19/07/2018

Synthèse de littérature

Traduction automatique

MTI815 : Système de communication vocale

Stephane Gu
GUXS01079507

Table des matières

Introduction

La traduction de la parole automatisée semble être un domaine avec beaucoup d'application utile. D'un point de vue touristique, cela permettrait aux touristes de communiquer librement dans un pays étranger. De plus, dans un monde de plus en plus mondialisé, la possibilité de communiquer facilement faciliterait les échanges internationaux d'un point de vue économique et diplomatique. Ainsi, dans cet article, nous nous intéressons aux différentes techniques existantes à la traduction automatisée.

Description du sujet

La difficulté de la traduction automatisée est qu'elle combine les problèmes liés à la reconnaissance et à la traduction. Ainsi dans cet article, nous analyserons les différentes techniques utilisées dans la traduction automatisée. Enfin, nous aborderons les différents problèmes spécifiques à la traduction automatisée, et leurs solutions possibles. Pour conclure sur comment optimiser le fonctionnement de la traduction automatisée en agissant sur le système de reconnaissances automatique de la parole.

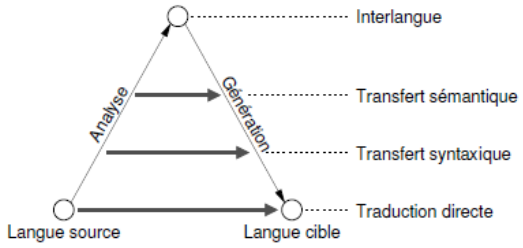

Figure 1: Le triangle de "Vauquois" pour la traduction

Synthèse des techniques

Commençons à décrire les différentes techniques employées pour la traduction automatisée par le principe de base de traduction. Cette pyramide décrit les différents niveaux de traduction, avec la traduction directe un passage direct des mots de la langue source vers la langue cible. Ainsi, si la traduction devient plus simple plus on se rapproche de l'interlangue, la généralisation devient plus complexe. Cette approche fut intéressante pour le principe d'interlangue car elle simplifierait le problème de la traduction par deux problèmes monolingues, un d'analyse et l'autre de synthèse. De plus, les modules monolingues ainsi développer devraient être réutilisables permettant ainsi de limiter le nombre de système de transfert. En effet, avec cette approche pour traduire entre n langues, il faudrait 2n modules d'analyse et synthèse contre n(n-1) systèmes de transfert. Cependant, on découvre rapidement que le principe d'interlangue a un problème de généralisation et avec la puissance limitée des machines de l'époque, l'approche est abandonnée (bien que toujours en recherches). Elle est remplacée maintenant par deux types d'approches, que nous allons décrire maintenant : la traduction automatique à base d'exemples et par méthodes statistiques.

La traduction automatique à base d'exemples

Pour cette approche, le système compare la phrase à traduire avec sa base de donnée d'exemple. Si elle se trouve dans sa base de donnée, la phrase est traduite trivialement. Dans le cas contraire, le système cherche à rassembler des exemples qui contiennent des fragments communs avec la phrase à traduire.
Pour cela, l'algorithme encode la phrase à traduire sous forme de vecteur avec comme données :

- Les mots/phrases fonctionnelles (functional words ou **fws**) : les prépositions, déterminants, pronoms… De manière générale, ce sont les mots de la phrase qui respectent les conditions suivantes :
    - Ils introduisent un comportement syntaxiquement standard
    - La plupart des fws appartienne à des classes fermées
    - Le comportement sémantique des fws est déterminé à travers le contexte
    - La plupart des fws établissent les frontières des phrases
    - Les fws ont une fréquence d'apparition élevée dans le corpus.
- Les lemmes et groupes nominaux entre les fws.

Les composantes principales sont bien sur les fws. Une fois identifiés, on les regroupe en groupe de mots basé sur leur interchangeabilité dans la structure de phrase. Ce regroupement prend en compte la multiplicité d'utilisation de certains mots qui auraient pu être identifiés comme fws.
En ce qui concerne l'extraction des lemmes et groupes nominaux, on introduit le concept de **classe d'ambigüité** (ac) tel que tous les termes non fonctionnels sont représentés par leur classe d'ambiguïté et leur lemme correspondant.

Une fois toutes les données extraites, il faut ensuite calculer leur score de similarité. Pour cela, chaque composante est importante. En effet, pour les fws, on cherche leur équivalent dans la base d'exemple. Il y a 2 cas d'équivalences, le cas où l'exemple et l'entrée sont **identiques** (I) et le cas où les fws ne sont pas identiques mais du **même groupe** (G). Pour les lemmes, il faut que les **sets de lemmes possibles se chevauchent** (L) ou que les **classes d'ambiguïté se chevauchent** (T).
Cependant, ce n'est pas suffisant car nous devons aussi examiner l'ordre des fws, l'absence ou l'ajout de fws entre les deux phrases. On ajoute ainsi un score de **pénalité** (P et PT) négatif qui augmente quand un fw ou un lemme est décalé. Il est donc important de bien définir les valeurs de ces poids pour que le score de similarité soit cohérent. Ainsi, on choisit des valeurs de I, G, L et T qui produisent un score de similarité de 100% en cas de phrases identiques (I et G sont normalisés par la longueur des phrases en fws, T et L sont normalisés par la taille des groupes de mots entre les fws). Tandis que les poids P et Pt reflètent à quelle point l'utilisateur veut pénaliser un ajout ou une suppression.
On constate ainsi que ce score de similarité semble prioriser les contraintes syntaxiques plutôt que les mots en eux même. C'est ce qui différencie cette approche de l'approche par méthode statistiques que nous allons présenter maintenant.

L'approche par méthode statistiques

De la même manière que pour l'approche précédente, cette approche repose sur un corpus parallèle qui est analysé pour estimer les lois de probabilités du modèle statistique. En effet, on suppose que la traduction d'un texte nécessite une prise de décision basé sur des dépendances compliquées à quantifier. C'est ainsi l'approche statistique qui rend compte de ces dépendances. Cela permet ainsi qu'à chaque phrase source il existe une traduction qui transmettra le sens original (erreur de syntaxe possible). La traduction statistique se traduit donc comme une **combinaison** d'un

modèle linguistique et d'une **prise de décision statistique**. Comme il existe plusieurs modèles statistiques de traduction, commençons par le plus simple : le modèle log-linéaire.

Le modèle log-linéaire

Le modèle log-linéaire est fortement inspiré de la reconnaissance automatique de la parole. Pour la prise de décision statistique, on retrouve la notion d'alignement. Pour cette approche, les alignements de groupes de mots à d'autres groupes de mots sont autorisés ainsi que l'alignement au mot NUL lorsque le groupe de mots n'a pas de correspondance dans la phrase cible.

