphe de recherche pour une traduction non monotone de *the red car*.

1.4 Structure d'un décodeur

Dans la section précédente, j'ai présenté le décodeur du voyageur ainsi qu'une extension lui permettant de traduire les mots dans un ordre arbitraire. Même s'ils sont différents, ces décodeurs ont beaucoup en commun. La séparation d'un décodeur en plusieurs modules qui pourront être facilement combinés et remplacés semble donc être pertinente.

Lors de la mise en œuvre d'un décodeur, deux modules généraux doivent être identifiés. Le premier pour représenter le modèle et le deuxième pour décrire la manière dont l'espace de recherche sera exploré. Cette section s'attarde à décrire plus en détails ces deux modules.

1.4.1 Représentation du modèle

La représentation du modèle peut à son tour être divisée en quatre modules : les traductions partielles, les transformations, le coût et le générateur de transformations.

Une *traduction partielle* représente une phrase source en cours de traduction. Elle est composée de la phrase source, de la phrase cible (complète ou incomplète) et d'un indicateur de progression. La phrase source et la phrase cible peuvent être de simples séquences de mots, des mots accompagnés d'étiquettes morphosyntaxiques ou des arbres. De son côté, l'indicateur de progression indique si la phrase cible est complète ou sinon comment la compléter. La traduction partielle est représentée par un sommet dans le graphe de recherche.

Dans le décodeur du voyageur, la phrase source et la phrase cible sont de simples séquences de mots et l'indicateur de progression est la position du prochain mot à traduire.

La phrase source est traduite à l'aide d'une succession de *transformations*. Une transformation peut prendre plusieurs formes, dont par exemple la traduction d'un mot, la traduction d'une séquence de mots ou un réordonnancement des noeuds dans un arbre. La transformation est un arc dans le graphe de recherche.

Dans le décodeur du voyageur, une transformation ajoute un mot à la fin de la phrase cible et incrémente l'indice du prochain mot à traduire de un.

Pour décider des candidats à élaguer et de la phrase cible à retourner, le décodeur

doit évaluer la qualité d'une traduction partielle. Le *coût* permet une telle évaluation en quantifiant la qualité d'une traduction partielle à l'aide d'un nombre réel. Habituellement, le coût évalue au moins les aspects suivants d'une traduction :

- la qualité des transformations appliquées
- la qualité de la phrase cible générée
- le réordonnancement des mots entre la phrase source et la phrase cible

Un coût est associé à une traduction partielle, il fait donc partie du sommet du graphe de recherche.

Le voyageur évalue la qualité d'une traduction partielle à l'aide de son bon sens. Cette métrique a été choisie pour simplifier la présentation du décodeur, mais elle ne peut être employée en traduction automatique.

Le dernier module de la représentation du modèle est le générateur de transformations. Le *générateur de transformations* détermine comment une traduction partielle peut être complétée. Il prend en paramètre la traduction partielle à compléter et retourne les transformations qui peuvent lui être appliquées. C'est lui qui décide des arcs sortant de chaque sommet du graphe de recherche, il peut donc élaguer le graphe de recherche en omettant certains arcs.

Dans le décodeur du voyageur, le générateur de transformations retourne les transformations traduisant le mot suivant le dernier traduit.

1.4.2 Exploration de l'espace de recherche

La représentation du modèle permet de traduire une phrase source à l'aide d'une succession de transformations, mais elle ne spécifie pas l'ordre dans lequel les traductions candidates sont explorées ni quand les abandonner. C'est le rôle de l'explorateur d'espace de recherche. L'explorateur d'espace de recherche peut être divisé en deux modules indépendants : l'hypothèse et la stratégie de recherche.

Le rôle principal de l'*hypothèse* est de synchroniser une traduction partielle avec son coût. Ce module n'est pas essentiel, mais il facilite la discussion. En unissant une traduction partielle à son coût, une hypothèse devient un sommet du graphe de recherche.

La *stratégie de recherche*, de son côté, décide des hypothèses à explorer et de l'ordre dans lequel les explorer. Elle peut élaguer le graphe de recherche en mettant certains sommets de côté. Elle prend en entrée une hypothèse initiale et un générateur de trans-

formations et retourne une traduction complète.

La stratégie de recherche du décodeur du voyageur est une recherche en largeur où les phrases agrammaticales sont mises de côté. Comme il a déjà été mentionné, dans un décodeur réel, l'élagage du graphe de recherche se fait sur la base d'un coût exprimé par un nombre réel. La stratégie de recherche ne pourrait donc pas utiliser ce critère directement, mais elle pourrait le faire indirectement si la grammaticalité des phrases est considérée lors du calcul du coût. La grammaticalité pourrait aussi être renforcée par le générateur de transformations, qui pourrait omettre les transformations menant à des traductions partielles agrammaticales.

Une même stratégie de recherche peut être utilisée avec plusieurs représentations de modèles. L'inverse est aussi vrai, plusieurs stratégies de recherche peuvent être utilisées avec une même représentation de modèle.

1.4.3 Retour sur le décodeur du voyageur

Une extension au décodeur du voyageur où les mots peuvent être traduits dans un ordre arbitraire a été présentée à la section 1.3.1. Deux modifications sont nécessaires pour mettre cette extension en œuvre.

La première consiste à redéfinir l'indicateur de progression comme un ensemble contenant la position des mots sources déjà traduits. Ensuite, le générateur de transformations peut être modifié pour retourner les transformations traduisant n'importe quel mot source dont la position n'est pas dans l'indicateur de progression.

En ayant nommé les différents modules, il devient plus facile de décrire et de comparer différents décodeurs.

1.5 Résumé

Un décodeur traduit une phrase source à l'aide d'un modèle. Pour des raisons pratiques, il ne cherche pas à générer la phrase cible ayant la plus grande probabilité de traduire une phrase source (équation 1.2), mais plutôt la séquence de transformations ayant la plus grande probabilité de générer la traduction (équation 1.3).

Pour trouver cette séquence de transformations, un décodeur utilise un algorithme semblable à l'Algorithme 1.1. Souvent, l'algorithme se résume à transformer plusieurs

traductions candidates en parallèle jusqu'à ce quelles soient complètes ou abandonnées. Lorsque le processus se termine, la meilleure traduction complète est retournée.

Ce problème est complexe, car le nombre de sommets dans le graphe de recherche augmente rapidement. Il faut donc établir une politique d'élagage intelligente qui permettra d'atteindre un bon compromis entre la qualité des traductions produites et le temps nécessaire pour les produire.

Un décodeur peut être décrit à l'aide des modules qui le composent. La séparation entre la représentation du modèle et la stratégie de recherche était déjà établie. Ma contribution a été d'établir une structure de plus bas niveau pour la représentation du modèle en introduisant quatre modules : les traductions partielles, les transformations, le coût et le générateur de transformations. J'ai ensuite utilisé cette structure pour mettre en œuvre MOOD, un cadre d'applications facilitant le développement de décodeurs. MOOD est implémenté en C++ et utilise une architecture modulaire permettant la réutilisation de modules existants pour faciliter la conception de nouveaux décodeurs.

Quelques décodeurs sont disponibles gratuitement, comme REWRITE (Germann, 2003) et PHARAOH (Koehn, 2004), mais seulement dans leur forme binaire. C'est sur ce point que MOOD s'en différentie, car le cadre d'applications et son code sont tous deux distribués librement. Cette décision a été prise pour stimuler la recherche et encourager la diffusion des connaissances (Walker, 2005).

Pour confirmer l'utilité de MOOD, je l'ai utilisé pour développer RAMSES, un décodeur semblable à PHARAOH. RAMSES ne devait être qu'un système de base qui pourrait ensuite être utilisé pour créer de nouveaux décodeurs, mais il s'est avéré que la qualité des traductions qu'il produit est comparable à la qualité des traductions produites par des systèmes suivant les règles de l'art.

Au chapitre 2 je présente différentes stratégies de recherche qui ont été utilisées dans la littérature. Ensuite, au chapitre 3, je décris le fonctionnement de PHARAOH, un décodeur de référence dans la littérature. Le chapitre 4 présente MOOD et RAMSES en détails. Les expériences qui ont été menées pour évaluer RAMSES sont présentées au chapitre 5. Les conclusions de cette étude sont formulées au chapitre 6.

CHAPITRE 2

STRATÉGIES DE RECHERCHE

Jusqu'à maintenant, nous avons vu qu'un décodeur utilise une représentations de modèle et une technique de recherche pour trouver la meilleure séquence de transformations traduisant une phrase source. Ce chapitre présente différentes stratégies de recherche qui ont été puisées dans la littérature.

L'exploration de l'espace de recherche est d'abord définie comme un problème classique d'intelligence artificielle et ensuite, différentes stratégies de recherche sont présentées.

2.1 Le graphe de recherche

Représenter le décodage d'une phrase à l'aide d'un graphe a été naturel au chapitre 1. Dans cette section, je présente formellement le concept de graphe et je confirme notre intuition selon laquelle cette représentation est appropriée.

2.1.1 Représentation

Un graphe peut être exprimé par $\langle V, A \rangle$ où V est l'ensemble de ses sommets et $A \subseteq V \times V$ l'ensemble de ses arêtes. Si la direction des arêtes est importante ($\langle u, v \rangle \in A \nleftrightarrow \langle v, u \rangle \in A$), les arêtes sont appelées des arcs et le graphe est dit orienté.

Lorsque $\langle u, v \rangle \in A$, nous disons que v est successeur de u et que u est parent de v. Une séquence de sommets $v_1, v_2, \cdots, v_n$ est un chemin si v_{j+1} est un successeur de v_j pour toutes les valeurs de j entre 1 et $n-1$.

Si une hypothèse (traduction partielle et coût) correspond à un sommet et que deux sommets sont liés par un arc lorsqu'une transformation permet de passer de l'hypothèse du premier vers l'hypothèse du deuxième, nous obtenons un graphe orienté.

Nous avons vu que les hypothèses ne sont pas énumérées explicitement, mais qu'elles sont définies par le générateur de transformations. Le graphe de recherche est donc implicite. Un graphe orienté implicite est équivalent à un graphe orienté explicite (Nilsson, 1971), mais plutôt que d'avoir ses sommets et ses arcs énumérés, il est décrit à l'aide

d'un ensemble de sommets initiaux et d'un opérateur de succession $\Gamma(\cdot)$. L'opérateur de succession prend en entrée un sommet et retourne la liste des arcs qui en sortent.

Nous pouvons séparer les sommets du graphe en trois catégories : le sommet initial v_0, l'ensemble des sommets solutions $\mathcal{G} \subseteq V$ (sommets correspondant à des traductions complètes) et les autres sommets qui représentent les hypothèses intermédiaires. Trouver une séquence de transformations menant à une traduction complète revient donc à trouver un chemin entre v_0 et un sommet solution quelconque.

Comme le graphe est implicite, les sommets solutions ne sont pas connus à l'avance. Ils sont plutôt définis par un prédicat prenant en paramètre un sommet et retournant *vrai* si c'est un sommet solution.

Pour simplifier la discussion, j'utiliserai les termes sommet et hypothèse de façon interchangeable. Par exemple, le coût et la probabilité d'un sommet seront le coût et la probabilité de l'hypothèse qui lui est associée.

2.1.2 Recombinaison

L'*état* d'une hypothèse est l'ensemble des informations nécessaires pour trouver les transformations qui peuvent lui être appliquées et pour calculer les coûts induits par ces transformations. Deux hypothèses sont dans le même état si elles ne peuvent être différenciées par le générateur de transformations et par la fonction de coût.

Comme compléter la traductions de deux hypothèses dans le même état induit le même sous-graphe de recherche, ces hypothèses peuvent être recombinées. La *recombinaison* d'un ensemble d'hypothèses consiste à ne garder que l'hypothèse étant la plus prometteuse et à oublier les autres.

Dans le graphe de recherche, recombiner un ensemble de sommets consiste à ajouter au sommet ayant la plus grande probabilité les arcs entrants des autres sommets. Une fois ces arcs ajoutés, les sommets dont les arcs ont été redirigés et leurs successeurs peuvent être oubliés. Par exemple, les sommets 4 et 8 du graphe de recherche de la Figure 2.1 sont dans le même état. Une fois ces sommets recombinés, le graphe de recherche devient celui présenté à la Figure 2.2. La partie du graphe ayant été oubliée après la recombinaison est représentée en pointillés.

C'est en permettant la recombinaison des hypothèses dans le même état que le graphe de recherche peut avoir plus d'un chemin entre deux sommets.

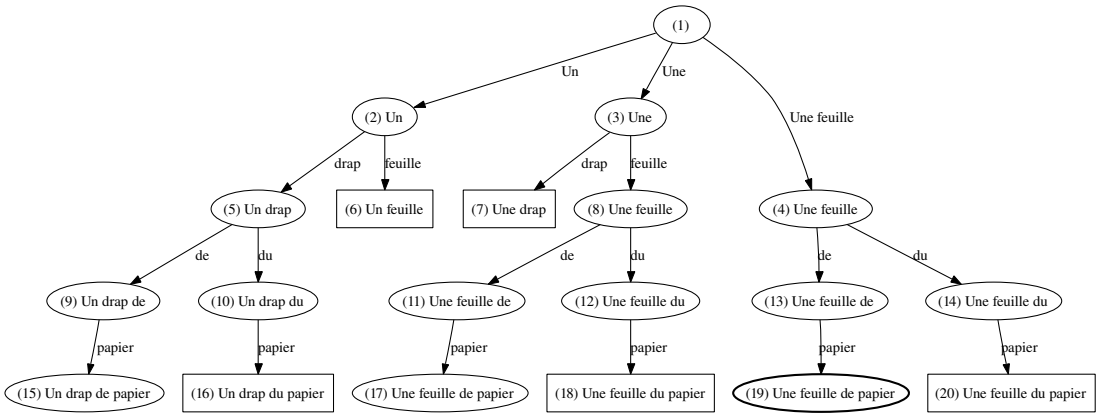

Figure 2.1 – Graphe de recherche où les sous-graphes enracinés aux sommets 4 et 8 sont les mêmes.

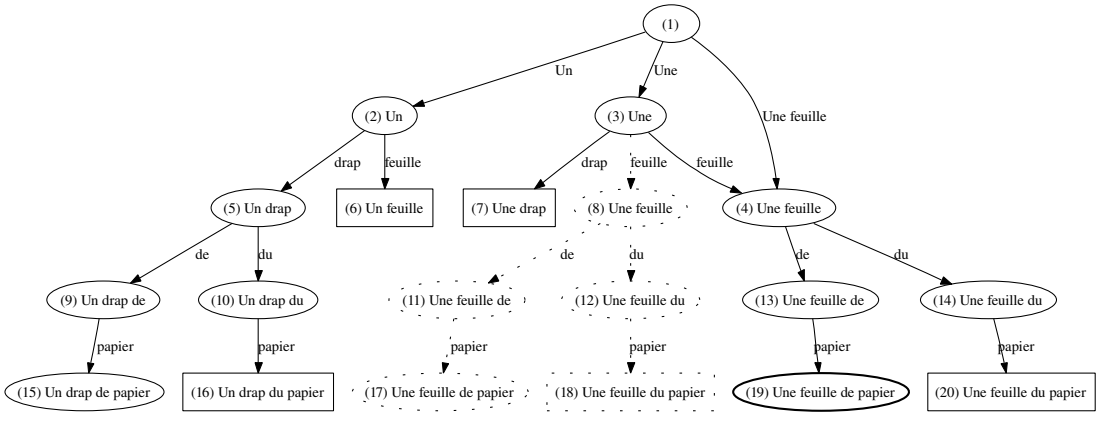

Figure 2.2 – Graphe de recherche de la Figure 2.1 où les sommets 4 et 8 ont été recombinés.

2.1.3 Un décodeur intelligent

L'objectif d'un décodeur est de trouver la séquence de transformations ayant la plus grande probabilité d'être appliquée à une hypothèse initiale pour arriver à une traduction complète. La probabilité de cette meilleure séquence de transformations peut être exprimée par :

$$\begin{aligned} p^\star &= \max_{\gamma \in \mathcal{G}} \Pr(\gamma) \\ &= \max_{\gamma \in \mathcal{G}} \max_{v_0^n \in \text{chemins}(v_0,\gamma)} \Pr(v_0) \prod_{i=0}^{n-1} \frac{\Pr(v_{i+1})}{\Pr(v_i)} \\ &= \Pr(v_0) \max_{\gamma \in \mathcal{G}} \max_{v_0^n \in \text{chemins}(v_0,\gamma)} \prod_{i=0}^{n-1} \frac{\Pr(v_{i+1})}{\Pr(v_i)} \end{aligned} \tag{2.1}$$

où $\Pr(v)$ est la probabilité de l'hypothèse associée au sommet v et $\text{chemins}(u, v)$ est l'ensemble des chemins liant le sommet u au sommet v. Il est habituellement trivial de modifier un algorithme optimisant cette expression pour qu'il garde trace du meilleur chemin.

Une partie de la communauté œuvrant en intelligence artificielle s'attaque à un problème similaire, celui de la résolution de problème (*problem solving*). En intelligence artificielle, la résolution de problème se résume à trouver le chemin de plus faible coût entre un sommet initial et un des sommets solutions (Nilsson, 1971). Le coût de ce chemin peut être exprimé par :

$$c^\star = \min_{\gamma \in \mathcal{G}} \min_{u_0^n \in \text{chemins}(v_0,\gamma)} \sum_{i=0}^{n-1} \text{cout}(\langle u_i, u_{i+1} \rangle) \tag{2.2}$$

où $c^\star$ est le coût du meilleur chemin et $\text{cout}(\langle u, v \rangle)$ est le coût associé à l'arc $\langle u, v \rangle$.

Parce que log est une fonction monotone et que $\Pr(v_0)$ est une constante, le chemin menant à la solution de l'équation 2.1 est le même que celui menant à la solution de

l'équation suivante :

$$\begin{aligned} c' &= \log \max_{\gamma \in \mathcal{G}} \max_{v_0^n \in \text{chemins}(v_0,\gamma)} \prod_{i=0}^{n-1} \frac{\Pr(v_{i+1})}{\Pr(v_i)} \\ &= \max_{\gamma \in \mathcal{G}} \max_{v_0^n \in \text{chemins}(v_0,\gamma)} \log \prod_{i=0}^{n-1} \frac{\Pr(v_{i+1})}{\Pr(v_i)} \\ &= \max_{\gamma \in \mathcal{G}} \max_{v_0^n \in \text{chemins}(v_0,\gamma)} \sum_{i=0}^{n-1} \log \frac{\Pr(v_{i+1})}{\Pr(v_i)} \end{aligned} \tag{2.3}$$

Finalement, parce que l'argument maximisant une fonction est le même que celui minimisant son opposé, le coût de la séquence de transformations la plus probable peut être exprimé par :

$$c^\star = \min_{\gamma \in \mathcal{G}} \min_{v_0^n \in \text{chemins}(v_0,\gamma)} \sum_{i=0}^{n-1} -\log \frac{\Pr(v_{i+1})}{\Pr(v_i)} \tag{2.4}$$

Si on définit le coût d'un arc par la fonction suivante :

$$\text{cout}(\langle u, v \rangle) = -\log \frac{\Pr(v)}{\Pr(u)} \tag{2.5}$$

traduire une phrase se réduit donc à un problème classique d'intelligence artificielle. Dans le cadre de ce mémoire, nous établissons la convention qu'un décodeur cherche la traduction maximisant un coût arbitraire.