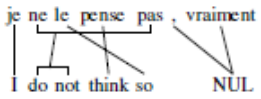

Figure 2: Exemple d'alignement

Ainsi, le modèle statistique cherche à évaluer la probabilité que la phrase **f** = $f_1 \ldots f_J$ soit une traduction de la phrase **e** = $e_1 \ldots e_I$:

$$P(\boldsymbol{f}|\boldsymbol{e}) = \sum_A P(\boldsymbol{f}, A|\boldsymbol{e}) \approx max_A P(\boldsymbol{f}, A|\boldsymbol{e})$$

Avec A l'ensemble des alignements. On considère en pratique que seul l'alignement le plus probable est utilisé.

Comme, on cherche à trouver la phrase cible **e** d'une phrase source **f**, on utilise la relation de Bayes pour faire apparaître les probabilités cherchées :

$argmax_e P(\boldsymbol{e}|\mathbf{f}) = argmax_e P(\boldsymbol{e}) P(\boldsymbol{f}|\boldsymbol{e})$, avec P(e) le modèle de la langue cible.

En pratique, il est bénéfique de pondérer les informations du modèle de langage et traduction tel que l'équation devient :

$argmax_e P(\boldsymbol{e}|\boldsymbol{f}) \approx argmax_e P(\boldsymbol{e})^{\propto} P(\boldsymbol{f}|\boldsymbol{e})^{1-\propto}$, avec α à choisir entre 0 et 1.

On introduit maintenant les **fonctions caractéristiques** $h_i(e,f)$ qui incluent généralement un ou plusieurs modèles de langage h(e,f) = P(e) et tout modèle de traduction h(e,f) = $max_A P(f,A|e)$ ce qui permet d'avoir le modèle log-linéaire :

$$argmax_e P(e|f) \approx argmax_e \prod_i h_i(e,f)^{\lambda i} = argmax_e \exp\left(\sum_i \lambda i \log(h_i(e,f))\right)$$

Les $h_i(e,f)$ sont appris sur un corpus d'entrainement, mais ce sont les λ_i qui paramètrent les modèles. Il est donc important de les estimer sur un ensemble de développement différent de l'ensemble d'entrainement, pour éviter des problèmes de surapprentissage.

Les modèles à base de mots

Dans cette partie, nous abordons différents modèles statistiques de traduction, les modèles à base de mots, c'est-à-dire que l'unité de traduction qui apparait dans les lois de probabilité est le mot. Commençons par les modèles « IBM », il en existe 5 différents qui apportent chacun une expression de P(f|e).

Initialisons ainsi P(f|e) : Soit la phrase cible **e** = $e_1^l \equiv e_1 \ldots e_l$ de l mots et la phrase source **f** = f_1^m de m mots et l'alignement **a** = a_1^m de m valeurs compris entre [0,1] tel que

$$P(\boldsymbol{f}, \boldsymbol{a}|\boldsymbol{e}) = P(m|\boldsymbol{e}) \prod_{j=1}^{m} P\left(a_j|a_1^{j-1}, f_1^{j-1}, m, \boldsymbol{e}\right) P(f_j|a_1^{j-1}, f_1^{j-1}, m, \boldsymbol{e})$$

De plus, nous indiquons les propositions suivantes pour l'alignement :

- Pour tout j de l'intervalle [1,J], a_j appartient à [0,I]

- aj = i > 0 signifie que le mot cible f_j est aligné a e_i.
- aj = 0 signifie que f_j n'est pas aligné ou est aligné avec le mot NUL e_0.
- Pour tout j appartenant à [1,J], $a_j \neq i$ est acceptable, e_i n'a généré aucun f_j

IBM-1

C'est le seul modèle à ne pas aligner les mots sources aux mots cibles, il considère que les alignements entre **tous les mots sources et cibles sont équiprobables**. Ainsi, on suppose donc que

$$P\left(a_j \middle| a_1^{j-1}, f_1^{j-1}, m, e\right) = \frac{1}{l+1}.$$

De plus, on suppose que P(m|e) est indépendant de e et m et que $P\left(f_j \middle| a_1^{j-1}, f_1^{j-1}, m, e\right)$ ne dépend que de f_j et e_{aj}.

Les paramètres du modèle sont donc $\epsilon \equiv P(m|e)$ et $t(f_j|e_{aj}) \equiv P(f_j|a_1^j, f_1^{j-1}, m, e)$ que l'on appelle la probabilité de traduction de f_j sachant e_{aj}. C'est la loi lexicale du modèle, et la seule tel que le modèle est :

$$P(\boldsymbol{f}, \boldsymbol{a}|\boldsymbol{e}) = \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} t(f_j|e_{aj}) \qquad (1)$$

Comme l'alignement est déterminer en spécifiant les valeurs de a_j pour j de 1 a m, on peut réécrire l'équation

$$P(\boldsymbol{f}|\boldsymbol{e}) = \frac{\epsilon}{(l+1)^m} \sum_{a1=0}^{l} \cdots \sum_{am=0}^{l} \prod_{j=1}^{m} t\left(f_j \middle| e_{aj}\right) \qquad (2)$$

Cherchons maintenant à ajuster la loi lexicale dans le **but de maximiser P(f|e)**. Pour cela, on rappelle que e,

$$\sum_f t(f|e) = 1 \qquad (3)$$

Pour cela, nous introduisons les multiplicateurs de Lagrange λ_e et nous cherchons l'*extremum sans contrainte* (unconstrained extremum) de la fonction auxiliaire :

$$h(t, \lambda) \equiv \frac{\epsilon}{(l+1)^m} \sum_{a1=0}^{l} \cdots \sum_{am=0}^{l} \prod_{j=1}^{m} t\left(f_j \middle| e_{aj}\right) - \sum_e \lambda_e \left(\sum_f t(f|e) - 1\right) \qquad (4)$$

On obtient ainsi l'extremum en annulant les dérivées partielles selon les paramètres. Le cas où la dérivée partielle selon λ est nulle est une conséquence des contraintes sur la probabilité de traduction. Pour la dérivée partielle selon h, on a :

$$\frac{\partial h}{\partial t(f|e)} = \frac{\epsilon}{(l+1)^m} \sum_{a1=0}^{l} \cdots \sum_{am=0}^{l} \sum_{j=1}^{m} \frac{\delta(f, f_j)\delta(e, e_{aj})}{t(f|e)} \prod_{k=1}^{m} t(f_k|e_{ak}) - \lambda_e \qquad (5)$$