2.1.4 Reconstruction de la meilleure solution

Une fois le décodage terminé, la meilleure traduction doit être reconstruite. Une traduction est reconstruite en appliquant à la traduction initiale les transformations associées aux arcs formant le meilleur chemin entre le sommet initial et un des sommets solutions.

Lorsque la recombinaison est permise, il peut y avoir plus d'un chemin entre le sommet initial et un sommet solution. Reconstruire la meilleure traduction n'est donc pas direct.

Pour simplifier la reconstruction de la meilleure traduction, chaque sommet garde une référence vers son meilleur parent. Cette référence doit être initialisée lorsque le sommet est exploré pour la première fois et elle doit être mise à jour à chaque recombinaison. Une fois la recherche terminée, retrouver le meilleur chemin se résume donc à suivre ces

références en commençant par le meilleur sommet solution trouvé.

Conserver le graphe de recherche permet aussi de retrouver la liste des N meilleures traductions ou de traduire en deux passages (Ueffing et al., 2002). La traduction en deux passages (*rescoring*) se fait en générant un premier graphe de recherche avec un coût simple et ensuite en utilisant ce graphe de recherche avec un coût plus complexe pour trouver une meilleure traduction.

2.2 Algorithmes de recherche

2.2.1 Programmation dynamique

La programmation dynamique évite de visiter deux fois le même sous-graphe en recombinant les sommets des hypothèses ayant le même état avant de les visiter. L'Algorithme 2.1 réussit cela en visitant les sommets dans l'*ordre topologique* (si $\langle u, v\rangle$ est un arc du graphe alors u est visité avant v). Une preuve que cet algorithme arrive à une solution optimale est présentée par Cormen et al. (1994, chapitre 25).

Algorithme 2.1 Technique de recherche utilisant la programmation dynamique.

1. initialiser l'ensemble des hypothèses candidates $\mathcal{H}$ avec une hypothèse initiale
2. choisir l'hypothèse $h \in \mathcal{H}$ étant la première selon l'ordre topologique
3. pour chaque transformation δ s'appliquant à h selon le générateur de transformations
   (a) mémoriser le résultat de l'application de δ sur h dans h_δ
   (b) ajouter h_δ à $\mathcal{H}$
   (c) si h_δ et une autre hypothèse ont le même état, les recombiner
4. si $\mathcal{H}$ contient au moins une hypothèse incomplète, aller à 2
5. retourner l'hypothèse de $\mathcal{H}$ ayant le plus grand coût

Comme l'ordre topologique n'est défini que sur les graphes orientés acycliques, l'Algorithme 2.1 ne peut être utilisé que si le graphe de recherche est acyclique. Donc, le modèle ne doit pas permettre une transformation et son opposée. Heureusement, la plupart des modèles ont cette propriété.

L'Algorithme 2.1 a été utilisé avec succès avec quelques décodeurs utilisant un modèle basé sur les mots (Tillman et al., 1997b; Nießen et al., 1998). Cependant, pour garder un temps de calcul raisonnable, le graphe de recherche est élagué en limitant les trans-

formations retournées par le générateur de transformations (ligne 3 de l'algorithme). Ce faisant, la garantie que le décodeur retourne une solution optimale ne tient plus.

2.2.2 Recherche A*

La programmation dynamique évite de visiter deux fois le même sous-graphe, mais elle explore tout de même tous les états de l'espace de recherche. Une recherche A* n'explore plutôt que les sommets prometteurs.

Pour guider la sélection des sommets à explorer, on utilise une fonction heuristique. La *fonction heuristique* estime le coût futur d'un sommet, le coût qu'engendrera le meilleur chemin entre ce sommet et un sommet solution. Une fois cette fonction heuristique intégrée au coût, une recherche A* se fait à l'aide de l'Algorithme 2.2.

Algorithme 2.2 Recherche A*. La fonction heuristique est intégrée au coût.

1. initialiser l'ensemble des hypothèses candidates $\mathcal{H}$ avec une hypothèse initiale
2. choisir l'hypothèse $h \in \mathcal{H}$ ayant le plus grand coût
3. si h est complète, la retourner
4. pour chaque transformation δ qui s'applique à h selon le générateur de transformations
   (a) mémoriser le résultat de l'application de δ sur h dans h_δ
   (b) ajouter h_δ à $\mathcal{H}$
5. si $\mathcal{H}$ n'est pas vide, aller à 2

Si la recherche A* trouve toujours le meilleur chemin, nous disons qu'elle est admissible. Il est prouvé que la recherche est admissible quand la fonction heuristique est une borne supérieure du coût futur (Nilsson, 1971).

Pour que la recherche soit efficace, la fonction heuristique doit accomplir deux objectifs qui sont souvent contradictoires : être un bon estimé du coût futur et s'exécuter rapidement. Comme cette fonction n'est qu'une partie du décodeur, la rapidité est habituellement le facteur qui l'emporte. Souvent, cette fonction est pré-calculée à partir de la phrase source avant que le décodage ne commence (Wang et Waibel, 1997; Och et al., 2001).

Pour limiter la mémoire utilisée pour la recherche, on peut limiter la taille de la pile[1].

[1] L'usage du terme pile est un accident historique. La structure utilisée demande plus que les opérations *push* et *pop*, mais j'ai néanmoins utilisé ce terme pour rester consistant avec la littérature.

Cela élimine cependant la garantie d'optimalité.

2.2.3 Recherche A* à plusieurs piles

Comme il a été mentionné à la section précédente, pour limiter la mémoire utilisée par une recherche A*, on peut limiter la taille de la pile de recherche. Si la fonction heuristique favorise certains groupes d'hypothèses (par exemple les hypothèses moins complètes ou les phrases cibles moins longues), il faut prendre des précautions lors de l'élagage parce que si l'hypothèse optimale n'appartient pas à ce groupe privilégié, elle pourrait être élaguée. Pour limiter ce risque, Wang et Waibel (1997) suggèrent d'utiliser plusieurs piles pour regrouper et élaguer ensembles les hypothèses comparables. La notion d'hypothèses comparables dépend du modèle, mais souvent le point de comparaison est le sous-ensemble des mots sources traduits ou leur nombre.

Une recherche A* à plusieurs piles est décrite dans l'Algorithme 2.3. Tout comme proposé par Germann et al. (2001), l'algorithme visite systématiquement la meilleure hypothèse de chaque pile. Une autre stratégie utilisée par Wang et Waibel (1997) est de n'explorer la meilleure hypothèse d'une pile que si son coût est près du coût de la meilleure parmi toutes les piles (la proximité peut être configurée par un seuil).

Algorithme 2.3 Recherche A* à plusieurs piles. Chaque hypothèse appartient à un groupe d'hypothèses qui sont comparables entre elles.

---

1. initialiser une pile pour chacun des n groupes $\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_n$
2. ajouter l'hypothèse initiale dans la pile de son groupe
3. $h_c \leftarrow nil$
4. pour $i \in [1, n]$
   (a) élaguer $\mathcal{H}_i$
   (b) choisir l'hypothèse $h \in \mathcal{H}_i$ ayant le meilleur coût
   (c) pour chaque transformation δ s'appliquant à h selon le générateur de transformations
      i. mémoriser le résultat de l'application de δ sur h dans h_δ
      ii. si h_δ est complète et meilleure que h_c, alors mémoriser h_δ dans h_c
      iii. si h_δ est incomplète, alors l'ajouter à la pile de son groupe
5. si $h_c \neq nil$ et que h_c est meilleure que les hypothèses de toutes les piles, alors la retourner
6. si au moins une pile n'est pas vide, aller à 4

Lorsqu'il y a plusieurs piles, une hypothèse complète ne peut être retournée aussitôt qu'elle est rencontrée. On doit s'assurer qu'elle est belle et bien la meilleure parmi toutes les piles avant de la retourner (ligne 5).

2.2.4 Recherche en faisceau à plusieurs piles

La recherche en faisceau à plusieurs piles allie les avantages de la programmation dynamique à ceux d'une recherche A* : elle organise sa recherche pour éviter de visiter plus d'une fois le même sous-graphes et elle utilise une fonction heuristique pour reconnaître les hypothèses prometteuses.

Pour accélérer la recherche, les piles sont élaguées. La recherche en faisceau n'offre donc aucune garantie d'optimalité. Comme dans la recherche A* à plusieurs piles, chaque pile ne devrait contenir que des hypothèses comparables. Donc, lors de l'élagage, seulement des hypothèses comparables sont comparées et la solution optimale risque moins d'être manquée. La recherche en faisceau à plusieurs piles est présentée dans l'Algorithme 2.4.

Algorithme 2.4 Recherche en faisceau à plusieurs piles.

1. initialiser une pile pour chacun des n groupes $\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_n$
2. ajouter l'hypothèse initiale dans la pile de son groupe
3. $h_c \leftarrow nil$
4. pour $i \in [1, n]$
   (a) recombiner les hypothèses de $\mathcal{H}_i$ ayant le même état
   (b) élaguer $\mathcal{H}_i$
   (c) pour chaque $h \in \mathcal{H}_i$
      i. pour chaque transformation δ s'appliquant à h selon le générateur de transformations
         A. mémoriser le résultat de l'application de δ sur h dans h_δ
         B. si h_δ est complète et meilleure que h_c, alors mémoriser h_δ dans h_c
         C. si h_δ est incomplète, alors l'ajouter à la pile de son groupe
5. retourner h_c

Les piles sont parcourues en ordre (ligne 4), l'hypothèse résultante de la transformation d'une hypothèse de la pile i doit donc toujours être ajoutée à une pile dont l'indice est plus grand que i. Si ce n'est pas le cas, la nouvelle hypothèse ne sera jamais explorée.

Quand le graphe de recherche est acyclique, il est avantageux de ne regrouper dans une pile que des hypothèses ayant le même rang selon l'ordre topologique. Ainsi, la recombinaison de la ligne 4a assurera que le même sous-graphe ne soit pas visité plus d'une fois.

Un autre aspect important de la recherche en faisceau est l'élagage des piles (ligne 4b). Och et al. (2001) et Tillmann et Ney (2003) proposent d'utiliser un *élagage par histogramme* qui ne conserve que les N meilleures hypothèses de chaque pile. Tillmann et Ney (2003) et Koehn et al. (2003) proposent aussi d'utiliser un élagage par seuil relatif. L'*élagage par seuil relatif* ne conserve que les hypothèses dont le coût est plus élevé que t multipliée par le coût de la meilleure hypothèse de la pile. Ces deux techniques d'élagage peuvent être utilisées ensemble et N et t sont des paramètres spécifiés au décodeur.

Parce que les piles sont élaguées, il n'y a aucune garantie que la recherche trouve toujours la solution optimale. Une recherche en faisceau bien paramétrée peut cependant la trouver dans la plupart des cas. En pratique, le compromis entre la vitesse et la qualité des traductions justifie souvent le préférence de la recherche en faisceau à plusieurs piles par rapport à un algorithme admissible (Och et al., 2001).

2.2.5 Autres algorithmes

Je n'ai présenté que les stratégies de recherche qui ont été à mon avis les plus importantes. Il y en a cependant d'autres qui ont été utilisées avec succès en traduction statistique.

Par exemple, une stratégie de recherche utilisant la programmation dynamique itérative est présentée par Garcia-Varea et Casacuberta (2001) et une stratégie gloutonne cherchant à améliorer une traduction existante par Germann (2003). Knight (1999) a réduit le décodage au problème du commis voyageur et utilise un algorithme de programmation entière pour le mener à bien.

2.3 Conclusion

Le recherche de la meilleure séquence de transformations à appliquer à une hypothèse initiale pour traduire une phrase source est un problème classique d'intelligence artificielle. Plusieurs algorithmes d'intelligence artificielle utilisés avec succès en traduction

statistique ont été présentés dans ce chapitre.

La qualité d'une stratégie de recherche peut dépendre de plusieurs facteurs comme la politique d'élagage, la qualité de la fonction heuristique, le critère décidant de la pile dans laquelle sera entreposée une hypothèse et le nombre d'hypothèses pouvant être recombinées. L'algorithme choisi doit donc être paramétré soigneusement.

Le constat général semble cependant être que la recherche en faisceau à plusieurs piles est la solution à privilégier (Och et al., 2001; Tillmann et Ney, 2003). Si l'élagage est bien paramétré, la solution optimale est généralement trouvée et le temps de calcul est nettement amélioré. Il ne faut pas oublier non plus que le temps de calcul peut être réduit en implantant une politique d'élagage dans le générateur de transformations.

Une comparaison entre une recherche en faisceau et une recherche utilisant la programmation dynamique a été menée par Tillman et al. (1997a). La qualité de la sortie des deux décodeurs était similaire, mais la recherche en faisceau réduisait le nombre de sommets visités d'un facteur d'environ 230.

Une autre comparaison cette fois-ci entre une recherche en faisceau et une recherche A* est présentée par Och et al. (2001). Ils ont rapporté que pour la traduction de 50 phrases de 12 mots, une recherche en faisceau a retourné une hypothèse sous-optimale pour quatre phrases, mais qu'elle a nécessité environ 14 fois moins de temps que la recherche A*. Une autre comparaison entre ces deux algorithmes a été menée par Ortiz et al. (2003) où les auteurs ont obtenu un temps d'exécution pour la recherche en faisceau qui est similaire au temps nécessaire pour mener une recherche A*. Ils expliquent que la recherche en faisceau passe la majeure partie de son temps à élaguer des hypothèses, mais ils ne spécifient pas les détails de leur implémentation. Ces résultats pourraient s'expliquer par un mauvais choix d'algorithmes ou de structures de données.

La recherche en faisceau à plusieurs piles a donc été adoptée dans la plupart des décodeurs récemment développés (Tillmann et Ney, 2003; Vogel et al., 2003; Koehn, 2004; Simard et al., 2005). La recherche A* garde tout de même son attrait, car elle permet d'obtenir la solution optimale, elle peut donc être utilisée pour évaluer l'optimalité d'autres stratégies de recherche.

CHAPITRE 3

REVUE DES DÉCODEURS

Dans le chapitre 2, j'ai présenté plusieurs stratégies pour explorer un espace de recherche, mais sans décrire les modèles définissant ces espaces de recherche.

Dans ce chapitre, je présente un décodeur de référence pour les modèles basés sur les segments de phrases. Ce décodeur est décrit à l'aide des différents modules présentés à la section 1.4 : les traductions partielles, les transformations, le coût, le générateur de transformations et la stratégie de recherche.

3.1 Modèles basés sur les segments de phrases

Les modèles basés sur les segments de phrases alignent des segments de la phrase source à des segments de la phrase cible. Capturer les relations au niveau des segments de phrases plutôt qu'au niveau des mots facilite la traduction d'expressions comme *piece of cake* qui se traduit par *jeu d'enfants* plutôt que par *part de gâteau* dans la phrase *it was a piece of cake*. Un exemple d'alignement de segments de phrases est présenté à la Figure 3.1. Habituellement, les modèles basés sur les segments de phrases capturent les relations entre des segments quelconques, qu'ils soient motivés grammaticalement ou non.

Un décodeur pour les modèles basés sur les segments de phrases cherche à optimiser l'équation :

$$\hat{t} = \underset{t \in \mathcal{T}}{\operatorname{argmax}} \max_{p \in \mathcal{P}(s,t)} \Pr(t, p|s) \tag{3.1}$$

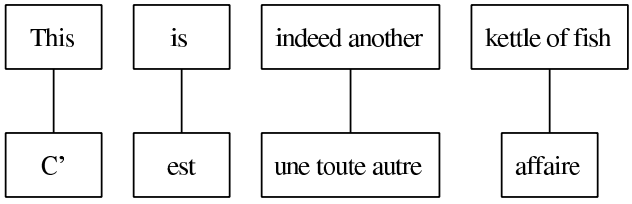

Figure 3.1 – Un alignement au niveau des segments de phrases entre une phrase source et une phrase cible.

où $\mathcal{P}(s,t)$ est l'ensemble des alignements de segments de phrases possibles entre s et t.

3.1.1 Pharaoh

Dans cette section, je décris PHARAOH (Koehn, 2004), un décodeur populaire pour la traduction à l'aide de modèles basés sur les segments de phrases. PHARAOH construit la phrase cible de la gauche vers la droite, mais n'impose aucune restriction sur l'ordre dans lequel la phrase source est traduite. Ce décodeur est distribué gratuitement dans sa forme binaire[1].

3.1.1.1 Modèle

PHARAOH utilise un modèle exponentiel (Och et Ney, 2002) :

$$\Pr(t|s) = \frac{\exp \sum_{m=1}^{M} \lambda_m h_m(s,t)}{\sum_{t' \in \mathcal{T}} \exp \sum_{m=1}^{M} \lambda_m h_m(s,t')} \tag{3.2}$$

$$t^\star = \underset{t \in \mathcal{T}}{\operatorname{argmax}} \sum_{m=1}^{M} \lambda_m h_m(s,t) \tag{3.3}$$

où h_m est une fonction de coût et λ_m son poids.

Le modèle utilisé par PHARAOH est fait de deux sous-modèles. Le premier est un modèle de langue, qui évalue la probabilité qu'une certaine phrase appartienne au langage cible. Le calcul de cette probabilité est simplifié en posant une hypothèse markovienne, voulant qu'un mot ne dépende que des n mots qui le précède. PHARAOH utilise un trigram, un modèle de langue d'ordre deux :

$$p(t_1^l) = \prod_{i=1}^{l+1} p(t_i|t_{i-1}, t_{i-2}) \tag{3.4}$$

où t_1^l est la phrase de longueur l à évaluer et t_i est le i-ème mot de cette phrase. Les mots cibles t_{-1} et t_0 sont des mots spéciaux pour indiquer le début de la phrase et le mot t_{l+1} un mot spécial pour indiquer la fin de la phrase. Un tel modèle de langue est habituellement entraîné à l'aide d'outils comme SRILM (Stolcke, 2002) ou CMU-LM (Clarkson et Rosenfeld, 1997) qui sont disponibles gratuitement.