Avec δ le symbole de Kronecker, qui égal à 1 quand les deux variables sont égales, 0 sinon.
On trouve donc la dérivée nulle pour :

$$t(f|e) = \frac{\epsilon}{\lambda_e (l+1)^m} \sum_{a1=0}^{l} \cdots \sum_{am=0}^{l} \sum_{j=1}^{m} \delta(f, f_j)\delta(e, e_{aj}) \prod_{k=1}^{m} t(f_k|e_{ak}) \qquad (6)$$

Cette équation ne nous permet pas de trouver l'extremum, puisque la loi lexicale est des 2 côtés de l'équation. On obtient cependant une méthode d'estimation de la probabilité de traduction, en ayant une idée initiale de cette probabilité, on peut utiliser l'équation établi pour raffiner l'estimation.
En utilisant l'équation de P(f,a|e), on réécrit l'équation :

$$t(f|e) = \frac{1}{\lambda_e}\sum_a P(\boldsymbol{f},\boldsymbol{a}|\boldsymbol{e})\sum_{j=1}^{m}\delta(f,f_j)\delta(e,e_{aj}) \qquad (7)$$

tel que le nombre attendu de fois que e est aligné à f dans la traduction (f|e) est

$$c(f|e;\boldsymbol{f},\boldsymbol{e}) = \sum_a P(\boldsymbol{a}|\boldsymbol{e},\boldsymbol{f})\sum_{j=1}^{m}\delta\big(f,f_j\big)\delta\big(e,e_{aj}\big) \qquad (8)$$

avec P(**a**|**e**,**f**) = $\frac{P(\boldsymbol{f},\boldsymbol{a}|\boldsymbol{e})}{P(\boldsymbol{f}|\boldsymbol{e})}$ et si on remplace λ_e par λ_eP(**f**|**e**), l'équation devient alors

$$t(f|e) = \frac{c(f|e;\boldsymbol{f},\boldsymbol{e})}{\lambda_e} \qquad (9)$$

Comme les données d'entrainement contiennent des doublons de traduction de la forme ($f^{(1)}|e^{(1)}$),…,($f^{(S)}|e^{(S)}$), alors on a :

$$t(f|e) = \frac{\sum_{s=1}^{S} c(f|e;\boldsymbol{f}^{(s)},\boldsymbol{e}^{(s)}}{\lambda_e} \qquad (10)$$

Avec λ_e un rappel que la probabilité de traduction doit être normalisée.
Comme la loi lexicale semble être une somme de monômes contenant m probabilité de traduction, un pour chaque mot de la phrase f, on en déduit que les différents monômes correspondent à différents alignements de f à e **uniques**. Ainsi, on peut réécrire l'équation sous la forme suivante :

$$\sum_{a1=0}^{l}\cdots\sum_{am=0}^{l}\prod_{j=1}^{m}t\big(f_j\big|e_{aj}\big) = \prod_{j=1}^{m}\sum_{i=0}^{l}t\big(f_j\big|e_i\big) \qquad (11)$$

Ainsi, en réutilisant cette expression dans l'équation de P(f|e), on a alors

$$P(\boldsymbol{f}|\boldsymbol{e}) = \frac{\epsilon}{(l+1)^m}\prod_{j=1}^{m}\sum_{i=0}^{l}t(f_j|e_i) \qquad (12)$$

De plus, cette équation nous permet de simplifier l'équation de la fonction auxiliaire h, tel que :

$$c(f|e;\boldsymbol{f},\boldsymbol{e}) = \frac{t(f|e)}{t(f|e_0)+\cdots+t(f|e_l)}\sum_{j=1}^{m}\delta(f,f_j)\sum_{i=0}^{l}\delta(e,e_i) \qquad (13)$$

tel que $\sum_{j=1}^{m}\delta(f,f_j)$ compte le nombre de mots dans la phrase source et $\sum_{i=0}^{l}\delta(e,e_i)$ compte le nombre de mots dans la phrase cible. Cette nouvelle équation permet ainsi de simplifier le compte puisque le nombre d'opération nécessaire passe de l+m au lieu des $(l+1)^m$ de l'équation initiale.

On en déduit alors une manière d'estimer t(f|e) en utilisant les équations de c(f|e ;f,e) et t(f|e) :

1. Initialiser t(f|e)
2. Pour chaque doublon de phrases ($\boldsymbol{f}^{(s)}$,$\boldsymbol{e}^{(s)}$) s allant de 1 à S, utiliser la nouvelle équation de c(f|e ;**f**,**e**). En sachant que ces comptes seront différents de 0 que quand f est l'un des mots dans $\boldsymbol{f}^{(s)}$ et e l'un des mots dans $\boldsymbol{e}^{(s)}$. De plus, c(f|e ;$\boldsymbol{f}^{(s)}$,$\boldsymbol{e}^{(s)}$) ne dépend pas de l'ordre des mots mais juste de leurs nombres d'apparition dans leurs phrases respectives.
3. Pour chaque e qui apparaît au moins une fois dans $\boldsymbol{e}^{(s)}$:
   a. Calculer $\lambda_e = \sum_f\sum_{s=1}^{S}c\big(f\big|e\,;\boldsymbol{f}^{(s)},\boldsymbol{e}^{(s)}\big)$ (14)
   b. Pour chaque f qui apparaît dans au moins $\boldsymbol{f}^{(s)}$, recalculer t(f|e) avec la dernière équation de t(f|e)
4. Réitérer les étapes 2 et 3 jusqu'à ce que les valeurs de t(f|e) convergent vers le degré voulu.

A noter que ce modèle n'existe que pour entrainer les modèles qui suivent, par conséquent la plupart des équations trouver pour le modèle 1 s'appliquent pour les modèles consécutifs. Nous indiquerons les équations inchangées et les équations a adaptées.

IBM-2

Pour ce modèle, on suppose que $P(a_j|a_1^{j-1},f_1^{j-1},m,\mathbf{e})$ dépend de j, a_j, m et l. Comme on introduit la notion d'alignement dans ce modèle, on a donc une **probabilité d'alignement** de la forme suivante :

$$a(a_j|j,m,l) \equiv P(a_j|a_1^{j-1},f_1^{j-1},m,l) \quad (15)$$

qui vérifie la contrainte suivante pour chaque triplet j,m et l :

$$\sum_{i=0}^{l} a(i|j,m,l) = 1 \quad (16)$$

On retrouve alors l'équation (2) suivante pour le modèle IBM 2 :

$$P(\boldsymbol{f}|\boldsymbol{e}) = \epsilon \sum_{a1=0}^{l} \dots \sum_{am=0}^{l} \prod_{j=1}^{m} t(f_j|e_{aj}) a(a_j|j,m,l) \quad (17)$$

La fonction auxiliaire (4) devient ainsi :

$$h(t,a,\lambda,\mu) \equiv \epsilon \sum_{a1=0}^{l} \dots \sum_{am=0}^{l} \prod_{j=1}^{m} t(f_j|e_{aj}) a(a_j|j,m,l) \sum_e \lambda_e \left(\sum_f t(f|e) - 1 - \sum_j \mu_{jml}\left(\sum_i a(i|j,m,l) - 1\right)\right) \quad (18)$$

A noter que les équations (7),(9) et (10) ne changent pas. Cependant, nous avons besoin d'une nouvelle fonction c(i|j,m,l ;**f,e**) qui compte le nombre de fois qu'un mots est attendu à la position j de **f** qui s'aligne au mot a la position i de **e** :

$$c(i|j,m,l;\boldsymbol{f},\boldsymbol{e}) = \sum_a P(\boldsymbol{a}|\boldsymbol{e},\boldsymbol{f}) \delta(i,a_j) \quad (19)$$

De la même manière que (9) et (10), on a pour une traduction unique :

$$a(i|j,m,l) = \frac{c(i|j,m,l;\boldsymbol{f},\boldsymbol{e})}{\mu_{jml}} \quad (20)$$

et pour un set de traduction :

$$a(i|j,m,l) = \frac{\sum_{s=1}^{S} c(i|j,m,l;\boldsymbol{f}^{(s)},\boldsymbol{e}^{(s)})}{\mu_{jml}} \quad (21)$$

A noter que si $\mathbf{f}^{(s)}$ n'a pas une longueur de m ou que si $\mathbf{e}^{(s)}$ n'as pas une longueur de l, alors le compte correspondant est de 0. De la même manière que pour les λ dans les équations précédentes, μ n'est là que pour nous rappeler de normaliser les probabilités.
On réalise le même raisonnement que sur l'équation (8) sur l'équation (19), nous pouvons réécrire l'équation (17) :

$$P(\boldsymbol{f}|\boldsymbol{e}) = \epsilon \prod_{j=1}^{m} \sum_{i=0}^{l} t(f_j|e_i) a(i|j,m,l) \quad (22)$$

On trouve alors :

$$c(f|e;\boldsymbol{f},\boldsymbol{e}) = \sum_{j=1}^{m} \sum_{i=0}^{l} \frac{t(f|e) a(i|j,m,l) \delta(f,f_j) \delta(e,e_i)}{t(f|e_0) a(0|j,m,l) + \dots + t(f|e_l) a(l|j,m,l)} \quad (23)$$

$$c(i|j,m,l;\boldsymbol{f},\boldsymbol{e}) = \frac{t(f_j|e_i) a(i|j,m,l)}{t(f_j|e_0) a(0|j,m,l) + \dots + t(f_j|e_l) a(l|j,m,l)} \quad (24)$$

On note que la différence des sommes entre l'équation (23) et (13) est dû au fait que i et j sont lié a travers la loi d'alignement.
Ainsi, on se rend compte que le modèle 1 est un cas particulier du modèle 2 où a(i|j,m,l) est fixé a $(l+1)^{-1}$. Cela explique que la plupart des données collectés par le modèle 1 sont utilisables par le modèle 2. Il suffit d'utiliser le modèle de compte du modèle 2 mais en compilant les probabilités d'alignement initié sur le modèle 1.

IBM-3

A partir de ce modèle, la méthode de détermination de P(**f,a|e**) change. En effet, pour les deux premières méthodes nous déterminions les alignements des f_i consécutivement. Pour les méthodes 3,4 et 5 nous introduisons maintenant la notion de **fertilité**. Ainsi, on définit le nombre de mots source aligné a e par ϕ_e la fertilité de e.
Le raisonnement appliqué pour le modèle IMB-3 est le suivant :
Soit une phrase cible **e**, pour chaque mot de cette phrase nous cherchons d'abord la fertilité et sa *tablet* qui est la liste des mots source alignés. On note T la variable aléatoire de l'ensemble des tablets nommé le *tableau* de **e**, avec T_i le tablet du ième mot cible et T_{ik} le kème mot source du ième tablet. Une fois le tableau choisi, on produit **f** en permutant ses mots avec la variable aléatoire Π et Π_{ik} la position dans **f** du kème mot du ième tablet tel que la probabilité conjointe d'un tableau τ et d'une permutation π :

$$P(\tau,\pi|\boldsymbol{e}) = \prod_{i=1}^{l} P\left(\phi_i\middle|\phi_1^{i-1},\boldsymbol{e}\right)P(\phi_i|\phi_1^{l},\boldsymbol{e}) * \prod_{i=0}^{l}\prod_{k=1}^{\phi i} P(\tau_{ik}|\tau_{i1}^{k-1},\tau_0^{i-1},\phi_0^{l},\boldsymbol{e}) * \prod_{i=1}^{l}\prod_{k=1}^{\phi i} P(\pi_{ik}|\pi_{i1}^{k-1},\pi_1^{i-1},\tau_0^{l},\phi_0^{l},\boldsymbol{e}) * \prod_{k=1}^{\phi 0} P(\pi_{0k}|\pi_{01}^{k-1},\pi_1^{l},\tau_0^{l},\phi_0^{l},\boldsymbol{e}) \quad (25)$$

Ici, τ_{i1}^{k-1} représente la serie $\tau_{i1},\ldots,\tau_{ik-1}$ et π_{i1}^{k-1} représente la série $\pi_{i1},\ldots,\pi_{ik-1}$ et ϕ_i est l'abréviation de ϕ_{ei}.
En général, différents couples τ, π permettent d'avoir le même couple **f, a** tel que :

$$P(\boldsymbol{f},\boldsymbol{a}|\boldsymbol{e}) = \sum_{(\tau,\pi)\in(\boldsymbol{f},\boldsymbol{a})} P(\tau,\pi|\boldsymbol{e}) \qquad (26)$$

On sait qu'il y a $\prod_{i=0}^{l}\phi_i!$ éléments dans (**f,a**) car pour chaque τ_i il y a ϕ_i ! arrangement possible vers le couple **f, a**.
De plus, en général, il n'y a qu'un seul alignement qui maximise P(**a|e, f)** que l'on nomme l'alignement de *Viterbi* noté V(**f|e**). Il n'existe cependant pas d'algorithme pour trouver cet alignement. Ainsi, l'alignement de Viterbi dépend du modèle choisi et nous notons l'alignement de Viterbi du modèle i V(**f|e** ;i). Ainsi, si on note $A_{i\leftarrow j}$(**e,f**) le set d'alignement pour lequel a_j = i, alors on note $V_{i\leftarrow j}$(**f|e**) l'élément de $A_{i\leftarrow j}$(**e,f**) pour lequel P(**a|e,f**) est maximale.