[1] `http://www.isi.edu/licensed-sw/pharaoh/`

Le deuxième sous-modèle est une table de traduction $\Pr(\bar{t}|\bar{s})$ retournant la probabilité que le segment de phrase $\bar{s}$ soit traduit par le segment de phrase $\bar{t}$. Le modèle $\Pr(\bar{t}|\bar{s})$ est approximé par $t(\bar{t}|\bar{s})$, qui est une table de traduction associant des probabilités à un ensemble de paires $\langle \bar{s}, \bar{t} \rangle$.

Deux approches ont été développées pour construire une table de traduction. La première considère qu'une phrase est faite d'un ensemble de concepts, représentés par des variables cachées, qui se manifestent par des segments de phrases (Marcu et Wong, 2002). Ce modèle est théoriquement justifié, mais il est lourd à calculer.

La deuxième utilise des techniques ad-hoc sur un alignement de mots, habituellement obtenu à l'aide du logiciel GIZA++ (Och et Ney, 2003), pour extraire des segments de phrases (Venugopal et al., 2003; Koehn et al., 2003; Och et Ney, 2004). Koehn et al. (2003) rapportent avoir obtenu des traductions de meilleure qualité avec une approche ad-hoc qu'avec le modèle de Marcu et Wong (2002). Le programme THOT (Ortiz-Martínez et al., 2005), disponible gratuitement, permet de générer une table de traduction en utilisant des techniques ad-hoc.

En plus du modèle de langue et de la table de traduction, PHARAOH utilise un coût pour contrôler le réordonnancement des segments de phrases et un autre pour contrôler la longueur de la phrase cible. L'équation que le décodeur cherche à résoudre est :

$$\begin{aligned} \hat{t} = \underset{t \in \mathcal{T}}{\operatorname{argmax}} \max_{\langle \bar{s}_i^I, \bar{t}_i^I \rangle \in \mathcal{P}(s,t)} & \lambda_p \sum_{i=1}^{l} \log p(t_i|t_{i-1}, t_{i-2}) + \\ & \lambda_t \sum_{i}^{I} \log t(\bar{t}_i|\bar{s}_i) + \\ & \lambda_d \sum_{i=1}^{I} -|a_i - (b_{i-1} + 1)| + \lambda_w(-l) \end{aligned} \tag{3.5}$$

où a_i et b_i sont respectivement la position du premier et du dernier mot de $\bar{s}_i$ et I est le nombre de segments de phrases. Le coût $\sum_{i=1}^{I} -|a_i - (b_{i-1} + 1)|$ permet de contrôler le réordonnancement des segments de phrases en ajoutant une pénalité proportionnelle aux nombre de mots sources séparant deux transformations et le coût $-l$ permet de contrôler la longueur des phrases produites en ajoutant une pénalité proportionnelle aux nombre de mots générés dans la phrase cible. Les valeurs optimales pour les différents λ_m peuvent être obtenues à l'aide de l'algorithme décrit par Och (2003).

3.1.1.2 Traductions partielles

La phrase source et la phrase cible sont de simples séquences de mots. L'indicateur de progression est une paire $\langle \Pi, j \rangle$ où Π est l'ensemble des positions des mots sources déjà traduits et j est la position du prochain mot à traduire pour que la traduction soit monotone.

La traduction initiale est composée de la phrase source, d'une phrase cible vide et d'un indicateur de progression ayant la forme $\langle \{\}, 1 \rangle$. Une traduction est complète si toutes les positions de la phrase source se retrouvent dans Π.

3.1.1.3 Transformations

Le seul type de transformation nécessaire dans PHARAOH est la traduction d'un segment de la phrase source $\bar{s}$ par un segment de phrase cible $\bar{t}$. Si a et b sont respectivement la première et la dernière position occupée par $\bar{s}$, cette transformation ajoute $\bar{t}$ à la fin de la phrase cible et change l'indicateur de progression pour qu'il devienne $\langle \Pi \cup \{a, a+1, \ldots, b\}, b+1 \rangle$.

3.1.1.4 Coût

Le coût d'une traduction est défini dans l'équation 3.5. La fonction heuristique utilisée par PHARAOH correspond à la somme du coût de la traduction de chaque séquence de mots non traduits comme si ces séquences étaient des phrases indépendantes. Cette fonction peut être implémentée à l'aide d'une table calculée avant que le décodage ne commence (Koehn, 2004).

L'état du coût est composé des deux derniers mots de la phrase cible (modèle de langue), de la position du mot suivant le dernier segment de phrase source couvert (coût pour le contrôle du réordonnancement) et des positions de l'ensemble des mots sources déjà traduits (coût de la fonction heuristique).

3.1.1.5 Générateur de transformations

Chaque fois qu'il est appelé, le générateur de transformations doit retourner l'ensemble des transformations applicables à la traduction partielle qu'il reçoit en paramètre. Pour qu'une transformation soit applicable, elle ne doit pas traduire un mot source

déjà traduit. Habituellement, le générateur de transformations ne retourne qu'un sous-ensemble des transformations permises pour que le temps de calcul reste raisonnable.

PHARAOH permet de restreindre le nombre de transformations traduisant un même segment de phrase $\bar{s}$ en ne considérant que celles pour lesquelles $t(\bar{t}|\bar{s})$ est le plus élevé. Il permet aussi de limiter le nombre de mots séparant deux segments de phrase source traduits consécutivement.

L'état du générateur de transformation est donc composé des positions de la phrase source qui sont couvertes et, si le réordonnancement est limité, de la position du mot source suivant le dernier segment de phrase source couvert.

3.1.1.6 Stratégie de recherche

Comme chaque transformation traduit au moins un mot source, une hypothèse est plus petite qu'une autre selon l'ordre topologique si moins de mots sources y sont traduits.

PHARAOH utilise une recherche en faisceau à plusieurs piles où deux hypothèses sont dans la même pile si elles ont le même nombre de mots sources traduits. Comme les hypothèses sont explorées dans l'ordre topologique, elles sont recombinées.

3.1.2 Autres décodeurs

Federico et Bertoldi (2005) ont vérifié si l'ajout de segments de phrases de fertilité nulle (alignés à un segment de phrase vide) améliorait la qualité des traductions. Leurs expériences ont plutôt démontré le contraire. La détérioration des résultats pourrait être expliquée par le fait que les mots de fertilités nulles sont déjà présents dans les segments de phrases de la table de traduction.

Les décodeurs présentés par Koehn et al. (2003) et Federico et Bertoldi (2005) utilisent un modèle de canal bruité plutôt qu'un modèle exponentiel pour modéliser le coût maximisé. Une comparaison de ces deux modèles menée par Och et Ney (2002) est cependant arrivée à la conclusion que le modèle exponentiel arrive à des traductions de meilleure qualité.

Le décodeur présenté par Vogel et al. (2003) fait un décodage en deux passages (voir section 2.1.4). Le premier passage ne permet aucun réordonnancement et n'utilise que le coût $t(\bar{t}|\bar{s})$. Le deuxième passage est exécuté à l'aide d'une recherche en faisceau utilisant

le coût $p(t)t(s|t)$ et en limitant le réordonnancement à l'aide d'un modèle d'alignement HMM (Vogel et al., 1996).

Le choix d'une recherche en faisceau semble donc être unanime dans les décodeurs basés sur les segments de phrases.

3.2 Autres modèles

Les premiers modèles à avoir été populaires en traduction statistique sont les modèles basés sur les mots (Brown et al., 1993; Vogel et al., 1996). Même si les traductions produites par ces modèles sont de moins bonne qualité que celles produites en utilisant des modèles basés sur les segments de phrases (Koehn et al., 2003; Federico et Bertoldi, 2005), les modèles basés sur les mots sont encore inévitables en traduction statistique. Ils sont entre autres utilisés pour produire des alignements de mots qui peuvent, à l'aide de méthodes ad-hoc, être utilisés pour construire une table de traduction pour un système de traduction basé sur les segments de phrases. Les modèles basés sur les mots sont habituellement entraînés à l'aide du logiciel libre GIZA++ (Och et Ney, 2003).

Simard et al. (2005) et Langlais et al. (2005a) utilisent des modèles où le segment de phrase ajouté par une transformation peut contenir des trous qui seront remplis plus tard par une autre transformation. Le décodeur qu'ils utilisent est fortement inspiré de PHARAOH.

D'autres recherches sont aussi menées sur les modèles hiérarchiques. Ces modèles considèrent la phrase cible comme un arbre plutôt que comme une séquence de mots. Un premier essai avait été entrepris sans trop de succès par Knight (1999), mais de nouveaux modèles hiérarchiques (Galley et al., 2004; Chiang, 2005) semblent très prometteurs.

3.3 Conclusion

Dans ce chapitre, j'ai utilisé la structure présentée à la section 1.4 pour présenter PHARAOH, un décodeur décodeur de référence dans la littérature.

La recherche semble maintenant s'orienter vers les modèles basés sur les segments de phrases troués et les modèles hiérarchiques. Je crois que la structure avec laquelle j'ai décrit les décodeurs présentés dans ce chapitre pourrait aussi être utilisée pour décrire des décodeurs pour ces nouveaux modèles.

CHAPITRE 4

MISE EN OEUVRE D'UN DÉCODEUR

Alors que les chapitres précédents présentaient différents décodeurs, ce chapitre décrit comment les mettre en oeuvre à l'aide de MOOD, un cadre d'applications pour le développement de décodeurs.

Ce chapitre débute par la présentation du cadre d'applications et il décrit ensuite comment MOOD a été utilisé pour mettre en oeuvre RAMSES, un décodeur basé sur les segments de phrases. Les avantages de l'architecture modulaire de MOOD sont ensuite exposés. Finalement, le chapitre se termine en présentant les décisions qui ont été prises pour faire de MOOD un projet encourageant la recherche et la diffusion des connaissances.

4.1 MOOD

Tout au long de ce mémoire, la description d'un décodeur a toujours été séparée en six parties distinctes ayant chacune un rôle particulier (traduction partielle, transformation, coût, générateur de transformations, hypothèse et stratégie de recherche). Cette séparation est aussi utilisée pour mettre en oeuvre un décodeur à l'aide de MOOD (*Modular Object-Oriented Decoder*), un cadre d'applications pour développer des décodeurs.

Chaque partie du décodeur est implémentée dans une classe. Comme il sera argumenté à la section 4.3, cette modularisation a comme objectif principal la réutilisation des différents modules lors de la création de nouveaux décodeurs. Toujours par esprit de réutilisation, MOOD garde aussi la représentation du modèle indépendante des techniques de recherche.

Les classes permettant de mettre en oeuvre un décodeur à l'aide de MOOD sont présentées dans la Figure 4.1. Cette section décrit les méthodes, les dépendances et les contraintes de chacune de ces classes.

4.1.1 Représentation du modèle

Le modèle spécifie comment faire évoluer une traduction partielle et comment en évaluer la qualité. MOOD représente un modèle à l'aide de quatre classes. Une servant à

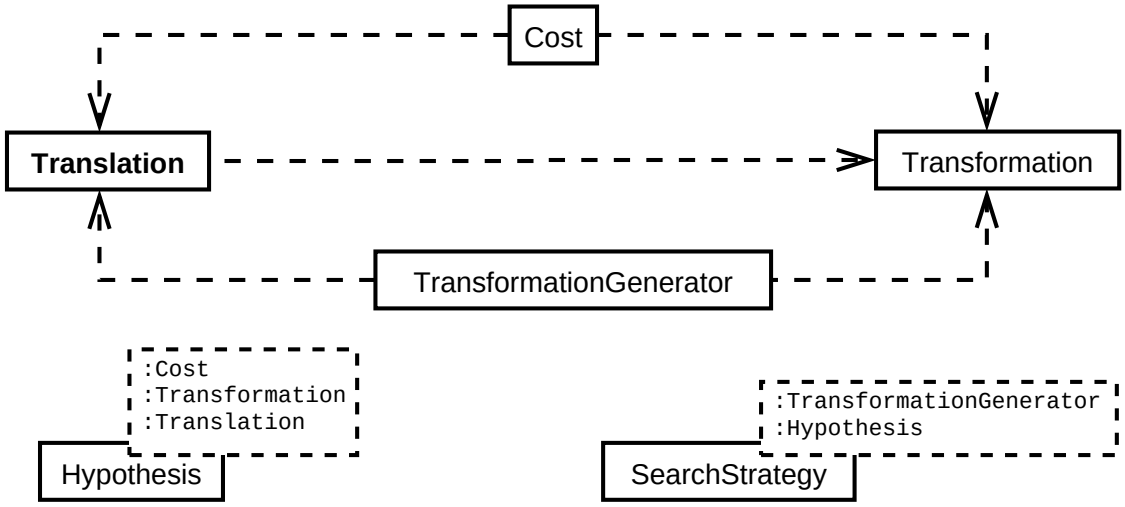

Figure 4.1 – Diagramme UML de la structure d'un décodeur développé à l'aide de MOOD. Les flèches pointillées indiquent les interdépendances entre les différentes classes.

définir une traduction partielle (*Translation*), une autre à représenter une transformation (*Transformation*), une pour quantifier la qualité d'une traduction partielle (*Cost*) et une dernière générant les transformations pouvant être appliquées à une traduction partielle (*TransformationGenerator*).

4.1.1.1 Traduction partielle

Un décodeur ne traduit pas une phrase source instantanément, mais fait plutôt évoluer une traduction initiale vers une traduction complète, une transformation à la fois. Une traduction partielle représente une traduction à n'importe quel moment pendant ce processus et est implémentée dans MOOD à l'aide d'une classe semblable à celle présentée à la Figure 4.2.

Une traduction partielle est composée de la phrase source à traduire, de la phrase cible en cours de construction et d'un indicateur de progression. Ce dernier mémorise des informations sur la progression de la traduction qui pourront être utilisées par le générateur de transformations pour déterminer les transformations étant applicables à la traduction partielle.

La tâche principale d'une traduction partielle est de se transformer pour éventuellement devenir complète. Une transformation est spécifiée à l'aide d'un objet de type

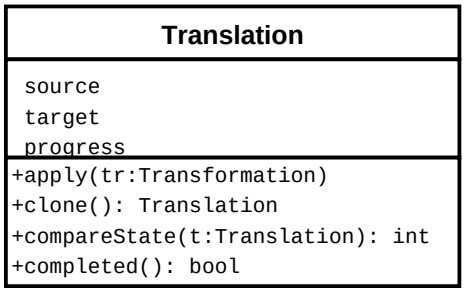

Figure 4.2 – Une classe représentant une traduction partielle.

Transformation qui est appliqué à l'aide de la méthode *apply*. Lorsque la traduction partielle est complète, la méthode *completed* retourne *vrai*.

Plusieurs transformations sont habituellement applicables à une même traduction partielle. Une copie de la traduction partielle doit alors être produite pour chacune de ces transformations. C'est le rôle rempli par la méthode *clone*, qui retourne une copie de sa traduction partielle.

La mise en oeuvre d'une traduction partielle dépend de l'implémentation des transformations, car la méthode *apply* doit en connaître les détails. Cela signifie que si la classe *Transformation* est modifiée, il est fort probable que la classe *Translation* doive l'être aussi.

Des traductions partielles seront transformées et copiées fréquemment tout au long du décodage. Cela doit être considéré lors du choix des structures de données utilisées pour implémenter une traduction partielle. Ces structures devraient permettre des applications de transformations rapides tout en étant rapides à copier.

4.1.1.2 Transformation

Traduire une phrase source se fait étape par étape. Chaque changement d'étape est représenté par une transformation, qui spécifie à une traduction partielle comment s'approcher un peu plus d'une traduction complète. Une transformation est définie dans la Figure 4.1 par la classe *Transformation*.

Il est très difficile de définir une interface qui sera commune à tous les types de transformations. Dans un modèle basé sur les segments de phrases, une transformation pourrait ajouter quelques mots à la phrase cible alors que dans un modèle hiérarchique

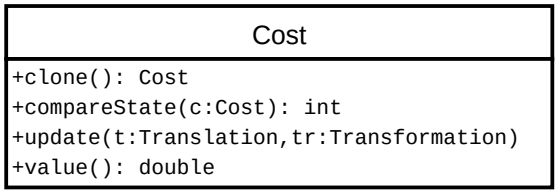

Figure 4.3 – Une classe représentant un coût.

elle pourrait ajouter un noeud intermédiaire à un arbre en cours de construction. Pour que le modèle reste général, j'ai décidé de n'imposer aucune contrainte sur l'implémentation des transformations.

4.1.1.3 Coût

La stratégie de recherche ne peut habituellement pas explorer l'espace de recherche en entier. Elle évalue chaque traduction partielle afin de déterminer s'il vaut la peine d'investir des ressources pour la compléter. Chaque traduction partielle est donc associée à un objet lui associant un coût, qui par convention sera élevé pour les traductions partielles prometteuses. Un coût est représenté dans MOOD à l'aide d'une classe semblable à celle présentée à la Figure 4.3.

La tâche principale d'un coût est de se mettre à jour lorsqu'une traduction partielle est transformée. Cette mise à jour est effectuée lors de l'appel à la méthode *update*, qui prend en paramètre la traduction partielle ainsi que la transformation qui vient tout juste de lui être appliquée. La valeur d'un coût est obtenue à l'aide de la méthode *value*.

Pendant le décodage, chaque traduction partielle est associée à un coût. Un coût doit donc être copié quand sa traduction partielle l'est. Cette copie est implémentée dans la méthode *clone*.

Il est mentionné à la section 2.1.2 qu'il y a souvent intérêt à recombiner les hypothèses étant dans le même état. Une des exigences requises pour qu'un ensemble d'hypothèses soient dans le même état est que leur coût le soient. C'est la méthode *compareState* qui s'occupe de faire cette comparaison. Le comportement de la méthode *compareState* est discuté en détails à la section 4.1.2.1.

Comme la méthode *update* doit habituellement connaître les détails des traductions partielles et des transformations pour se mettre à jour, l'implémentation d'un coût est

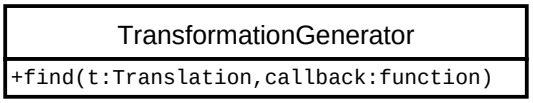

Figure 4.4 – Une classe représentant un générateur de transformations.

dépendante de l'implémentation de ces deux classes.

Un coût est soumis aux mêmes contraintes opérationnelles qu'une traduction partielle. Sa mise à jour et sa copie doivent donc se faire rapidement pour que le décodeur soit efficace.

4.1.1.4 Générateur de transformations

Pour faire évoluer une traduction partielle, le décodeur doit trouver les transformations qui lui sont applicables. C'est la tâche du générateur de transformations, qui est représenté dans MOOD à l'aide d'une classe semblable à celle présentée à la Figure 4.4.

La recherche des transformations applicables se fait à l'aide de la méthode *find*, qui prend en paramètres une traduction partielle et retourne, à l'aide de la fonction de rappel (*callback*), les transformations qui lui sont applicables. Chaque transformation trouvée est passée à la fonction de rappel, qui peut par exemple la mémoriser dans un tableau ou la traiter immédiatement. La fonction de rappel permet donc une gestion flexible des transformations retournées.