Maintenant que nous avons défini l'équation centrale (25) du modèle, nous devons choisir les paramètres indépendants du système. En effet, le modèle dépend du choix que l'on fait maintenant, ainsi on suppose que soit i allant de 1 à l :

- $P(\phi_i|\phi_1^{i-1}, \boldsymbol{e})$ dépend que de ϕ_i et e_i
- $P(\tau_{ik}|\tau_{i1}^{k-1}, \tau_0^{i-1},\phi_0^{l},\boldsymbol{e})$ dépend que de τ_{ik} et e_i
- $P(\pi_{ik}|\pi_{i1}^{k-1}, \pi_1^{i-1}, \tau_0^{l}, \phi_0^{l}, \boldsymbol{e})$ dépend que de π_{ik}, i, m et l.

Ainsi, on a donc la liste de paramètres suivant pour le modèle IMB-3 :

- Un set de probabilité de fertilité $n(\theta|e_i) \equiv P(\phi|\phi_1^{i-1},\boldsymbol{e})$
- Un set de probabilité de traduction $t(f|e_i) \equiv P\left(T_{ik}=f\middle|\tau_{i1}^{k-1},\tau_0^{i-1},\phi_0^{l},\boldsymbol{e}\right)$
- Un set de probabilité de distorsion $d(j|i,m,l) \equiv P\left(\Pi_{ik}=j\middle|\tau_{i1}^{k-1},\pi_1^{i-1},\tau_0^{l},\phi_0^{l},\boldsymbol{e}\right)$

Enfin, ce modèle traite la fertilité et l'alignement du mot e_0 différemment. En effet, on suppose que ce lemme occupe la position 0 alors qu'en réalité il n'en occupe aucune. Ainsi le terme e_0 a pour but de comptabiliser les mots de la phrase source qui ne peuvent pas être pris en compte par les mots de

la phrase cible. Cela permet ainsi à la phrase d'être grammaticalement correcte par exemple. On ne place ces termes qu'une fois tous les autres l'ont été, on suppose donc que

$$P(\Pi_{0k+1} = j | \pi_{01}^{k}, \pi_{1}^{l}, \tau_{0}^{l}, \phi_{0}^{l}, \boldsymbol{e}) = \begin{cases} (\phi_0 - k)^{-1} & si\ la\ position\ j\ est\ disponible \\ 0 & sinon \end{cases}$$

De plus, on suppose que plus la phrase source est longue, plus elle contiendra des termes superflus donc on peut supposer que :

$$P(\phi_0 | \phi_1^l, \boldsymbol{e}) = \binom{\phi_1 + \cdots + \phi_l}{\phi_0} p_0^{\phi 1 + \cdots \phi l - \phi 0} p_1^{\phi 0} \qquad (27)$$

avec p_0 et p_1 des paramètres auxiliaires. L'équation s'interprète de la manière suivante, on imagine que chaque mot de τ_1^l a besoin d'un mot superflu avec une probabilité de p_1 et que ce terme superflu doit être aligné a e_0. L'équation nous donne ainsi la probabilité qu'exactement ϕ_0 des mots de τ_1^l vont avoir besoin d'un mot superflu. Ainsi, si nous appliquions ce modèle adapté au cadre des modèle 1 et 2, nous avons l'équation suivante :

$$P(\boldsymbol{f}|\boldsymbol{e}) = \sum_{a1=0}^{l} \cdots \sum_{am=0}^{l} P(\boldsymbol{f}, \boldsymbol{a}|\boldsymbol{e}) =$$
$$\sum_{a1=0}^{l} \cdots \sum_{am=0}^{l} \binom{m-\phi_0}{\phi_0} p_0^{m-2\phi 0} p_1^{\phi 0} \prod_{i=1}^{l} \phi_i!\, n(\phi_i|e_i) \prod_{j=1}^{m} t(f_j | e_{aj}) d(j|a_j, m, l) \qquad (28)$$

avec $\sum_f t(f|e) = 1$, $\sum_j d(j|i,m,l) = 1$, $\sum_\phi n(\phi|e) = 1\ et\ p_0 + p_1 = 1$.

De la même manière que les modèles précédents, nous établissons les équations suivantes :

$$h(t,d,n,p,\lambda,\mu,\nu,\xi) = P(\boldsymbol{f}|\boldsymbol{e}) - \sum_e \lambda_e \left(\sum_f t(f|e) - 1 - \sum_i \mu_{iml} \left(\sum_j d(j|i,m,l) - 1\right) - \sum_e \nu_e \left(\sum_\phi n(\phi|e) - 1\right) - \xi(p_0 + p_1 - 1)\right) \qquad (29)$$

$$c(f|e;\boldsymbol{f},\boldsymbol{e}) = \sum_a P(\boldsymbol{a}|\boldsymbol{e},\boldsymbol{f}) \sum_{j=1}^{m} \delta(f, f_j)\delta(e, e_{aj}) \qquad (30)$$
$$c(j|i,m,l;\boldsymbol{f},\boldsymbol{e}) = \sum_a P(\boldsymbol{a}|\boldsymbol{e},\boldsymbol{f})\delta(i, a_j) \qquad (31)$$
$$c(\phi|e;\boldsymbol{f},\boldsymbol{e}) = \sum_a P(\boldsymbol{a}|\boldsymbol{e},\boldsymbol{f}) \sum_{i=1}^{l} \delta(\phi, \phi_i)\delta(e, e_i) \qquad (32)$$
$$c(0;\boldsymbol{f},\boldsymbol{e}) = \sum_a P(\boldsymbol{a}|\boldsymbol{e},\boldsymbol{f})(m - 2\phi_0) \qquad (33)$$
$$c(1;\boldsymbol{f},\boldsymbol{e}) = \sum_a P(\boldsymbol{a}|\boldsymbol{e},\boldsymbol{f})\phi_0 \qquad (34)$$

avec c(0 ;**f, e**) et c(1 ;**f**, **e**) les fonctions de compte pour les paramètres p_0 et p_1 respectivement et détermine la fertilité de e_0 dans la langue cible.

De la même manière, nous avons les fonctions d'estimations suivantes pour le modèle 3 :

$$t(f|e) = \frac{\sum_{s=1}^{S} c(f|e;\boldsymbol{f}^{(s)},\boldsymbol{e}^{(s)}}{\lambda_e} \qquad (35)$$
$$d(j|i,m,l) = \frac{\sum_{s=1}^{S} c(j|i,m,l;\boldsymbol{f}^{(s)},\boldsymbol{e}^{(s)})}{\mu_{iml}} \qquad (36)$$
$$n(\phi|e) = \frac{\sum_{s=1}^{S} c(\phi|e;\boldsymbol{f}^{(s)},\boldsymbol{e}^{(s)}}{\nu_e} \qquad (37)$$
$$p_k = \frac{\sum_{s=1}^{S} c(k;\boldsymbol{f}^{(s)},\boldsymbol{e}^{(s)}}{\xi} \qquad (38)$$

On note que les équations (30) et (35) ne changent pas par rapport aux précédents modèles. Cependant, la méthode utilisée jusqu'à maintenant pour évaluer les différentes fonctions de compte n'est plus applicable ici. En effet, avec la présence des paramètres de fertilité, l'inversion des sommes des alignements et le produit sur j est impossible. C'est pour cela que pour ce modèle nous n'appliquons les équations (28) et (30)-(34) que aux alignements les plus probables. Pour identifier ces alignements, nous commençons par l'alignement le plus probable trouvable, puis nous évaluons

les alignements qui sont obtenus par petits changements. Pour cela nous donnons la définition suivante de changement : Soit **a** et **a'** deux alignements,

- **a** et **a'** diffère d'un déplacement (*move*) s'il existe une valeur de j pour laquelle $a_j \neq a'_j$
- **a** et **a'** diffère d'un échange (*swap*) s'il existe 2 valeurs j_1 et j_2 tel que $a_{j1} = a'_{j2}\ et\ a_{j2} = a'_{j1}$
- **a** et **a'** sont des voisins s'ils sont identiques ou diffère d'un déplacement ou échange tel que l'ensemble de tous les voisins de **a** est N(**a**).

Ainsi, supposons que b(**a**) soit le voisin de **a** qui maximise P(b(**a**)|**f,e**). Supposons que ij index **a** (*ij is pegged for* $\boldsymbol{a}$) et que parmi les voisins de **a** qui sont aussi indexés a ij, $b_{i\leftarrow j}$(**a**) maximise P(b(**a**)|**f,e**). La suite d'alignement **a**, b(**a**),b^2(**a**),…., converge en un nombre fini d'étapes vers un alignement $b^{\infty}(\boldsymbol{a})$. De la même manière, la séquence d'alignement **a**, $b_{i\leftarrow j}$(**a**), $b_{i\leftarrow j}{}^2$(**a**) converge vers ${b_{i\leftarrow j}}^{\infty}(\boldsymbol{a})$.
Alors si **a'** est un voisin d'**a** obtenu en déplacent j de i vers i' et si i et i' sont différents de 0 alors :

$$P(\boldsymbol{a}'|\boldsymbol{e},\boldsymbol{f}) = P(\boldsymbol{a}|\boldsymbol{e},\boldsymbol{f})\frac{\phi_{i'}+1}{\phi_i}\frac{n(\phi_{i'}+1|e_{i'})}{n(\phi_{i'}|e_{i'})}\frac{n(\phi_i-1|e_i)}{n(\phi_i|e_i)}\frac{t(f_j|e_{i'})}{t(f_j|e_i)}\frac{d(j|i',m,l)}{d(j|i,m,l)} \qquad (39)$$

avec $\phi_{i'}$ la fertilité du mot en position i' de l'alignement **a** tel que la fertilité de ce mot pour l'alignement **a'** est $\phi_{i'}$+1. On peut donc en déduire le sous ensemble S de A(**f|e**) dans lequel nous évaluons les équations (28) et (30)-(34) :

$$S = N(b^{\infty}\big(V(\boldsymbol{f}|\boldsymbol{e};2)\big)) \cup \cup_{ij}\, b^{\infty}_{i\leftarrow j}(V_{i\leftarrow j}(\boldsymbol{f}|\boldsymbol{e};2) \qquad (40)$$

avec $b^{\infty}\big(V(\boldsymbol{f}|\boldsymbol{e};2)\big)$ et $b^{\infty}_{i\leftarrow j}(V_{i\leftarrow j}(\boldsymbol{f}|\boldsymbol{e};2)$ comme approximations pour V(**f|e** ;3) et $V_{i\leftarrow j}$(**f|e** ;3) que nous ne savons pas calculer efficacement.
Pour l'initialisation du modèle 3, nous calculons les comptes avec les équations (30)-(34) avec le modèle 2 pour évaluer P(**a|e,f**).
On constate cependant un problème avec ce modèle, les probabilités de distorsions pour positionner les mots suivants ne dépendent pas de la position des mots précédents. Ainsi, on gaspille des ressources sur ce qu'on appelle des *phrases généalisées* (generalized strings), c'est-à-dire des phrases qui ont des alignements avec soit plusieurs mots soit aucun. On constate donc que le modèle est déficient, dans le sens où il ne se concentre pas sur des événements d'intérêts. C'est une conséquence directe de la simplicité du modèle, qui nous permet d'écrire l'équation (39).

IBM-4

Comme énoncé précédemment, la loi d'alignement du modèle 3 n'est pas capable de gérer des alignements entre un mot et un groupe de mots. Ainsi, pour résoudre ce problème, on cherche à modifier $P(\Pi_{ik} = j\big|\pi_{i1}^{k-1}, \pi_1^{i-1}, \tau_0^l, \phi_0^l, \boldsymbol{e})$. Pour cela, nous établissons les nouvelles définitions suivantes : soit i la position du ième cept a un mot de la phrase cible.

- Un cept est la classe attribuée au mot qui sont reliés en partie par un même concept.
- Le **centre** de ce cept $\odot_i$ est le plafond de la valeur moyenne des position dans la langue source des mots de sa tablet.
- La **tête** (head) est le mot de la tablet qui a la position dans la phrase source la plus petite.

Ainsi, dans ce modèle nous remplaçons d(j|i,m,l) par deux sets de paramètres, un pour placer la tête de chaque cept, et un pour les mots restants tel que si on veut pour [i]>0, que la tête pour le cept i soit $\tau_{[i]1}$ on ait :

$$P\big(\Pi_{[i]1} = j\big|{\Pi_1}^{[i]-1}, \tau_0^l, \phi_0^l, \boldsymbol{e}\big) = d_1(j - \odot_{i-1}|A\big(e_{[i-1]}\big), B\big(f_j\big)) \qquad (41)$$

Ici, A et B représentent des fonctions des phrases cibles et sources respectivement. Elles prennent des valeurs dépendamment de leurs arguments en fonction de leur vocabulaire respectif.
$j - \odot_{i-1}$est ce qu'on appelle le décalage de la tête pour le cept i et peut être positif ou négatif.