La méthode *find* analyse une traduction partielle et génère les transformations s'y appliquant. Un générateur de transformations dépend donc de l'implémentation de ces deux classes.

J'ai spécifié à la section 2.1.2 que l'état d'une traduction partielle dépend du générateur de transformations. Cependant, plutôt que d'ajouter une méthode *compareState* au générateur de transformations, je l'ai ajoutée à la traduction partielle. Cette décision a été prise à un moment où le formalisme du chapitre 2 ne m'était pas encore entièrement clair. À l'époque, je croyais que l'état d'une traduction partielle était composé des attributs de l'indicateur de progression. Je n'avais pas encore constaté qu'il dépend plutôt de l'usage que fait le générateur de transformations de ces attributs. En effet, les attributs ignorés par le générateur de transformations ne font pas partie de l'état.

J'ai tout de même décidé de laisser la méthode *compareState* dans la traduction par-

tielle, car son implémentation reste valide dans la plupart des scénarios. Il est aussi pratique de pouvoir comparer les états de deux traductions partielles sans avoir de référence vers le générateur de transformations. Quand ce comportement par défaut n'est pas approprié, une fonction de comparaison personnalisée peut être spécifiée.

4.1.2 Exploration de l'espace de recherche

La représentation du modèle permet de définir un espace de recherche implicite à l'aide d'une traduction initiale et d'un générateur de transformations. Comme cet espace de recherche est habituellement gigantesque, il ne peut être exploré en entier. Les traductions partielles associées aux coûts les plus bas doivent donc être élaguées.

Une technique pour explorer l'espace de recherche est définie à l'aide de deux classes. Une première implémentant une hypothèse (*Hypothesis*) et une deuxième implémentant une stratégie de recherche (*SearchStrategy*).

Pour que les techniques d'exploration de l'espace de recherche restent indépendantes des classes représentant le modèle, les classes définissant les hypothèses et les stratégies de recherche sont implémentée à l'aide de types génériques. Une fois écrites, elles pourront donc être utilisées avec toutes les représentations de modèles sans avoir à être copiées et modifiées.

4.1.2.1 Hypothèse

Le principal rôle d'une hypothèse est de synchroniser une traduction partielle et son coût pendant le décodage. Elle est implémentée dans MOOD à l'aide d'une classe semblable à celle présentée à la Figure 4.5.

Une hypothèse est composée de la traduction partielle et du coût qu'elle synchronise. Elle est transformée à l'aide de sa méthode *apply*, qui applique la transformation à la traduction partielle et met ensuite le coût à jour. La traduction partielle d'une hypothèse peut être accédée à l'aide de la méthode *translation* et la valeur de son coût à l'aide de la méthode *value*.

Quand plusieurs transformations sont applicables à une même hypothèse, la stratégie de recherche doit la cloner. La méthode *clone* retourne une copie de l'hypothèse sur laquelle elle a été appelée. L'hypothèse appelle à son tour les méthodes *clone* de sa

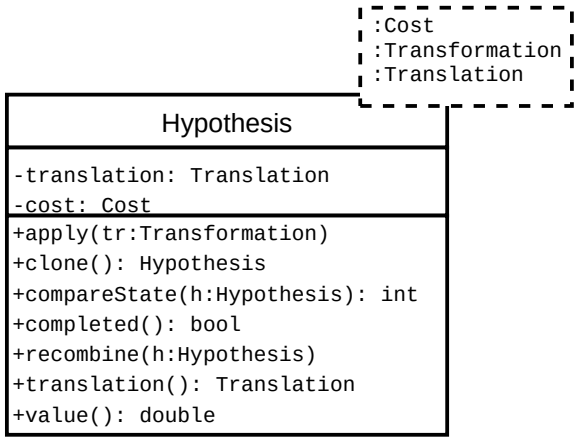

Figure 4.5 – Une classe représentant une hypothèse.

traduction partielle et de son coût.

Quand plusieurs hypothèses sont dans le même état, elles ont avantage à être recombinées (voir la section 2.1.2). C'est la tâche de la méthode *recombine*, qui s'assure que l'hypothèse sur laquelle elle est appelée contient le résultat de sa recombinaison avec l'hypothèse en paramètre.

Avant de recombiner deux hypothèses, il faut s'assurer qu'elles soient dans le même état. Cette vérification se fait à l'aide de la méthode *compareState*, qui pose un ordre total sur les états des hypothèses. La comparaison est faite entre l'état de l'hypothèse sur laquelle la méthode a été appelée ($h1$) et l'état de l'hypothèse passée en paramètre ($h2$). La valeur que *compareState* retourne est négative lorsque l'état de $h1$ est plus petit que l'état de $h2$, zéro si les deux hypothèses sont dans le même état et une valeur positive sinon. Cette méthode est implémentée en appelant les méthodes *compareState* de la traduction partielle et du coût, qui posent aussi un ordre total sur les états.

La façon dont l'ordre total sur les états est défini importe peu. Ce qui importe est qu'une fois qu'il est défini, cet ordre peut être utilisé pour recombiner les hypothèses d'une pile en $O(n \log n)$ (un tri sur les états suivi d'un parcours repérant les hypothèses consécutives ayant le même état) plutôt qu'en $O(n^2)$ (comparer toutes les hypothèses ensembles).

Une hypothèse, à la base, n'ajoute aucune fonctionnalité au décodeur. Elle ne fait que synchroniser une traduction partielle et son coût. Il n'y a cependant rien qui empêche

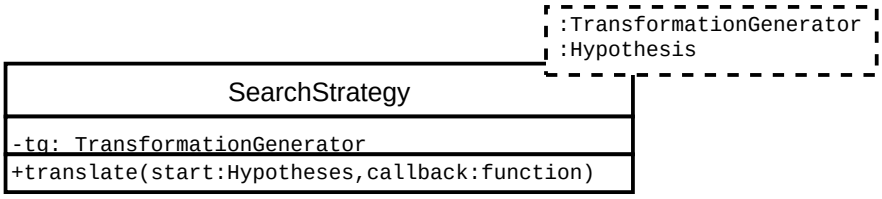

Figure 4.6 – Une classe représentant une stratégie de recherche.

une hypothèse d'en faire plus, comme par exemple mémoriser le graphe de recherche ou afficher une trace à l'écran.

Peu importe la représentation du modèle, une hypothèse fera toujours le même traitement. Pour exploiter cette indépendance, il est recommandé d'implémenter une hypothèse comme un type générique. Ainsi, elle pourra être utilisée avec toutes les représentations de modèle. Comme l'hypothèse contient une traduction partielle, un coût et qu'elle utilise des transformations, le type générique doit être paramétré sur ces trois classes.

4.1.2.2 Stratégie de recherche

La stratégie de recherche est le coeur du décodeur, car c'est elle qui s'occupe de traduire une phrase source. Pour y arriver, elle explore le graphe de recherche implicite défini par une hypothèse initiale et un générateur de transformations. Une stratégie de recherche est implémentée dans MOOD à l'aide d'une classe semblable à celle présentée à la Figure 4.6.

La traduction d'une phrase source est lancée par la méthode *translate*, qui a pour paramètres une hypothèse initiale et une fonction de rappel (*callback*). Quand une traduction complète est trouvée, la fonction de rappel est tout de suite appelée avec son hypothèse en paramètre. Plutôt que de ne retourner que la meilleure traduction, la stratégie de recherche retourne donc toutes les traductions complètes trouvées. Cela peut permettre, entre autres, d'offrir un ensemble de traductions candidates à un utilisateur ou d'arrêter la recherche aussitôt qu'une traduction jugée raisonnable est trouvée.

Pour faire évoluer une hypothèse, la stratégie de recherche nécessite un générateur de transformations. Ce générateur de transformation est habituellement spécifié à la création de la stratégie de recherche.

Tout comme l'hypothèse, peu importe la représentation du modèle, la stratégie de

recherche fera toujours le même traitement. Il est donc recommandé de l'implémenter à l'aide d'un type générique. En supposant que le type des transformations soit déduit à partir d'une hypothèse, le type générique d'une stratégie de recherche est paramétré sur le type des hypothèses qu'elle fait évoluer et le type du générateur de transformations qu'elle utilise.

Plusieurs stratégies de recherche ont été présentées au chapitre 2. Le passage d'une de ces stratégies de recherche vers une classe semblable à celle la Figure 4.6 est direct.

4.1.3 Résumé

Pour implémenter un décodeur avec MOOD, il suffit donc de créer les six classes présentées dans cette section et d'implémenter chacune des méthodes décrites.

MOOD est présentement distribué avec des classes permettant de représenter un modèle basé sur les segments de phrases et une stratégie de recherche en faisceau à plusieurs piles. Avec le temps, j'espère ajouter des classes permettant la représentation d'autres modèles et l'usage d'autres stratégies de recherche.

4.1.4 Choix technologiques

Le principal choix technologique fait lors de l'implémentation de MOOD a été celui du langage de programmation. Comme le langage utilisé devait engendrer des programmes rapides, offrir une gestion de la mémoire efficace pour la création massive de petits objets, permettre les types génériques, permettre la programmation orientée objet et être maîtrisé par les artisans de la traduction statistique, le choix de C++ s'est imposé.

MOOD est développé sous linux. Il utilise la bibliothèque de fonctions *boost*[1] pour l'ensemble d'outils généraux qu'elle offre (pointeurs intelligents, expressions régulières, structure de données pour représenter des graphes, ...) et la bibliothèque de fonctions SRILM (Stolcke, 2002) pour manipuler les modèles de langues.

4.2 Ramses

La section précédente décrivait en détail les six classes qu'il faut implémenter pour mettre en oeuvre un décodeur avec MOOD. Cette section présente comment elles ont été

[1] `http://www.boost.org`

merveilleux	wonderful	0.8 0.7
monde	world	0.6 0.5
quel	what a	0.9 0.4

Figure 4.7 – Un extrait d'une table de traduction.

implémentées pour créer un décodeur basé sur les segments de phrase.

Ce décodeur, nommé RAMSES, a été mis en oeuvre à partir des spécifications de PHARAOH (Koehn, 2004), qui a été présenté à la section 3.1.1. Ce décodeur a été choisi car il offre des performances suivant les règles de l'art et qu'il est décrit en détails dans le manuel qui l'accompagne.

RAMSES supporte la plupart des fonctionnalités offertes par PHARAOH. Il permet de traduire un document source, de faire une traduction en deux passages ou de sauvegarder le graphe de recherche parcouru pour chaque phrase. Le graphes de recherche d'une phrase source peut ensuite être utilisé par CARMEL, (Knight et Al-Onaizan, 1999), un transducteur à états finis, pour obtenir la liste de ses N meilleures traductions.

4.2.1 Table de traduction

Une table de traduction est spécifiée au démarrage du décodeur. Cette table agit comme un dictionnaire bilingue où chaque entrée associe un segment de phrase source à un segment de phrase cible et quantifie la qualité de ces associations par un ensemble de scores. La Figure 4.7 présente une table de traduction contenant trois règles traduisant du français vers l'anglais où chaque règle est évaluée par deux scores.

Dans RAMSES, une table de traduction est implémentée à l'aide d'une classe héritant de la classe *Indexer* (Figure 4.8) et chacune de ses entrées est appelée une règle (*Rule*). Une règle est composée d'un segment de phrase source, d'un segment de phrase cible et d'une séquence de scores quantifiant la qualité de la règle.

La tâche principale de la table de traduction est de retourner les règles permettant de traduire un segment de phrase source. Ces règles sont retournées par la méthode *find*, qui prend en paramètre une phrase source, la position et la longueur du segment pour lequel chercher les règles et une fonction de rappel pour les retourner.

Au démarrage de RAMSES, toutes les règles sont mémorisées dans un arbre de préfixes (*TrieIndexer*) (Knuth, 1998, section 6.3). Cela permet un accès efficace aux règles tra-

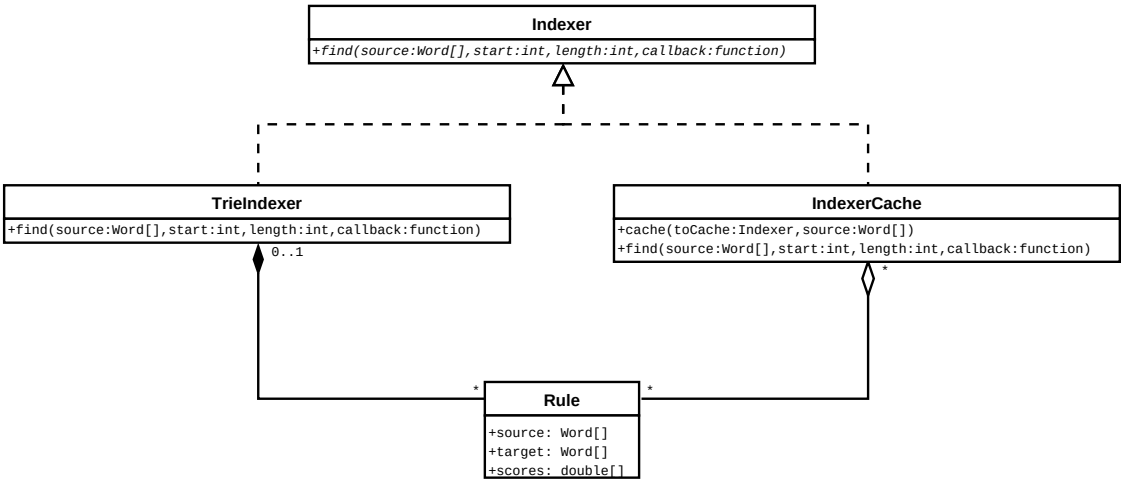

Figure 4.8 – La classe indexant les règles dans RAMSES.

duisant un segment de phrase source quelconque.

Avant que le décodage d'une phrase source s_1^m ne commence, RAMSES recherche toutes les règles pouvant lui être appliquées et les mémorise dans un tableau à deux dimensions. L'item à la position $\langle i, j \rangle$ de ce tableau contient les règles permettant de traduire le segment s_i^{i+j-1}. Une fois que ce tableau est construit, les règles traduisant un segment de la phrase source peuvent être retrouvées en temps constant.

Ce tableau est implémenté dans la classe *IndexerCache* et il est mis à jour lorsque le méthode *cache* est appelée. Cette méthode prend en paramètre la table de traduction contenant toutes les règles et la phrase source pour laquelle mémoriser les règles.

Cacher les détails de la table de traduction derrière l'interface *Indexer* permet beaucoup de flexibilité. Si d'autres implémentations recherchant par exemple les règles dans une base de données ou interrogeant un serveur central étaient implémentées, elles pourraient être utilisées sans modification du code existant.

4.2.2 Transformation

Une classe représentant une transformation dans RAMSES est présentée à la Figure 4.9. Une transformation est composée d'une règle (*Rule*) et de la position de départ du segment qu'elle traduit dans la phrase source.

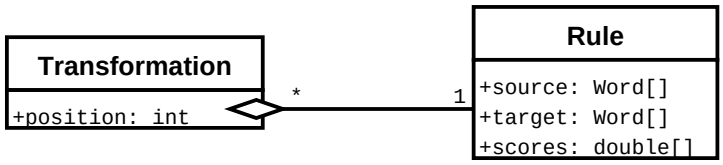

Figure 4.9 – La structure d'une transformation dans RAMSES.

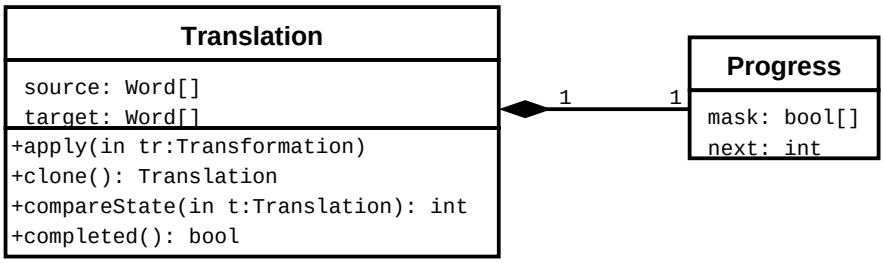

Figure 4.10 – La structure d'une traduction partielle dans RAMSES.

4.2.3 Traduction

Une classe représentant une traduction partielle dans RAMSES est présentée à la Figure 4.10. La phrase source et la phrase cible sont de simples séquences de mots. L'indicateur de progression est quant à lui composé d'un masque (*mask*) indiquant quels mots sont déjà traduits dans la phrase cible et de la position du prochain mot source à traduire pour que la traduction soit monotone (*next*). Le masque associe une valeur booléenne à chaque mot de la phrase source. Cette valeur est *vrai* si le mot qui lui correspond est déjà traduit dans la phrase cible.

Lorsqu'une traduction partielle est transformée, sa phrase cible et son indicateur de progression sont mis à jour. C'est ce que fait la méthode *apply*. Elle ajoute le segment de phrase cible de la règle à la phrase cible, met à jour le masque pour qu'il reflète les nouveaux mots sources traduits et met à jour *next* à la position du mot source suivant le dernier traduit. Une traduction est donc complète lorsque toutes les valeurs de son masque sont vraies. Quelques exemples d'applications de transformations sont présentées à la Figure 4.11.

Le générateur de transformations, détaillé à la section 4.2.5, utilise le masque et la position *next* de l'indicateur de progression pour déterminer les transformations applicables à une traduction partielle. La méthode *compareState* utilise donc ces deux attributs pour

	Traduction partielle initiale
source	*quel monde merveilleux*
progression	masque=`000`, prochain=1
cible	
coûts	distorsion=0, mots=0, score1=log(1), score2=log(1)
	Transformation 1
règle	source=*quel*, cible=*what a*, scores=$\langle 0.9, 0.4 \rangle$
position	1
	Traduction partielle 2
source	*quel monde merveilleux*
progression	masque=`100`, prochain=2
cible	*what a*
coûts	distortion=$0 + \lvert 1 - 1 \rvert$, mots=$0 + 2$, score1=$\log(1 * 0.9)$, score2=$\log(1 * 0.4)$
	Transformation 2
règle	source=*merveilleux*, cible=*wonderful* scores=$\langle 0.8, 0.7 \rangle$
position	3
	Traduction partielle 2
source	*quel monde merveilleux*
progression	masque=`101`, prochain=4
cible	*what a wonderful*
coûts	distortion=$0 + \lvert 3 - 2 \rvert$, mots=$2 + 1$, score1=$\log(0.9 * 0.8)$, score2=$\log(0.4 * 0.7)$
	Transformation 3
règle	source=*monde*, cible=*world*, scores=$\langle 0.6, 0.5 \rangle$
position	2
	Traduction complète
source	*quel monde merveilleux*
progression	masque=`111`, prochain=3
cible	*what a wonderful world*
coûts	distortion=3, mots=4, score1=log(0.432), score2=log(0.14)

Figure 4.11 – Un exemple d'application de transformations à une traduction partielle.

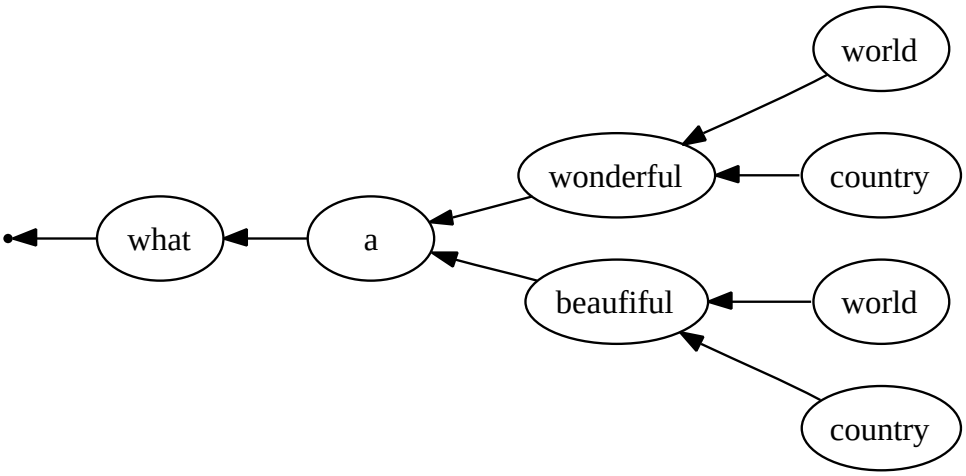

Figure 4.12 – Quatre phrases cibles partageant leurs préfixes communs à l'aide d'un arbre.

imposer un ordre total aux états des traductions partielles.

Lorsque la méthode *clone* est appelée, elle doit copier la phrase source, la phrase cible et l'indicateur de progression dans une nouvelle traduction partielle. Certaines précautions ont été prises pour que ce clonage soit efficace. La première est de partager la même phrase source entre une traduction partielle et ses clones. La copie de la phrase source ne se résume donc qu'à la copie d'une référence.

De plus, aucune transformation ne permet de modifier les mots ajoutés à une phrase cible. Une traduction partielle et ses clones peuvent donc partager leur préfixe à l'aide d'un arbre semblable à celui présenté à la Figure 4.12. L'arbre donné en exemple permet aux phrases *what a wonderful world*, *what a wonderful country*, *what a beautiful world* et *what a beautiful country* de partager leurs préfixes communs. La copie d'une phrase cible se résume donc à copier une référence vers le noeud de son dernier mot. La phrase cible peut être reconstruite en remontant l'arbre jusqu'à sa racine.

4.2.4 Coût

Le coût assigne un score à une traduction partielle. La mise en oeuvre du coût utilisé dans RAMSES peut se faire directement à partir de la description donnée à la section 3.1.1.4. Pour garder les coûts modulaires, chaque score est implémenté dans sa propre classe suivant l'interface *Cost* et tous ces coûts sont synthétisés dans un coût

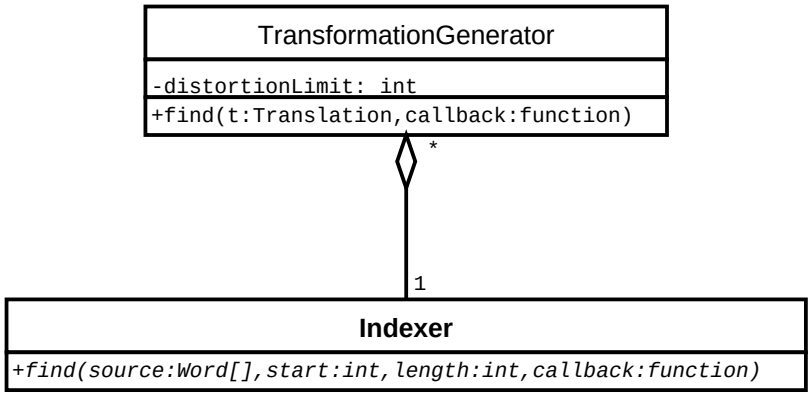

Figure 4.13 – Le générateur de transformations de RAMSES.

implémentant la somme pondérée.

Afin de garder le clonage rapide, les structures de données partagées entre un coût et ses clones sont partagées et non copiées. C'est le cas pour la structure mémorisant le modèle de langue et pour la table qui est utilisée dans la fonction heuristique.

4.2.5 Générateur de transformations

La classe représentant un générateur de transformations dans RAMSES est présentée à la Figure 4.13. Lorsqu'un générateur de transformations est initialisé, il reçoit en paramètre une table de traduction (*Indexer*) et un paramètre permettant de limiter le nombre de mots source séparant deux transformations consécutives (*distortionLimit*).

Les transformations applicables à une traduction partielle sont retournées par la méthode *find*, qui implémente l'algorithme suivant :

1. Utiliser le masque de l'indicateur de progression pour trouver les segments de la phrase source n'étant pas encore traduits.
2. Limiter le réordonnancement en ne considérant que les segments dont la position de départ n'est pas à plus de *distortionLimit* mots du mot suivant la dernière transformation appliquée. La position du mot suivant la dernière transformation appliquée est mémorisée dans l'attribut *next* de l'indicateur de progression.
3. Utiliser la table de traduction pour trouver les règles permettant de traduire les segments de phrase considérés.

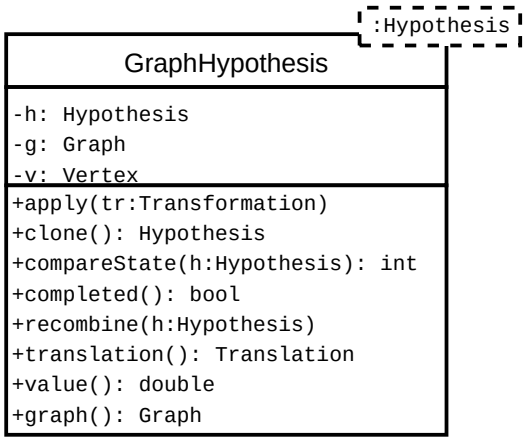

Figure 4.14 – Une hypothèse mémorisant le graphe de recherche.

4. Créer une transformation pour chaque règle trouvée et la retourner en appelant la fonction de rappel (*callback*).

4.2.6 Hypothèse

Lorsque seule la meilleure traduction d'une phrase source est demandée, l'hypothèse décrite à la section 4.1.2.1 est suffisante. Si le graphe recherché pendant le décodage est nécessaire, elle ne suffit malheureusement pas. Une classe implémentant une hypothèse qui mémorise le graphe de recherche est présenté à la Figure 4.14. Le graphe de recherche mémorisé est décrit en détails à la section 2.1.

Cette nouvelle hypothèse est composée d'une hypothèse se chargeant de la synchronisation entre la traduction partielle et le coût (h), d'une structure de donnée pour représenter le graphe (g) et d'une référence vers le sommet du graphe correspondant à l'hypothèse (v). Afin que le graphe puisse servir à construire une liste des N meilleures traductions, chacun de ses arcs est associé à une transformation et à la valeur qu'il ajoute au coût.

Chaque appel de méthode est propagé à l'hypothèse synchronisant la traduction partielle et le coût. Cette section ne fait que décrire les traitements ajoutés par la classe *GraphHypothesis*.

Quand la méthode *apply* est appelée, l'hypothèse est associée à un nouveau sommet et un arc entre ce sommet et l'ancien est ajouté.

Si deux hypothèses sont dans le même état, cela signifie que leurs sommets sont équivalents. La méthode *recombine* doit donc les fusionner. Cette fusion consiste à rediriger les arcs de l'hypothèse passée en paramètre vers l'hypothèse appelante.

Finalement, pour que le tout fonctionne, il faut qu'une hypothèse et ses clones partagent le même graphe de recherche. Une fois le décodage terminé, le graphe recherché peut être obtenu à l'aide de la méthode *graph*.

Afin de garder cette hypothèse indépendante du modèle et de la stratégie de recherche, elle a été mise en oeuvre à l'aide d'un type générique. En supposant que le type des transformations puisse être déduit à partir du type d'une hypothèse, la classe n'est paramétrée que sur le type de l'hypothèse utilisée pour synchroniser la traduction partielle et le coût.

4.2.7 Stratégie de recherche

La stratégie de recherche utilisée par RAMSES est une recherche en faisceau à plusieurs piles similaire à celle présentée à la section 2.2.4. L'implémentation de sa méthode *translate* est présentée à l'Algorithme 4.1,

Algorithme 4.1 Recherche en faisceau utilisée par RAMSES.

1. initialiser les n piles $\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_n$ qui seront nécessaires pendant la recherche
2. ajouter l'hypothèse initiale dans la pile de son groupe
3. pour $i \in [1, n]$
   (a) appliquer la politique d'élagage à la pile $\mathcal{H}_i$
   (b) pour chaque $h \in \mathcal{H}_i$
      i. pour chaque transformation tr que le générateur de transformations retourne pour $h.translation()$
         A. $copie \leftarrow h.clone()$
         B. $copie.apply(tr)$
         C. si $copie.completed()$ est vrai, alors passer *copie* en paramètre à la fonction de rappel
         D. si $copie.completed()$ est faux, alors mémoriser *copie* dans la pile de son groupe

Lorsqu'elle est créée, la stratégie de recherche en faisceau à plusieurs piles reçoit en paramètre un générateur de transformations, une politique de gestion des piles et une politique d'élagage.

La politique de gestion des piles est implantée à l'aide de deux fonctions prenant chacune en paramètre une traduction partielle. La première fonction retourne le nombre de piles nécessaires pour le décodage (utilisée à la ligne 1) et la deuxième l'indice de la pile dans laquelle mémoriser une hypothèse (utilisée aux lignes 2 et D).

La politique de gestion des piles utilisée par RAMSES réserve autant de piles qu'il y a de mots dans la phrase source et elle mémorise une hypothèse dans la pile $j + 1$ si j mots sources y sont traduits.

Le choix d'une pile est dépendant des détails de l'implémentation de la traduction partielle. Encapsuler ce choix dans une politique permet de conserver l'indépendance entre la traduction partielle et la recherche en faisceau à plusieurs piles. Cette stratégie de recherche peut donc être utilisée avec un autre modèle à la condition de définir une nouvelle politique de gestion des piles.

La politique d'élagage est implémentée avec une fonction qui élague la pile qu'elle reçoit en paramètre. RAMSES utilise la politique d'élagage suivante :

1. Trier les hypothèses par ordre décroissant de coût.
2. Appliquer un élagage par histogramme.
3. Appliquer un élagage par seuil relatif. Comme les coûts peuvent être négatifs, cet élagage utilise la valeur exponentielle des coûts.
4. Trier les hypothèses par ordre d'état.
5. Parcourir la pile et recombiner les hypothèses consécutives ayant le même état.

L'élagage par histogramme et l'élagage par seuil relatif sont décrits à la section 2.2.4.

Lorsque j'ai examiné le comportement de RAMSES pour connaître les opérations étant les plus coûteuses à exécuter, j'ai été surpris de constater que l'élagage arrivait en tête de liste. J'ai réussi à diminuer d'au moins de moitié le temps d'exécution du décodeur en apportant deux petites modifications à la politique d'élagage.

La première est de ne faire qu'un tri partiel des piles. En effet, si l'élagage par histogramme ne garde que les 100 meilleures hypothèses, il est inutile de trier la pile complète (qui en contient habituellement des dizaines de milliers). Le premier gain a donc été obtenu en remplaçant un appel à la fonction *sort* de C++ par un appel à la fonction *partial_sort*. Les deux fonctions utilisent l'algorithme de tri rapide (*quicksort*) (Cormen

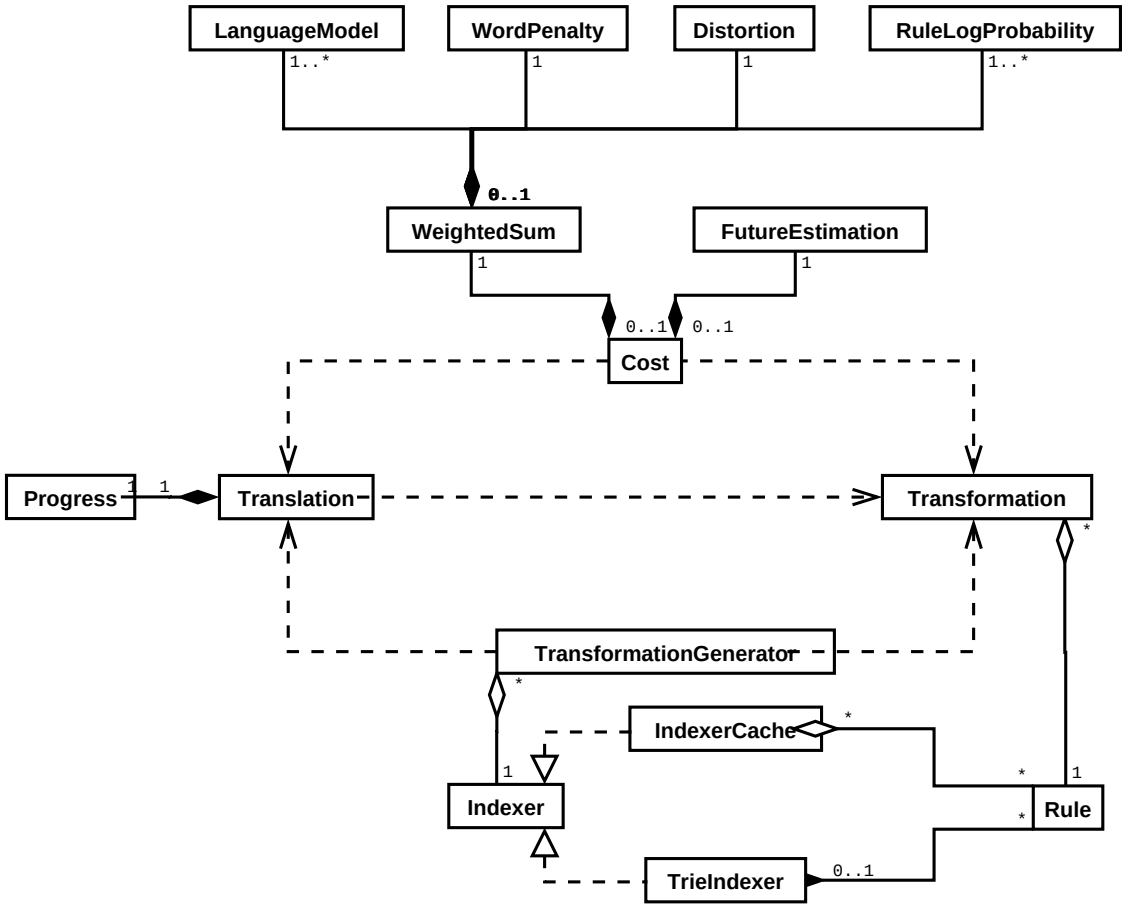

Figure 4.15 – Diagramme UML donnant une vue d'ensemble de la représentation du modèle utilisé par RAMSES.

et al., 1994, chapitre 8), mais la fonction *partial_sort* ne trie une partition qui si elle contient au moins un des n meilleurs éléments, où n est un paramètre de la fonction.

La deuxième modification a été de recombiner les hypothèses après que la pile ait été élaguée. Il est en effet beaucoup moins long de recombiner 100 hypothèses que des dizaines de milliers.

4.2.8 Donner vie au décodeur

Une vue d'ensemble de toutes les classes du modèle définies dans cette section est présentée dans la Figure 4.15. Ces classes sont maintenant intégrées à MOOD et RAMSES ne fait que les utiliser à l'aide de l'Algorithme 4.2.

Pour obtenir le graphe de recherche exploré, il ne suffit que de modifier la ligne 5c

Algorithme 4.2 Algorithme utilisé par RAMSES pour traduire un document.

1. Charger la table de traduction dans un objet de type *TrieIndexer* et nommer cet objet *ti*.
2. Créer un objet de type *IndexerCache* et le nommer *ci*.
3. Initialiser le générateur de transformations en lui passant *ci* en paramètre.
4. Initialiser la stratégie de recherche en faisceau en lui passant en paramètre le générateur de transformation.
5. Pour chaque phrase source *s* à traduire
   (a) Créer une traduction partielle initiale pour *s*. La phrase cible initiale ne contient aucun mot et un indicateur de progression initial est composé d'un masque ne contenant que des valeurs fausses et d'une position *next* valant un.
   (b) Créer un coût inital.
   (c) Créer une hypothèse faite de la traduction et du coût initial.
   (d) Appeler la méthode *cache* de *ic* en lui passant en paramètre *ti* et *s*.
   (e) Afficher la meilleure traduction trouvée par la méthode *decode* de la stratégie de recherche.

pour qu'elle crée une hypothèse initiale de type *GraphHypothesis* et d'afficher le graphe plutôt que la meilleure traduction trouvée à la ligne 5e.

4.3 Architecture orientée objet à l'oeuvre

Une des grandes promesses de la programmation orientée objet est la réutilisation des différentes classes déjà implémentées. Cette section présente comment l'architecture orientée objet de MOOD remplit cette promesse lors de la mise en oeuvre d'un nouveau décodeur.

Créer un décodeur ne se différenciant de RAMSES que par son coût ne demande que d'écrire une nouvelle classe implémentant ce coût. En effet, aucune classe du décodeur ne dépend d'un coût autrement que par son interface. Donc, sur les six classes définissant un décodeur, ce nouveau décodeur pourra en réutiliser cinq.

Lorsqu'un nouveau coût doit être utilisé, l'ajout d'une nouvelle classe est préférable à la modification d'une classe existante. En ajoutant une nouvelle classe, l'ancien et le nouveau décodeur peuvent facilement coexister.

Comme aucune classe ne dépend du générateur de transformations autrement que par son interface, un décodeur avec un nouveau générateur de transformations peu aussi

réutiliser cinq des six classes. Maintenir un ensemble de décodeurs ne se différenciant que par des générateurs de transformations spécialisés pour certaines langues devient donc aisé, car tous ces décodeurs partageront une grande partie de leurs classes.

Il est intéressant de noter que si un nouveau coût et un nouveau générateur de transformations sont implémentés, non pas trois, mais quatre décodeurs peuvent maintenant être créés. En effet, ces deux modules sont indépendants, ils peuvent donc être combinés à volonté.

Il a déjà été souligné que les hypothèses et les stratégies de recherche sont indépendantes de la représentation du modèle. Un décodeur pour un nouveau modèle ne nécessite donc que d'écrire les quatre classes servant à la représentation du modèle.

Les hypothèses sont aussi indépendantes de la stratégie de recherche. L'hypothèse mémorisant le graphe de recherche peut donc être utilisée avec n'importe quelle stratégie de recherche.