Si on voulait placer le kème mots du cept i pour [i]>0 et k>1, nous supposons que :

$$P\left(\Pi_{[i]k} = j \middle| \pi_{[i]1}^{k-1}, \pi_1^{[i]-1}, \tau_0^l, \phi_0^l, \boldsymbol{e}\right) = d_{>1}(j - \pi_{[i]k-1} | B(f_j)) \quad (42)$$

En ce qui concerne les équations usuelles, nous réutilisons les équations établies précédemment. De la même manière que pour le modèle 3, il n'existe pas de vrai méthode d'évaluation, d'où la création d'un nouveau sous ensemble S utilisé à cet effet. La méthode utilisée est la suivante, nous commençons par classer par ordre décroissant les voisins de **a** tel que le 1er de la liste est celui qui maximise P(**a**|**e,f** ;3). Soit b(**a**) le voisin le plus haut classé tel que P(b(**a**)|**e,f** ;4) est aussi large que P(**a**|**e,f** ;4). Nous définissons $b_{\widetilde{i \leftarrow j}}(\boldsymbol{a})$de la même manière. Nous avons donc

$$S = N(\tilde{b}^{\infty}(V(\boldsymbol{f}|\boldsymbol{e}; 2))) \cup \cup_{ij} N(\tilde{b}_{i \leftarrow j}^{\infty}\left(V_{i \leftarrow j}(\boldsymbol{f}|\boldsymbol{e}; 2)\right)) \quad (43)$$

Ce qui est au final la même définition que pour le modèle 3 a une définition près.
On arrive alors à l'expression suivante du modèle :

$$P(f, A|e) = \prod_{i=1}^{I} n(\phi_i|e_i) * t\left(f_{\tau i,1}\middle|e_i\right) d_{=1}\left(\tau_{i,1} - \overline{\tau_{\iota-1}}\middle|\varepsilon_{i-1}, F\left(f_{\tau i,1}\right)\right)$$
$$* \prod_{k=2}^{\phi i} t\left(f_{\tau i,k}\middle|e_i\right) d_{>1}(\tau_{i,k} - \tau_{i,k-1}|F\left(f_{\tau i,k}\right)n$$
$$* \binom{J - \phi_0}{\phi_0} p_0^{J-2\phi_0}(1 - p_0)^{\phi_0} \frac{1}{\phi_0!} * \prod_{k=1}^{\phi 0} t(f_{\tau 0,k}|e_0)$$

avec $\tau_{i,k} \in$ [1, J] la position du k^{ème} mot cible généré par e_i
F(f) la classe d'un mot cible f
ε_{i-1} la classe du dernier mot source ferticle précedent e_i
$\overline{\tau_{\iota-1}}$ la moyenne des positions des mots cible générés par ce dernier mot source fertile.

IBM-5

De la même manière que le modèle 3 était déficient, nous savons que le modèle 4 l'est aussi. En effet, non seulement plusieurs mots peuvent être superposer au même position, mais les mots peuvent être placer avant ou après la dernière position dans la langue source. Le modèle 5 a donc pour but d'éliminer ces problèmes.

Pour cela, soit $v\left(j, \tau_1^{[i]-1}, \tau_{[i]1}^{k-1}\right)$ le nombre d'emplacement libre jusqu'à la position j inclus juste avant de positionner $\tau_{[i]k}$ que l'on va noter v_j. Comme au modèle précédent, nous conservons les deux lois de distorsion d_1 et $d_{>1}$. Pour [i]>0, nous supposons que

$$P\left(\Pi_{[i]1} = j \middle| \pi_1^{[i\}-1}, \tau_0^l, \phi_0^l, \boldsymbol{e}\right) = d_1\left(v_j \middle| B(f_j), v_{\odot i-1}, v_m - \phi_{[i]} + 1\right)(1 - \delta(v_j, v_{j-1}) \quad (44)$$

Dans le cas où la position j n'est pas libre, le nombre d'emplacement libre jusqu'à j est le même que le nombre d'emplacement jusqu'à j-1. Ainsi, la sortie du symbole de Kronecker est 1 quand j est libre, 0 sinon. On note v_m le nombre d'emplacement libre dans la phrase source tel que $\tau_{[i]1}$ peut être placé dans n'importe quel emplacement sauf les $\phi_{[i]}$-1 positions les plus à droite.
On a ainsi, pour [i]>0 et k>1 : (45)

$$P\left(\Pi_{[i]k} = j \middle| \pi_{[i]1}^{k-1}, \pi_1^{[i]-1}; \tau_0^l, \phi_0^l, \boldsymbol{e}\right)$$
$$= d_{>1}\left(v_j - v_{\pi_{[i]k-1}} \middle| B(f_j)\right), v_j - v_{\pi_{[i]k-1}} - \phi_{[i]} + k)(1 - \delta(v_j, v_{j-1})$$

Le modèle 5 est puissant mais assez peu utilisable. Les modèles 2 à 4 sont donc là pour permettre la réalisation du modèle 5, en initialisant ses paramètres. En effet, la récupération des différents comptes des modèles 2 à 4 avec le sous-espace S décrit avec l'équation (42) permet de calculer P(**a|e,f**) avec le modèle 5.

Pour conclure sur cette partie, intéressons-nous aux problèmes existants avec ce modèle. Commençons par le problème de déficience. Ce problème peut se résumer de la manière suivante, soit w(**e**) la somme de P(**f|e**) des phrases sources bien formé et i(**e**) la somme des phrases sources mal formée. On a alors un modèle déficient si w(**e**) + i(**e**) <1. Soit l'événement *échec* tel qu'on a l'égalité suivante w(**e**) + i(**e**) + P(*échec*|**e**) = 1. Ainsi, on retrouve la définition évidente qu'un modèle est déficient si P(*échec*|**e**)>0. Si P(*échec*|**e**) = 0 et que i(**e**) > 0, on dit que le modèle est spirituellement déficient.
Ainsi, si w(**e**) était indépendant de **e**, il n'y aurait pas de problème cependant, comme il n'existe aucune vraie contrainte à long terme, w(**e**) est inversement proportionnel à l. Le problème que l'on a alors est que dans le cas de la traduction automatisée, c'est-à-dire un cas ou on nous donne **f** et on cherche **e**, on se rend compte que P(**e**)P(**f|e**) est trop petit pour de longues phrases par rapport aux phrases courtes. Ainsi, le modèle semble favoriser les phrases courtes. On contourne le problème en introduisant une constante dans le calcul de P(**f|e**) mais cela ne corrige en rien le problème. Il faudrait améliorer la modélisation pour éviter ce problème.
De plus, on note un problème d'asymétrie des alignements. En effet, la notion de cept ici ne s'intéresse qu'à un seul mot par concept. Or cela devient un problème si on cherche à traduire certaines langues. En effet, la loi d'alignement choisit impose qu'un mot cible ne peut être aligné qu'a 1 ou 0 mot source. Par conséquent, dans le cas où un mot cible n'est pas traduisible en 1 seul mot source, le modèle ne sera pas en mesure de fournir une traduction correcte.