En résumé, l'architecture de MOOD permet une grande réutilisation du code existant. Cela entraîne deux principaux avantages. Le premier est bien entendu de réduire le temps de développement d'un nouveau décodeur. Le deuxième, qui n'est pas négligeable non plus, est de réduire les opportunités d'introduire des bogues dans le décodeur. En effet, plus le code est utilisé, plus ses problèmes risquent d'être découverts et réglés.

4.4 Un décodeur libre

Ma plus grande ambition en développant MOOD est de créer un lieu commun où les artisans de la traduction statistique pourront partager leur expertise en décodage. Cette section décrit les différentes décisions qui ont été prises en ce sens.

La décision la plus importante pour encourager le partage des connaissances est certainement de distribuer MOOD et RAMSES sous les termes de la licence *Gnu General Public License*[2] (GPL). Brièvement, cette licence permet à quiconque de télécharger, d'utiliser, de modifier et de distribuer le code source de MOOD, mais à deux conditions. Si une version modifiée de MOOD est distribuée, elle doit l'être sous les termes de la licence GPL et si un programme utilisant MOOD est distribué, il doit lui aussi être distribué sous les termes de la licence GPL. Cela signifie qu'un chercheur ou une entreprise voulant profiter

[2] `http://www.gnu.org/licenses/gpl.html`

de MOOD est encouragé à partager le fruit de ses efforts avec la communauté.

MOOD est hébergé par *SourceForge*[3]. Cet hébergeur offre gratuitement une suite d'outils facilitant le développement distribué. Parmi ces outils, on peut compter un système de contrôle de versions du code source, un système pour signaler les bogues et des listes de diffusions. Avec ces ressources, il est plus facile de participer au développement de MOOD.

Permettre à tout le monde de participer au développement de MOOD amène plusieurs avantages dont les décodeurs distribués seulement sous leur forme binaire ne peuvent bénéficier. Premièrement, le code sera vu par un plus grand nombre de personnes. Les bogues risquent donc d'être découverts plus vite. Un autre est d'offrir la chance à tout le monde d'apporter une petite contribution (par exemple, développer un nouveau coût) ou une plus grande (par exemple, supporter un nouveau modèle).

Une dernière action entreprise pour faciliter le partage de l'information est la mise en ligne d'un wiki. Un *wiki* est un site web pouvant être modifié librement par tout le monde. La documentation de MOOD sera diffusée sur ce wiki et pourra donc constamment être améliorée et maintenue par l'ensemble des utilisateurs de MOOD.

Offrir ces ressources ne rendra pas automatiquement le projet populaire et ne mobilisera pas automatiquement une communauté autour de MOOD, mais c'est définitivement un pas dans la bonne direction. MOOD et RAMSES peuvent être téléchargés à l'adresse `http://smtmood.sourceforge.net`.

4.5 Conclusion

Ce chapitre a présenté MOOD, un cadre d'applications facilitant le développement de décodeurs. Ce cadre d'applications propose une architecture modulaire et orientée objet permettant un développement rapide de nouveaux décodeurs. Afin de stimuler la recherche (Walker, 2005), MOOD est distribué sous la licence GPL, donnant ainsi un accès libre à son code source.

Pour montrer comment MOOD peut être utilisé pour le développement d'un décodeur suivant les règles de l'art, je l'ai utilisé pour mettre en oeuvre RAMSES, qui implémente les algorithmes PHARAOH (Koehn, 2004). Pour vérifier si RAMSES est à la hauteur de

[3] `http://www.sourceforge.net`

PHARAOH, une évaluation comparative de ces deux décodeurs est présentée au chapitre 5.

CHAPITRE 5

EXPÉRIENCES

Dans ce chapitre, nous présentons les expériences que nous avons menées pour comparer RAMSES à PHARAOH (Koehn, 2004), un décodeur de référence dans la littérature. Nous vérifions aussi où RAMSES se situe par rapport à d'autres décodeurs suivant les règles de l'art.

La première section de ce chapitre présente la tâche qui a été choisie pour mener cette évaluation. Les deux sections suivantes comparent en détails la qualité des traductions produites par RAMSES à celles produites par PHARAOH et la vitesse des deux décodeurs. La dernière section présente les résultats que RAMSES a obtenus lors de notre participation à la tâche partagée organisée dans le cadre de l'édition de 2006 de l'atelier *Workshop on Machine Translation* (WMT).

5.1 Description de la tâche

Nous avons évalué RAMSES sur les six tâches de traduction mises en place dans le cadre de la tâche partagée de l'atelier WMT. Pour les trois premières tâches, la langue cible était l'anglais et les langues sources étaient l'allemand, l'espagnol et le français, alors que dans les trois autres tâches, les langues sources et cibles étaient inversées.

Une description détaillée de la tâche partagée ainsi qu'un inventaire des ressources étant à la disposition des participants peuvent être trouvés à l'adresse `http://www.statmt.org/wmt06/`.

5.1.1 Corpus

Pour chaque paire de langues à traduire, quatre corpus ont été mis à notre disposition. Ces corpus sont tous des sections du corpus EUROPARL (Koehn, 2005), qui est composé de transcriptions de débats parlementaires européens.

Le premier corpus (DEV) est composé de plus d'un million de paires de phrases et est dédié à la construction de la table de traduction et à l'entraînement du modèle de langue. Le deuxième corpus (TUNE) est composé de 2000 paires de phrases et sert à la

configuration des différents paramètres du décodeur. Le troisième corpus (TEST) est aussi composé de 2000 paires de phrases, mais il sert à tester une configuration particulière de notre décodeur. Finalement, le dernier corpus (FINAL) est composé de 3064 phrases et est celui sur lequel les systèmes des différents participants ont été évalués.

5.1.2 Tables de traduction

Pour chaque paire de langues entre lesquelles traduire, le corpus DEV a été utilisé pour générer une table de traduction et pour entraîner un modèle de langue.

Les modèles de langue ont été entraînés à l'aide de SRILM (Stolcke, 2002) et les tables de traduction ont été obtenues à l'aide du script *train-phrase-model.perl*, rendu disponible par les organisateurs de la tâche partagée.

Ce script extrait un ensemble de paires de phrases à partir de deux textes parallèles ayant été alignés au niveau des mots par le programme GIZA++. Cinq coûts sont ensuite associés à chacun des segments de phrases obtenus. Ces coûts, qui sont calculés à partir de fréquences relatives et de probabilités sur les alignements de mots, sont décrits en détails dans le manuel accompagnant le script.

Ce qui importe pour comparer RAMSES à PHARAOH n'est pas la manière de générer les tables de traduction et les modèles de langues, mais plutôt d'utiliser les mêmes avec les deux décodeurs.

5.1.3 Ajustement des poids

RAMSES et PHARAOH cherchent à maximiser une somme pondérée semblable à celle de l'équation 3.5 où il y a huit poids à configurer : le poids de chacun des cinq coûts de la table de traduction, le poids du modèle de langue, le poids associé à la pénalité sur le réordonnancement des segments de phrases et le poids associé à la pénalité sur le nombre de mots dans la phrase cible.

Le choix de ces huit poids est critique pour que le décodeur produise des traductions de qualité. Nous les avons donc ajustés avec le script *minimum-error-rate-training.perl*, qui est distribué par les organisateurs de l'atelier. Ce script utilise l'algorithme décrit par Och (2003) pour chercher les poids minimisant le score BLEU (décrit à la section 5.2.1) obtenu en traduisant les 500 premières phrases du corpus TUNE.

Le script commence par initialiser les huit poids à des valeurs aléatoires et itère ensuite en modifiant un seul poids à la fois. À chaque itération, la modification du poids apportant le meilleur gain en score BLEU est apportée. La recherche se termine lorsqu'aucun poids ne peut être modifié pour améliorer le score BLEU. Comme la configuration résultante est sûrement un optimum local, cette recherche est relancée 20 fois avec des poids initiaux différents et la meilleure de ces 20 configurations est gardée.

Pour ne pas avoir à relancer le décodage chaque fois qu'un poids est mis à jour, seulement la liste des 100 meilleures traductions de chaque phrase source est considérée. Quand un poids change, plutôt que de relancer le décodage, la traduction de chaque phrase source est mise à jour à la phrase cible de sa liste ayant le coût le plus élevé avec les nouveaux poids. Och (2003) présente un algorithme efficace pour trouver la modification de poids apportant la meilleure amélioration en score BLEU lorsque ces listes sont utilisées.

Une fois une configuration optimale des poids atteinte, la liste des 100 meilleures traductions de chaque phrase source est générée une nouvelle fois par le décodeur. Si des phrases cibles des anciennes listes ne figurent pas dans les nouvelles listes, elles y sont ajoutées et la recherche des meilleurs poids est relancée avec ces nouvelles listes enrichies. Si les listes contiennent toutes les phrases cibles des listes précédentes, la recherche est terminée et les huit poids optimums sont retournés.

Seulement les 500 premières phrases du corpus TUNE sont utilisées parce que l'optimisation des poids est coûteuse. Pour chaque paire de langues, le script relançait la recherche avec de nouvelles listes entre 10 et 20 fois et son exécution pouvait nécessiter quelques jours de calcul.

5.1.4 Comparaison entre Ramses et Pharaoh

Les mêmes modèles de langue et les mêmes tables de traduction ont été utilisés lors de la comparaison entre RAMSES et PHARAOH. Après quelques observations, nous nous sommes aperçus qu'à configurations égales, les sorties de RAMSES et de PHARAOH étaient parfois différentes. Afin de ne pas favoriser un décodeur par rapport à l'autre, nous avons optimisé les poids des deux décodeurs de manière indépendante.

Nous avons cependant observé que les poids optimaux calculés pour PHARAOH peuvent être utilisés avec RAMSES sans subir une grande perte de qualité sur les traductions pro-

duites.

Le corpus TEST a été utilisé pour comparer RAMSES à PHARAOH, car nous n'avions pas accès aux traductions de références du corpus FINAL. Les résultats rapportés pour la tâche partagée organisée dans le cadre de l'atelier WMT ont cependant été obtenus sur le corpus FINAL.

De plus, les mêmes paramètres d'élagages ont été utilisés pour les deux décodeurs. Soient un élagage par histogramme où seulement les 100 meilleures hypothèses de chaque pile sont conservées et un élagage par seuil relatif où le seuil est fixé à 0.03. Aucune contrainte n'a été imposée sur le réordonnancement des segments de phrases.

5.2 Qualité des traductions

La qualité des traductions produites par les des deux décodeurs a été comparée à l'aide d'une évaluation automatique et d'une évaluation manuelle. L'évaluation automatique a les avantages d'être rapide à calculer, objective et d'offrir des points de repères connus aux pratiquants de la traduction statistique. Elle ne fait cependant qu'une évaluation en surface des traductions, sans aucune analyse sémantique.

C'est pourquoi nous avons aussi mené une évaluation manuelle, où deux sujets ont comparé la qualité des traductions produites par les décodeurs. Les évaluations manuelles ont l'avantage de comparer les traductions en profondeur. Cependant, comme elles sont produites par des humains, elles sont subjectives et coûteuses à mener.

5.2.1 Évaluation automatique

L'évaluation automatique de la qualité des traductions produites par RAMSES et par PHARAOH s'est faite à l'aide de trois métriques : le taux d'erreur au niveau des mots, le taux d'erreur au niveau des phrases et le score BLEU. Ces trois métriques opèrent toutes sous le même paradigme. Elles vérifient si la traduction produite par le décodeur (document candidat) est similaire à une traduction qui a été produite par un humain (document de référence).

Le *taux d'erreur au niveau des mots* (WER) est le nombre minimum d'opérations qui sont nécessaires pour transformer une phrase candidate en la phrase de référence, normalisé sur le nombre de mots dans la phrase de référence. Les opérations permises

sont l'ajout, le remplacement et la suppression d'un mot. C'est la moyenne du WER de toutes les phrases du document candidat qui est rapportée. Nous la multiplions par cent pour en améliorer la lisibilité. Le WER prend donc une valeur entre 0 et 100 où des scores bas sont préférables.

Le *taux d'erreur au niveau des phrases* est le ratio des phrases candidates étant différentes des phrases de référence. Les SER rapportés sont multipliés par 100 pour améliorer leur lisibilité. Un SER de 0 signifie donc que toutes les phrases candidates sont identiques aux phrases de référence et un SER de 100 qu'elles sont toutes différentes.

La métrique BLEU (Papineni et al., 2002) calcule une valeur synthétisant des mesures de précisions pour des n-grams (séquences de mots de taille fixe) de différentes longueurs (habituellement entre un et quatre mots) et une pénalité si le document candidat contient moins de mots que le document de référence (BP). Sa valeur est calculée à l'aide de l'équation suivante :

$$\begin{aligned} \text{BLEU} &= \text{BP} \cdot \exp(\sum_{n=1}^{4} \frac{1}{4} \log p_n) \\ \text{BP} &= \begin{cases} 1 & \text{si } c \leq r \\ \exp(1 - r/c) & \text{si } c > r \end{cases} \end{aligned} \tag{5.1}$$

où c est le nombre de mots dans le document candidat, r le nombre de mots dans le document de référence et p_i est le ratio des segments de phrases du document candidat étant présents dans leur phrase correspondante dans le document de référence.

Le score BLEU prend une valeur entre 0 et 1 où les grandes valeurs sont préférables aux petites. Les valeurs rapportées sont multipliées par 100 et ont été calculées à l'aide du script *multi-bleu.perl*, rendu disponible par les organisateurs de l'atelier.

Le score BLEU corrèle avec les jugements humains (Papineni et al., 2002), mais il n'est pas facile à interpréter. C'est pourquoi les taux d'erreurs au niveau des phrases et au niveau des mots sont aussi rapportés. Ces taux d'erreurs sont souvent trop stricts pour permettre de comparer deux décodeurs en détails. Ils sont cependant plus faciles à interpréter.

Décodeur	WER	SER	BLEU	p_1	p_2	p_3	p_4	BP
allemand → anglais								
PHARAOH	59.60	97.65	*25.15*	61.19	31.32	18.53	11.61	0.99
RAMSES	61.05	97.70	24.49	61.06	30.75	17.73	10.81	1.00
espagnol → anglais								
PHARAOH	50.71	97.65	30.65	64.10	36.52	23.70	15.91	1.00
RAMSES	50.96	97.60	30.48	64.08	36.30	23.52	15.76	1.00
français → anglais								
PHARAOH	51.04	97.55	30.42	64.28	36.45	23.39	15.64	1.00
RAMSES	51.01	97.70	30.43	64.58	36.59	23.54	15.73	0.99
anglais → allemand								
PHARAOH	61.98	98.15	18.03	52.77	22.70	12.45	7.25	0.99
RAMSES	62.18	98.30	18.14	53.38	23.15	12.75	7.47	0.98
anglais → espagnol								
PHARAOH	54.77	97.75	29.40	61.86	35.32	22.77	15.02	1.00
RAMSES	55.74	97.65	28.75	62.23	35.03	22.32	14.58	0.99
anglais → français								
PHARAOH	54.71	97.85	30.96	61.10	36.56	24.49	16.80	1.00
RAMSES	53.21	97.85	*31.79*	61.57	37.38	25.30	17.53	1.00

Tableau 5.1 – Évaluation automatique de la qualité des traductions produites par RAMSES et par PHARAOH. Les différences statistiquement significatives sont en italiques.

5.2.1.1 Résultats

Les résultats de l'évaluation automatique de la qualité des traductions produites par RAMSES et par PHARAOH sur le corpus TEST sont présentés dans le Tableau 5.1. Afin de mieux comprendre le score BLEU, la précision pour les segments de phrase de longueur i est présentée dans la colonne p_i et la pénalité pour les documents trop court dans la colonne BP.

Les scores obtenus par les deux décodeurs ne diffèrent que de très peu. Pour vérifier si les différences obtenues sur le score BLEU sont significatives, nous avons utilisé la technique de *bootstrapping* présentée par Zang et Vogel (2004). En utilisant un intervalle de confiance de 95%, seulement deux de ces différences sont statistiquement significatives. La première avantage PHARAOH pour la traduction de l'allemand vers l'anglais et la

Décodeur	[0,15]	[16,25]	[26,∞[
anglais → français			
PHARAOH	33.52	30.65	30.39
RAMSES	33.78	31.19	31.35
allemand → anglais			
PHARAOH	29.74	24.30	24.76
RAMSES	29.85	23.92	23.78
espagnol → anglais			
PHARAOH	34.23	28.32	30.60
RAMSES	34.46	28.39	30.40

Tableau 5.2 – Comparaison des scores BLEU pour des phrases de longueurs différentes.

deuxième avantage RAMSES pour la traduction de l'anglais vers le français.

Afin de vérifier plus en détails ces différences, nous avons séparé le corpus TEST de quelques paires de langues en trois sections conditionnées sur la longueur de leurs phrases. Les résultats sont présentés dans le Tableau 5.2. Nous remarquons que les performances des deux décodeurs sont très similaires pour les phrases courtes (moins de 16 mots). Cependant les score BLEU peuvent diverger de près de 1% sur les phrases plus longues.

Les résultats présentés jusqu'à présent nous indiquent que RAMSES et PHARAOH obtiennent des traductions de qualités équivalentes selon le score BLEU. Cependant, ils ne disent rien sur la ressemblance de ces traductions. Pour mesurer cette ressemblance, nous avons relancé les mêmes expériences, mais cette fois-ci en prenant la sortie de PHARAOH comme traduction de référence. Les résultats obtenus sont présentés dans le Tableau 5.3.

Les traductions produites par RAMSES et PHARAOH pour les traductions de l'anglais vers le français, du français vers l'anglais et de l'espagnol vers l'anglais sont celles ayant obtenues les meilleurs scores BLEU. Elles sont aussi celles partageant le plus grand nombre de séquences de mots (valeurs de p_i dans le Tableau 5.3). Étant donné que ce sont les traductions ressemblant le plus au document de référence, on pouvait s'y attendre. Cependant, les traductions produites par les deux décodeurs partagent plus de 70% de leurs séquences de quatre mots, alors qu'elles en partagent moins de 20% avec le document de référence. Cet écart s'explique sûrement par le modèle qui assigne un bien meilleur coût à certaines transformations par rapport aux autre, ou en d'autres mots qui est plus

Décodeur	WER	SER	BLEU	p_1	p_2	p_3	p_4	BP
allemand → anglais								
RAMSES	24.47	77.75	69.09	90.2	73.5	62.9	54.7	1.00
espagnol → anglais								
RAMSES	8.43	49.70	87.19	96.2	88.9	84.2	80.3	1.00
français → anglais								
RAMSES	12.05	65.65	81.93	94.4	84.7	78.5	73.3	0.99
anglais → allemand								
RAMSES	22.69	84.75	65.39	85.9	70.0	60.7	53.2	0.98
anglais → espagnol								
RAMSES	22.36	82.00	70.43	89.5	75.1	65.9	58.3	0.99
anglais → français								
RAMSES	15.05	70.75	80.37	92.6	83.1	76.6	71.2	1.00

Tableau 5.3 – Scores obtenus par RAMSES lorsqu'il est comparé à PHARAOH plutôt qu'au document de référence.

sûr de lui.

De leur côté, les traductions entre l'anglais et l'allemand sont celles ayant les pires scores BLEU et sont celles partageant le moins grand nombre de mots ou de séquences de mots. Cela signifie sûrement que le modèle assignait des coûts semblables à plusieurs transformations différentes, amenant ainsi à des traductions plus aléatoires.

Cela vient appuyer l'argument présenté par Collins et al. (2005), qui montrent que les modèles basés sur les segments de phrases sont inefficaces pour traduire entre l'anglais et l'allemand. Ils expliquent cette inefficacité en partie par les réordonnancements résultants des différentes syntaxes de ces deux langues. En effet, PHARAOH et RAMSES ne font que pénaliser le réordonnancement, ils ne le modélisent pas. De plus l'allemand à une morphologie entraînant la création de plusieurs mots, ce qui signifie que beaucoup de mots inconnus sont rencontrés pendant le décodage. Une façon de contourner le premier problème est d'utiliser un modèle pour réordonner les mots des phrases sources avant de les donner en entrée au décodeur (Collins et al., 2005; Langlais et al., 2005a). Si la traduction se fait de l'allemand vers l'anglais, un analyseur morphologique peut être utilisé pour décomposer les mots allemands, diminuant ainsi le nombre de mots inconnus.

5.2.2 Évaluation manuelle

L'évaluation automatique mesure la ressemblance entre les traductions candidates et la traduction de référence. Cependant, une traduction candidate peut être à la fois valide et très différente de la traduction de référence. Nous avons donc mené une évaluation manuelle qui vérifie si les deux décodeurs produisent des traductions de qualités équivalentes pour deux sujets humains.

Pour chaque paire de langues sur lesquelles les décodeurs ont été évalués, nous avons demandé à deux sujets de comparer la sortie des deux décodeurs sur 100 phrases choisies aléatoirement dans le corpus FINAL. Les 100 phrases choisies étaient composées de moins de 26 mots et étaient traduites différemment par les deux décodeurs. Les résultats de cette comparaison sont présentés dans le Tableau 5.4.

Sujet	Préférence			Amélioration	
	PHARAOH	RAMSES	Aucune	PHARAOH	RAMSES
			espagnol → anglais		
A	13	16	71	6	1
B	23	31	46	3	8
			français → anglais		
A	18	19	63	5	3
B	20	21	59	8	8
			allemand → anglais		
A	24	18	58	5	9
B	30	31	39	3	3
Total	128	136	336	30	32

Tableau 5.4 – Comparaison manuelle entre les traductions produites par RAMSES et celles produites par PHARAOH.

La comparaison de la sortie des deux décodeurs s'est faite une phrase à la fois en demandant aux sujets de répondre à deux questions. Chaque sujets devait d'abord spécifier s'il préférait la sortie d'un décodeur par rapport à la sortie de l'autre (*préférence*). Si c'était le cas, il devait indiquer si sa traduction préférée l'aidait à mieux comprendre le sens de la phrase source que l'autre traduction (*amélioration*). Pour répondre à ces questions, le sujet avait accès à la phrase source, à la phrase de référence et aux traductions produites par les deux décodeurs. Le décodeur ayant produit chacune des traductions

candidates était cependant inconnu des sujets. Les deux sujets comprenaient le français et l'anglais. Cependant, le sujet A ne comprenait pas l'espagnol et aucun des sujets ne comprenaient l'allemand.

Toutes les paires de langues évaluées confondues, les deux sujets n'ont pas préféré plus souvent la sortie d'un décodeur ou de l'autre. Seulement le quart des traductions que les sujets ont préférées améliorent leur compréhension de la phrase source.

Les réponses des sujets sur la traduction du français vers l'anglais sont plus similaires que les réponses sur les autres paires de langues. Cela s'explique peut-être par le fait que c'est la seule tâche où la langue source et la langue cible étaient maîtrisées par les deux sujets.

Les sujets A et B préfèrent RAMSES à PHARAOH respectivement dans 64% et 48% des cas. Avec cette variance et seulement deux sujets, il faut prendre ces résultats avec un grain de sel. En les ajoutant à ceux obtenus lors de l'évaluation automatique, nous pensons cependant pouvoir affirmer que les deux décodeurs produisent des traductions de qualité similaire.

5.2.3 Quelques exemples

Afin de faire parler les chiffres, des traductions de différentes longueurs pour chacune des paires de langues évaluées dans cette section sont présentées dans l'annexe II.

J'ai aussi utilisé RAMSES avec le modèle permettant de traduire de l'anglais vers le français pour traduire le résumé de ce mémoire (page iv). La traduction produite est présentée à l'annexe I. En lisant cette traduction il faut cependant garder en tête que le modèle a été entraîné sur des textes de débats parlementaires et non sur des textes semblables à ce mémoire. La qualité de la traduction en est donc affectée et son score BLEU est seulement de 16.92.

5.3 Rapidité des décodeurs

La rapidité des deux décodeurs a été évaluée sur deux points. Le premier est la vitesse absolue ou le nombre de phrases traduites par minute. Le deuxième est la vitesse relative ou le nombre d'états parcourus par seconde. Les résultats obtenus pour les traductions du corpus TEST sur un ordinateur ayant un processeur AMD Opteron 246 cadencé à deux

Décodeur	minutes	traductions/min	milliards d'états	états/sec
		allemand → anglais		
PHARAOH	81	24.6	30.4	624 944
RAMSES	224	8.9	43.9	327 926
		espagnol → anglais		
PHARAOH	126	15.7	45.6	600 847
RAMSES	332	6.0	60.6	304 881
		français → anglais		
PHARAOH	166	12.0	57.0	580 547
RAMSES	317	6.3	68.4	359 441
		anglais → allemand		
PHARAOH	81	24.6	30.4	624 944
RAMSES	223	8.9	43.1	327 926
		anglais → espagnol		
PHARAOH	125	15.9	43.0	573 385
RAMSES	261	7.6	53.0	342 837
		anglais → français		
PHARAOH	119	16.7	41.9	583 877
RAMSES	241	8.2	50.2	346 867

Tableau 5.5 – Évaluation de la vitesse de PHARAOH et de RAMSES.

gigahertz et huit gigaoctets de mémoire vive sont présentés dans le Tableau 5.5.

RAMSES parcourt environ deux fois moins d'états par seconde que PHARAOH. Cela signifie que si les deux implémentations utilisaient des algorithmes identiques, PHARAOH serait environ deux fois plus rapide que RAMSES. Ce n'est cependant pas le cas, RAMSES produit environ trois fois moins de phrases par minutes que PHARAOH. Cette différence est due au plus grand nombre d'états que RAMSES explore.

En analysant la trace de PHARAOH, nous nous sommes aperçus qu'il ne semble pas considérer la valeur de l'attribut *next* lorsqu'il compare l'état de deux traductions. Cela veut donc dire que PHARAOH recombine plus d'états, donc en explore moins. Nous ne savons cependant pas si cela seul explique la différence entre le nombre d'états explorés par les deux décodeurs.

En n'ayant pas accès au code source de PHARAOH, il est difficile de savoir si RAMSES

parcourt moins d'états par seconde à cause de la structure imposée par MOOD ou à cause de l'utilisation de structures de données et d'algorithmes différents. Je crois cependant que RAMSES pourrait être plus rapide si les piles étaient élaguées chaque fois que de nouvelles hypothèses y sont ajoutées plutôt que seulement avant d'être parcourues. Ainsi, le programme nécessiterait moins d'espace mémoire et son temps d'exécution en serait sûrement amélioré. De plus, calculer un code de hachage sur l'état de chaque hypothèse et utiliser une table de hachage accélérerait sûrement la détection des hypothèses ayant le même état.

Pour traduire les 2000 phrases du corpus TEST, selon la paire de langue, RAMSES a exploré entre 40 et 70 milliards d'états, et ce même si une politique d'élagage a été appliquée. Il est donc important d'avoir un décodeur rapide pour que le temps de traduction reste raisonnable.

5.4 Retour sur la tâche partagée de l'atelier WMT

Les résultats préliminaires de la tâche partagée de l'atelier WMT ont été disponibles tout juste avant le dépôt de ce mémoire. Notre participation est décrite par Patry et al. (2006) et une comparaison de RAMSES avec le système ayant eu le meilleur score BLEU pour chacune des tâches est présentée dans le Tableau 5.6. Ces résultats ont été calculés sur le corpus FINAL et ont été rapportés par Koehn et Monz (2006).

À l'exception des traductions de l'allemand vers l'anglais et de l'anglais vers l'espagnol, les différences entre les scores BLEU du meilleur système et de RAMSES sont inférieures à 1%. Cela confirme encore une fois que RAMSES est un décodeur suivant les règles de l'art quant à la qualité des traductions qu'il produit.

Les organisateurs de la tâche partagée ont aussi demandé à chaque participant d'offrir du temps pour mener une évaluation manuelle des systèmes. Cette évaluation portait sur la fluidité et la suffisance des phrases. Une phrase cible est *fluide* si elle est bien formée et elle est *suffisante* si elle traduit fidèlement le contenu de la phrase source. Les scores rapportés dans le Tableau 5.6 sont normalisés de telle sorte qu'un nombre positif indique que le système est mieux que la moyenne et un score négatif qu'il est pire que la moyenne. RAMSES semble donc avoir des résultats suivant la moyenne en ce qui concerne ces évaluations manuelles.

Décodeur	BLEU	Fluidité	Suffisance
	allemand → anglais		
RAMSES	24.6	+0.00	-0.1
uedin-phi	27.3	+0.33	+0.30
	espagnol → anglais		
RAMSES	30.80	-0.02	+0.00
lcc	31.46	+0.04	+0.08
	français → anglais		
RAMSES	30.39	+0.00	-0.02
lcc	30.81	+0.13	+0.14
	anglais → allemand		
RAMSES	17.93	+0.03	+0.08
ntt	18.15	+0.09	+0.19
	anglais → espagnol		
RAMSES	29.38	-0.03	-0.05
upc-mr	31.06	+0.17	+0.20
	anglais → français		
RAMSES	31.79	-0.09	-0.08
ntt	31.92	-0.06	-0.09

Tableau 5.6 – Comparaison de RAMSES avec le système ayant obtenu le meilleur score BLEU pour chacune des tâches de traductions de l'atelier WMT.

Au moment d'écrire ces lignes, il est difficile de dire en quoi RAMSES est différent des autres décodeurs, les publications les décrivant n'étant pas encore parues. Les organisateurs de la tâche partagée ont cependant mentionné que presque tous les systèmes ont utilisé un modèle basé sur les segments de phrases.

5.5 Conclusion

Les évaluations automatiques et manuelles que nous avons faites de RAMSES et de PHARAOH nous ont menés à la conclusion que les deux décodeurs produisent des traductions de qualités similaires. Même si les différences entre certains de leurs scores BLEU sont statistiquement significatives, l'évaluation manuelle nous porte à conclure que ces différences sont trop petites pour réellement avoir un impact sur la qualité des traductions produites.

La qualité des traductions produites par RAMSES est aussi similaire à la qualité des traductions produites par les systèmes des autres participants de la tâche partagée mise en place par les organisateur de l'atelier WMT. Je crois donc que nous sommes en mesure d'affirmer que RAMSES produit des traductions dont la qualité suit les règles de l'art.

D'un autre côté, RAMSES est de deux à trois fois plus lent que PHARAOH. Cela pourrait poser problème pour son utilisation à grande échelle. Je crois cependant que l'architecture modulaire de RAMSES et la disponibilité de son code source compensent largement cette différence de vitesse pour son utilisation en recherche.

CHAPITRE 6

CONCLUSION

Dans le paradigme de la traduction statistique, une phrase source est traduite par toutes les phrases de la langue cible, mais avec des probabilités différentes. Le rôle du modèle consiste à calculer ces probabilités et le rôle du décodeur à utiliser le modèle pour traduire une phrase source.

Pour accomplir sa tâche, le décodeur transforme progressivement une traduction initiale en traductions complètes. Donc, plutôt que de chercher la phrase cible ayant la plus grande probabilité de traduire une phrase source, le décodeur cherche la séquence de transformations la plus probable à appliquer à une traduction initiale pour générer une traduction de la phrase source.

L'ensemble des séquences de transformations pouvant être appliquées à une traduction initiale est cependant très grand. Pour opérer dans un temps raisonnable, le décodeur doit en évaluer seulement un sous-ensemble. Si ce sous-ensemble est trop petit, le décodeur sera rapide, mais il risquera aussi de ne pas trouver les meilleures traductions. Si ce sous-ensemble est trop grand, le décodeur trouvera sûrement des meilleures traductions, mais il sera plus lent. Le défi que doit relever un décodeur est de trouver un compromis entre la qualité des traductions qu'il produit et la vitesse à laquelle il les produit.

Il est déjà connu qu'un décodeur est composé de deux parties indépendantes, une représentation du modèle et une stratégie de recherche. Dans ce mémoire, je propose de séparer à son tour la représentation du modèle en quatre parties : les traductions partielles, les transformations, le coût et le générateur de transformations. Utiliser cette structure m'a permis de décrire systématiquement les différents décodeurs que j'ai présentés au chapitre 3. Cette structure m'a aussi aidé à comparer les différents décodeurs que j'ai rencontrés dans la littérature.

La contribution principale de ce mémoire est MOOD, un cadre d'applications permettant d'implémenter des décodeurs. Il est disponible à l'adresse `http://smtmood.sourceforge.net`. La structure d'un décodeur mis en œuvre avec MOOD est la même que celle que j'ai proposée pour en faire la description. Le passage entre la description d'un décodeur et son implémentation devient donc direct.

Lors de la conception de MOOD, j'ai réduit au minimum les dépendances entre ses différents modules. Cela permet d'utiliser le même module avec plusieurs décodeurs et facilite ainsi la création de nouveaux décodeurs.

Afin de mettre MOOD à l'épreuve, je l'ai utilisé pour mettre en œuvre RAMSES, que j'ai ensuite comparé à PHARAOH (Koehn, 2004), le décodeur de référence dans la littérature. Cette comparaison m'a mené à la conclusion que les deux décodeurs produisent des traductions de qualité similaire, mais que PHARAOH est de deux à trois fois plus rapide que RAMSES. En participant à une tâche partagée, nous avons aussi constaté que la qualité des traductions produites par RAMSES est très similaire à celle des traductions produites par des systèmes suivant les règles de l'art. Ces résultats sont encourageants, car ils nous indiquent que RAMSES est une bonne base pour créer de nouveaux décodeurs.

De plus, RAMSES offre plusieurs des fonctionnalités de PHARAOH, comme la génération du graphe de recherche et la traduction en plusieurs passages (*rescoring*). Comme RAMSES utilise la même ligne de commande que PHARAOH, il peut facilement être utilisé avec des outils conçus pour interagir avec PHARAOH.

Un objectif que je me suis fixé en créant MOOD est de mettre en place un lieu commun où les artisans de la traduction statistique peuvent partager leur expertise sur les décodeurs. C'est pourquoi son code source est distribué librement sous la licence GPL. Ainsi tout le monde peut utiliser MOOD et y contribuer.

Au moment de débuter l'écriture de ce mémoire aucun décodeur suivant les règles de l'art n'était distribué avec son code source. Quelques uns l'étaient gratuitement, comme REWRITE (Germann, 2003) ou PHARAOH (Koehn, 2004), mais seulement sous leur forme binaire. Les choses ont cependant changé depuis. Deux autres décodeurs sont maintenant distribués sous une licence similaire à celle de MOOD. Le premier est GENPAR (Burbank et al., 2005), qui offre des outils pour entraîner des modèles hiérarchiques et le décodeur permettant de les utiliser. Le deuxième est PHRAMER[1], qui tout comme RAMSES, est très similaire à PHARAOH. Comme je n'ai pas encore eu le temps de les examiner en détails, je ne peux rien ajouter d'autre sur ces deux décodeurs, sinon que leur existence confirme le besoin qu'a la communauté de la traduction statistique d'un décodeur distribué librement et facile à modifier.

Beaucoup de travail reste encore à faire. Pour nous en convaincre, nous n'avons qu'à

[1] `http://www.utdallas.edu/~mgo031000/phramer/index.html`

jeter un oeil à l'annexe I, qui présente la traduction française que RAMSES a produite à partir du résumé anglais de ce mémoire. Avant tout, je compte créer une documentation suffisamment riche pour que MOOD puisse être utilisé par un nouveau venu. Je compte aussi ajouter à MOOD du support pour les modèles basés sur les segments de phrases trouées (Simard et al., 2005; Langlais et al., 2005a) et pour les modèles hiérarchiques (Galley et al., 2004; Chiang, 2005). De plus, j'ai l'intention de vérifier comment améliorer la vitesse de RAMSES. Finalement, le plus important est que MOOD évoluera selon les idées que ses utilisateurs voudront évaluer et qu'il sera au coeur des développements en traduction automatique menés au RALI.

BIBLIOGRAPHIE

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra et Robert L. Mercer. The mathematics of statistical machine translation : parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993. ISSN 0891-2017.

A. Burbank, M. Carpuat, S. Clark, M. Dreyer, P. Fox, D. Groves, K. Hall, M. Hearne, D. Melamed, Y. Shen, A. Way, B. Wellington et D. Wu. Final report of the 2005 language engineering workshop on statistical machine translation by parsing. Rapport technique, John Hopkins University, 2005.

John Chandioux. Meteo : An operational translation system. Dans *Proceedings of the 2nd Conference on RIAO*, pages 829–839, Cambridge, MA, USA, 1988.

David Chiang. A hierarchical phrase-based model for statistical machine translation. Dans *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 263–270, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

Philip Clarkson et Ronald Rosenfeld. Statistical language modeling using the CMU-cambridge toolkit. Dans *Proc. Eurospeech '97*, pages 2707–2710, Rhodes, Greece, 1997.

Michael Collins, Philipp Koehn et Ivona Kucerova. Clause restructuring for statistical machine translation. Dans *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 531–540, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

Thomas Cormen, Charles Leiserson et Ronald Rivest. *Introduction à l'algorithmique*. The MIT Press, 1994.

Marcello Federico et Nicola Bertoldi. A word-to-phrase statistical translation model. *ACM Trans. Speech Lang. Process.*, 2(2):1–24, 2005. ISSN 1550-4875.

Michel Galley, Mark Hopkins, Kevin Knight et Daniel Marcu. What's in a translation rule ? Dans *HLT-NAACL*, pages 273–280, 2004.

Ismael Garcia-Varea et Francisco Casacuberta. Search algorithms for statistical machine translation based on dynamic programming and pruning techniques. Dans *MT Summit VIII*, Santiago de Compostela, Galicia, Spain, September 2001.

Ulrich Germann. Greedy decoding for statistical machine translation in almost linear time. Dans *NAACL '03 : Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 1–8, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

Ulrich Germann, Michael Jahr, Kevin Knight, Daniel Marcu et Kenji Yamada. Fast decoding and optimal decoding for machine translation. Dans *ACL '01 : Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 228–235, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

John Hutchins. Machine translation over fifty years. *Histoire Epistémologie Langage*, 23 (1):7–31, 2001.

K. Knight et Y. Al-Onaizan. *A Primer on Finite-State Software for Natural Language Processing*, August 1999. http ://www.isi.edu/licensed-sw/carmel/carmel-tutorial2.pdf.

Kevin Knight. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4):607–615, 1999. ISSN 0891-2017.

Donald E. Knuth. *The art of computer programming, volume 3 : (2nd ed.) sorting and searching*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1998. ISBN 0-201-89685-0.

Philipp Koehn. Pharaoh : A beam search decoder for phrase-based statistical machine translation models. Dans *AMTA*, pages 115–124, 2004.

Philipp Koehn. Europarl : A parallel corpus for statistical machine translation. Dans *MT Summit X*, Phuket, Thailand, September 2005.

Philipp Koehn et Christof Monz. Manual and automatic evaluation of machine translation between european languages. http ://www.statmt.org/wmt06/shared-task/, 2006.

Philipp Koehn, Franz Josef Och et Daniel Marcu. Statistical phrase-based translation. Dans *NAACL '03 : Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 48–54, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

Wessel Kraaij, Jian-Yun Nie et Michel Simard. Embedding web-based statistical translation models in cross-language information retrieval. *Computational Linguistics*, 29(3): 381–419, 2003. ISSN 0891-2017.

Philippe Langlais, Fabrizio Gotti, Didier Bourigault et Claude Coulombe. EBMT by tree-phrasing : a pilot study. Dans *2nd Workshop on Example-Based Machine Translation*, Phuket, Thailand, September 2005a.

Philippe Langlais, Simona Grandrabur, Thomas Leplus et Guy Lapalme. The long-term forecast for weather bulletin translation. *Machine Translation*, pages 83–112, 2005b.

Daniel Marcu et Daniel Wong. A phrase-based, joint probability model for statistical machine translation. Dans *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 133–139, Philadelphia, July 2002. Association for Computational Linguistics.

S. Nießen, S. Vogel, H. Ney et C. Tillmann. A DP based search algorithm for statistical machine translation. Dans *Proceedings of the 17th international conference on Computational linguistics*, pages 960–967, Morristown, NJ, USA, 1998. Association for Computational Linguistics.

Nils J. Nilsson. *Problem-Solving Methods in Artificial Intelligence*. McGraw-Hill Pub. Co., 1971. ISBN 0070465738.

Franz Josef Och. Minimum error rate training in statistical machine translation. Dans *ACL '03 : Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 160–167, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

Franz Josef Och et Hermann Ney. Discriminative training and maximum entropy models for statistical machine translation. Dans *ACL*, pages 295–302, 2002.

Franz Josef Och et Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.

Franz Josef Och et Hermann Ney. The alignment template approach to statistical machine translation. *Comput. Linguist.*, 30(4):417–449, 2004. ISSN 0891-2017.

Franz Josef Och, Nicola Ueffing et Hermann Ney. An efficient A* search algorithm for statistical machine translation. Dans *Data-Driven Machine Translation Workshop*, pages 55–62, Toulouse, France, July 2001.

Daniel Ortiz, Ismael García-Varea et Francisco Casacuberta. An empirical comparison of stack-based decoding algorithms for statistical machine translation. Dans *IbPRIA*, pages 654–663, 2003.

D. Ortiz-Martínez, I. García-Varea et F. Casacuberta. THOT : a toolkit to train phrase-based statistical translation models. Dans *Tenth Machine Translation Summit*. AAMT, Phuket, Thailand, September 2005.

Kishore Papineni, Salim Roukos, Todd Ward et Wei-Jing Zhu. Bleu : a method for automatic evaluation of machine translation. Dans *Proceedings of the Association of Computational Linguistics*, pages 311–318, 2002.

Alexandre Patry, Fabrizio Gotti et Philippe Langlais. MOOD at work : Ramses versus pharaoh. Dans *Proceedings on the Workshop on Statistical Machine Translation*, pages 126–129, New York City, June 2006. Association for Computational Linguistics.

Alexandre Patry et Philippe Langlais. Paradocs : un système d'identification automatique de documents parallèles. Dans *12e Conference sur le Traitement Automatique des Langues Naturelles (TALN)*, pages 223–232, Dourdan, France, June 6-10 2005.

Michel Simard, Nicola Cancedda, Bruno Cavestro, Marc Dymetman, Eric Gaussier, Cyril Goutte, Kenji Yamada, Philippe Langlais et Arne Mauser. Translating with non-contiguous phrases. Dans *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 755–762, Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics.

A. Stolcke. SRILM - an extensible language modeling toolkit. Dans *Proceedings of ICSLP*, Denver, Colorado, Sept 2002.

C. Tillman, S. Vogel, H. Ney, H. Sawaf et A. Zubiaga. Accelerated DP based search for statistical translation. Dans *Proceedings of the 5th European Conference on Speech Communication and Technology*, pages 2667–2670, Rhodes, Greece, September 1997a.

C. Tillman, S. Vogel, H. Ney et A. Zubiaga. A DP-based search using monotone alignments in statistical translation. Dans *Proceedings of the ACL/EACL '97*, pages 289–296, Madrid, Spain, July 1997b.

Christoph Tillmann et Hermann Ney. Word reordering and a dynamic programming beam search algorithm for statistical machine translation. *Comput. Linguist.*, 29(1): 97–133, 2003. ISSN 0891-2017.

Nicola Ueffing, Franz Josef Och et Hermann Ney. Generation of word graphs in statistical machine translation. Dans *Proc. Conference on Empirical Methods for Natural Language Processing*, pages 156–163, Philadelphia, July 2002.

Ashish Venugopal, Stephan Vogel et Alex Waibel. Effective phrase translation extraction from alignment models. Dans *ACL*, pages 319–326, 2003.

Stephan Vogel, Hermann Ney et Christoph Tillmann. HMM-based word alignment in statistical translation. Dans *Proceedings of the 16th conference on Computational linguistics*, pages 836–841, Morristown, NJ, USA, 1996. Association for Computational Linguistics.

Stephan Vogel, Ying Zhang, Fei Huang, Alicia Tribble, Ashish Venugopal, Bing Zhao et Alex Waibel. The CMU statistical machine translation system. Dans *Proceedings of the Machine Translation Summit IX*, New Orleans, LA, 2003.

Daniel J. Walker. The open A.I. kit™ : General machine learning modules from statistical machine translation. Dans *Workshop on MT Summit X, "Open-Source Machine Translation"*, Phuket, Thailand, 2005.

Ye-Yi Wang et A. Waibel. Decoding algorithm in statistical translation. Dans *Proceedings of the ACL/EACL '97*, pages 366–372, Madrid, Spain, July 1997.

Andy Way et Nano Gough. Comparing example-based and statistical machine translation. *Natural Language Engineering*, 11(3):295–309, 2005. ISSN 1351-3249.

Ying Zang et Stephan Vogel. Measuring confidence intervals for the machine translation evaluation metrics. Dans *Proceedings of the 10th International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 85–94, Baltimore, Maryland, USA, October 2004.

Annexe I

Traduction automatique du résumé

Ce résumé est une traduction du résumé anglais produite par RAMSES *en utilisant la configuration décrite au chapitre 5.*

ABSTRAIT

Un système statistique machine traduction se traduit par l'objectif d'avoir la plus haute probabilité de traduire une source document document. Un tel système est composé d'un modèle et un décodeur. Le modèle computes la probabilité que l'un document traduit l'autre et le décodeur utilise le modèle de trouver l'objectif document avec la plus haute probabilité de traduction.

Dans ce maître la thèse, je m'intéresse à introduire un cadre d'appliquer telle humeur, et je décodeurs C++ décodeurs. J'ai ensuite montrent combien j'ai utilisé humeur à mettre en oeuvre RAMSES, un décodeur similaire à PHARAOH (koehn 2004), un système largement utilisées dans la communauté baseline, phrase-based. D'encourager la recherche, atmosphère et RAMSES sont publiés dans le domaine public avec leurs code source accompagnée.

La comparaison entre RAMSES et PHARAOH m'amène à conclure que les décodeurs produire traductions de qualité que j'ai effectué similaires. Afin de comparer RAMSES contre d'autres décodeurs, j'ai participé à une tâche commune qui a été organisée dans l'atelier sur machine traduction que s'est tenue à New-York sur 9 juin 2006. Les résultats de cette tâche commune montrent que la qualité des traductions produite par RAMSES similaire à celle produite par des systèmes de pointe.

Les mots clés : d'intelligence artificielle décodeur, machine traduction, phrase-based décodeur, statistique machine de traduction.

Annexe II

Divers exemples de traductions

Cette annexe présente divers exemples de traductions produites par RAMSES et PHARAOH sur des phrases de différentes longueurs tirées aléatoirement du corpus FINAL (section 5.1.1).

Traduction de l'allemand vers l'anglais

Système	Phrase
	10 mots
source	auch zu hause steckt assads regime in einer zwickmühle .
référence	at home , assad ' s regime also finds itself in a bind .
PHARAOH	in the home is assads regime in a paradoxical situation .
RAMSES	at home assads regime is in a paradoxical situation .
	20 mots
source	in den kommenden monaten dürften mehrere pensionierungsrunden es dem präsidenten erlauben , weitere seiner leute in machtpositionen zu bringen .
référence	in the coming month , several rounds of retirements should allow the president to place more of his people in positions of power .
PHARAOH	the president may , in its power to bring in the coming months will pensionierungsrunden several other people .
RAMSES	the president may , in its power to bring it in the coming months will pensionierungsrunden several other people .
	40 mots
source	dies ist der grund , warum syriens leugnen , dass es im irak eine störrolle spiele , selten ernst genommen wird und warum assad keine unterstützung seitens der amerikaner für neuerliche verhandlungen mit israel über die golan-höhen erreicht hat .

référence	that is why syria ' s denials that it is playing a spoiler role in iraq are rarely taken seriously , and why assad has not gained american support for renewed negotiations with israel over the golan heights .
PHARAOH	this is the reason why syria deny that there will be taken seriously and why no support from the americans for further negotiations with israel on the golan heights has achieved a störrolle games in iraq , seldom assad .
RAMSES	this is the reason why it is taken seriously and why no support from the americans for further negotiations with israel on the golan heights has achieved a störrolle games deny that syria in iraq , rarely assad .

Traduction de l'espagnol vers l'anglais

Système	Phrase
	10 mots
source	ambos lados tendrán que trabajar duro si desean ganar .
référence	both sides will have to work hard if they want to win .
PHARAOH	both sides will have to work hard if they want to earn .
RAMSES	both sides will have to work hard if you want to win .
	20 mots
source	hay signos de que los sirios tampoco son capaces de medir las intenciones de estados unidos en otros temas .
référence	there are signs that the syrians are unable to gauge american intentions on other issues as well .
PHARAOH	there are signs that the syrians are not able to measure the intentions of the united states in other issues .
RAMSES	there are signs that the syrian authorities are also able to assess the intentions of the united states on other subjects .

	40 mots
source	tales exigencias crecieron dramáticamente después de la muerte de hariri , a medida que decenas de miles de partidarios del ex primer ministro , que antes se mantenían neutrales en cuanto a siria , gritaban ” fuera siria ” .
référence	such demands escalated dramatically after hariri ’ s death , as tens of thousands of the former prime minister ’ s partisans , who previously sat on the fence when it came to syria , shouted ” syria out . ”
PHARAOH	these requirements experienced growth dramatically after the death of hariri , as in favour of the former prime minister , that before remained neutral with regard to syria , gritaban ’ outside ’ tens of thousands of syria .
RAMSES	after the deaths of tens of thousands of supporters of the former prime minister , which remained neutral with regard to gritaban ’ outside ’ syria , syria , as such demands experienced growth dramatically hariri before .

Traduction du français vers l’anglais

Système	Phrase
	10 mots
source	chaque partie justifie son intransigeance avec des arguments tendancieux .
référence	both sides justify their intransigence with one-sided interpretations .
PHARAOH	each part justifies its intransigence with arguments tendentious .
RAMSES	each side justifies its intransigence with arguments tendentious .
	20 mots
source	m . assad entre maintenant en phase de consolidation , ce qui laisse peu de place pour les réformes .
référence	assad is now in a consolidation phase , leaving little room for reform .

PHARAOH	. i assad between now phase of consolidation , which leaves very little room for reforms .
RAMSES	i now between phase of consolidation , which leaves little room for reforms . assad .
	40 mots
source	le deuxième grand axe d ' une telle action de recréation de valeurs concerne ce que j ' appelle la constitution sociale et fiscale de la société , auxquelles il faudra ajouter les règles en matière d ' environnement .
référence	the second axis of values concerns social justice and the environment .
PHARAOH	the second major axis of such action is what i call on the constitution of society , which we should add the values of recréation of social and fiscal rules on the environment .
RAMSES	the second great axis of such action is what i call on the constitution of society , which we should add the rules on the environment . recréation values of social and fiscal

Traduction de l'anglais vers l'allemand

Système	Phrase
	10 mots
source	syria may pay a heavy price for that ambiguity .
référence	diese zweideutigkeit könnte syrien teuer zu stehen kommen .
PHARAOH	syrien kann teuer für die zweideutigkeit .
RAMSES	syrien vielleicht teuer dafür geben .
	20 mots
source	for example , assad apparently still does not realize how much the bush administration associates his regime with terrorism .
référence	so scheint assad beispielsweise noch immer nicht zu erkennen , wie sehr die bush-administration sein regime mit dem terrorismus in verbindung bringt .

PHARAOH	zum beispiel assad offensichtlich noch nicht wissen , wie sehr die bush-administration mitgliedern seines regimes mit dem terrorismus .
RAMSES	beispielsweise assad offensichtlich noch nicht erkennen , wieviel die bush-administration mitgliedern seines regimes mit dem terrorismus .

40 mots

source	chávez insists on the recall referendum mechanism included in his constitution , which the opposition cannot tolerate : removing the president in this way would require a larger absolute number of votes than chávez garnered in the last election .
référence	chávez besteht auf der in seine verfassung eingebauten möglichkeit , volksentscheide zu widerrufen , was wiederum die opposition nicht tolerieren kann : wollte man den präsidenten auf diese weise absetzen , bräuchte man in absoluten zahlen eine größere anzahl an ja-stimmen , als chávez in den letzten wahlen erhalten hat .
PHARAOH	chávez besteht auf der volksabstimmung in seiner verfassung , die die opposition nicht hinnehmen können : die beseitigung der präsident auf diese weise würde erfordern eine größere zahlenmäßige abstimmungen als chávez erzielt in den letzten wahlen . erinnern mechanismus aufgenommen
RAMSES	er die volksabstimmung in seiner verfassung , die die opposition nicht dulden können : die beseitigung der präsident auf diese weise würde erfordern eine größere anzahl der abstimmenden chávez erzielt als in den letzten wahlen . absolute chávez erinnern mechanismus aufgenommen

Traduction de l'anglais vers l'espagnol

Système	Phrase

10 mots

source	syria may pay a heavy price for that ambiguity .

référence	siria tal vez tendrá que pagar un precio alto por esa ambigüedad .
PHARAOH	siria puede pagar un precio muy elevado para que ambigüedad .
RAMSES	siria puede pagar un alto precio por esta ambigüedad .

20 mots

source	for example , assad apparently still does not realize how much the bush administration associates his regime with terrorism .
référence	por ejemplo , assad aparentemente no se ha dado cuenta del grado al que la administración bush relaciona su régimen con el terrorismo .
PHARAOH	por ejemplo , assad , al parecer , todavía no darse cuenta de lo mucho que la administración bush por su régimen con el terrorismo .
RAMSES	por ejemplo , al parecer , todavía no darse cuenta de lo mucho que la administración bush por su régimen assad con el terrorismo .

40 mots

source	chávez insists on the recall referendum mechanism included in his constitution , which the opposition cannot tolerate : removing the president in this way would require a larger absolute number of votes than chávez garnered in the last election .
référence	chávez insiste en recurrir al mecanismo de referendo incluido en su constitución , el que la oposición no puede tolerar : destituir al presidente de esta manera requeriría una cantidad absoluta de votos superior a la que chávez obtuvo en la última elección .
PHARAOH	chávez recordar incluidos en su constitución , que no podemos tolerar el presidente de este modo una mayor número de votos que chávez granjeado en las últimas elecciones . insiste en la referéndum mecanismo de la oposición : eliminación de exigiría absoluta
RAMSES	insiste en la constitución , no podemos tolerar que la oposición : eliminar el presidente de este modo una mayor número de votos que en las últimas elecciones chávez granjeado absoluta exigiría incluidos en su referéndum chávez recordar mecanismo .

Traduction de l'anglais vers le français

Système	Phrase
	10 mots
source	syria may pay a heavy price for that ambiguity .
référence	la syrie paiera peut-être très cher cette ambiguïté .
PHARAOH	syrie peut payer un lourd tribut pour que l ' ambiguïté .
RAMSES	la syrie peut payer un lourd tribut pour que l ' ambiguïté .
	20 mots
source	for example , assad apparently still does not realize how much the bush administration associates his regime with terrorism .
référence	m. assad , par exemple , ne comprend toujours pas , apparemment , combien le gouvernement bush associe son régime au terrorisme .
PHARAOH	par exemple , assad apparemment n ' a pas encore réaliser combien l ' administration bush acolytes son régime avec le terrorisme .
RAMSES	par exemple , assad apparemment n ' a pas encore réaliser combien l ' administration bush associés son régime avec le terrorisme .
	40 mots
source	chávez insists on the recall referendum mechanism included in his constitution , which the opposition cannot tolerate : removing the president in this way would require a larger absolute number of votes than chávez garnered in the last election .
référence	m . chávez insiste sur le mécanisme de référendum de mise en disponibilité prévu dans sa constitution , ce que l ' opposition ne peut tolérer : se débarrasser ainsi du président nécessiterait un nombre bien plus grand de votes que ceux réunis par m .
PHARAOH	chávez inculque le mécanisme inclus dans sa constitution , qui ne peut tolérer le président de cette manière exigerait un plus grand nombre absolu de suffrages que chávez meilleure dans les dernières élections , l ' opposition : élimination rappeler référendum .

RAMSES	le mécanisme inclus dans sa constitution , qui ne peut tolérer le président de cette manière exigerait un plus grand nombre de voix que chávez meilleure dans les dernières élections absolue : élimination l ' opposition chávez inculque rappeler référendum .