Comme ce dernier problème est assez majeur, on introduit le modèle par groupe de mots pour y remédier.

Modèles par groupes de mots

Pour résoudre les problèmes d'alignements avec les modèles par mots, nous nous intéressons maintenant au modèle qui choisissent comme unité de traduction les groupes de mots. Cependant, pour réaliser des alignements de groupes de mots, il nous faut deux niveaux d'alignement, un pour les mots et un pour les groupes de mots. Le niveau d'alignement entre les mots ne sera pas traité puisqu'elle correspond à l'alignement des modèles précédents.
Pour les alignements entre groupes de mots, nous introduisons le triplet z $(\tilde{F}, \tilde{E}, \tilde{A})$ nommé le **template d'alignement** :

- $\tilde{F}$: Classe bilingue de la phrase source. Compile le vocabulaire et on note $\mathcal{F}$ la fonction cartographie les mots de la classe.
- $\tilde{E}$: Classe bilingue de la phrase cible. Compile le vocabulaire et on note $\mathcal{E}$ la fonction cartographie les mots de la classe.
- $\tilde{A}$: la matrice binaire des alignements. Si un élément est égal à 1, alors les mots aux valeurs correspondantes sont alignés, 0 sinon. De la même manière que pour les modèles précédents, un mot source non aligné a un mot cible est aligné au mot NUL e_0.

Ainsi, un template d'alignement z est **applicable** à une séquence de mots source $\tilde{f}$ si les classes du template sont égales aux classes de la phrase source soit $\mathcal{F}(\tilde{f}) = \tilde{F}$. Par conséquent nous avons aussi l'égalité $\mathcal{E}(\tilde{e}) = \tilde{E}$. A noter que le template ne détermine pas les mots cibles mais établit juste des contraintes. Le choix des mots est réalisé avec un modèle statistique $p(\tilde{e}|z, \tilde{f})$ basé sur la loi de probabilité lexicale p(f|e) tel qu'on a les équations suivantes :

$$p\left(\tilde{f}\middle|\left(\tilde{F}, \tilde{E}, \tilde{A}\right), \tilde{e}\right) = \delta(\mathcal{E}(\tilde{e}), \tilde{E})\delta(\mathcal{F}(\tilde{f}, \tilde{F}) \prod_{j=1}^{l} p(f_j|\tilde{A}, \tilde{e}) \qquad (1)$$

$$p(f_j|\tilde{A},\tilde{e}) = \sum_{i=0}^{I} p(i|j;\tilde{A}) * p(f_j|e_i) \quad (2)$$
$$p(i|j;\tilde{A}) = \frac{\tilde{A}(i,j)}{\sum_i \tilde{A}(i,j)} \quad (3)$$

Attardons-nous maintenant sur la manière dont l'alignement par phrase est géré.
Soit f_1^J la phrase source et e_1^I la phrase cible, on commence par décomposée ces deux phrases en K séquences de sous-phrases tel que

$$f_1^J = \tilde{f}_1^K, \qquad \tilde{f}_k = f_{j_{k-1}+1}, \dots, f_{jk}$$
$$e_1^I = \tilde{e}_1^K, \qquad \tilde{e}_k = e_{i_{k-1}+1}, \dots, e_{ik}$$

On suppose qu'il n'y a qu'une seule segmentation possible pour simplifier les notations. Ainsi, soit $\tilde{a}_1^K$ entre la phrase source $\tilde{e}_1^K$ et la phrase cible $\tilde{f}_1^K$ on a :

$$P(f_1^J|e_1^I) = \sum_{\tilde{a}_1^K} \prod_{k=1}^{K} p(\tilde{a}_k|\tilde{a}_1^{k-1}, K) * p(\tilde{f}_k|\,\tilde{e}_{\tilde{a}_k}) \quad (4)$$

avec $p(\tilde{a}_k|\tilde{a}_1^{k-1}, K) = p(\tilde{a}_k|\tilde{a}_{k-1}, K)$ le modèle d'alignement d'ordre 1.
On introduit aussi le template d'alignement sous forme de variable :

$$p(\tilde{f}|\tilde{e}) = \sum_z p(z|\tilde{e}) * p(\tilde{f}|z,\tilde{e}) \quad (5)$$

avec $p(z|\tilde{e})$ la probabilité d'appliqué un template d'alignement.
L'entrainement des paramètres établis jusqu'à maintenant se fait ainsi de la manière suivante :

1. Entrainement de 2 modèles d'alignement HMM pour les traductions de f→e et e→f avec l'algorithme EM.
2. Pour chaque direction de traduction, on calcule les alignements de Viterbi que l'on nomme a_1^J et b_1^I. Pour améliorer la qualité de ces alignements, on crée une matrice d'alignement A=$A_1 \cap A_2$ avec A_1 = {(a_j,j)|j = 1….J} et A_2 = {(i,b_i)|i = 1…I}. L'idée est d'ajouter a A tous les couples (i,j) qui n'apparaissent que dans A_1 et A_2 et si ils ont un voisin déjà dans A ou si f_j et e_i ne sont aligné a aucun mot dans A. Par définition, (i,j) a comme voisins (i-1,j), (i+1,j), (i,j-1), (i,j+1).
3. On estime le lexique bilingue p(f|e) avec les fréquences relatives des alignements tel que
$$p(f|e) = \frac{n_A(f,e)}{n(e)} \quad (6)$$
avec n_A(f,e) la fréquence d'alignement du mot f avec e et n(e) la fréquence de e dans le corpus d'entrainement.
4. Déterminer les classes des mots sources et cibles. Pour cela, on utilise l'algorithme de maximum de ressemblance. Pour ce modèle, on maximise la probabilité conjointe d'un corpus d'entrainement bilingue (e_1^I,f_1^J) tel que :
$$(\tilde{\mathcal{E}},\tilde{\mathcal{F}}) = argmax_{\mathcal{E},\mathcal{F}} p(e_1^I|\mathcal{E}) * p(f_1^J|e_1^I;\mathcal{E},\mathcal{F}) \quad (7)$$
Pour modéliser le modèle monolingue, on utilise cette formule :
$$p(w_1^N|C) \coloneqq \prod_{i=1}^{N} p(C(w_i)|C(w_{i-1})) * p(w_i|C(w_i)) \quad (8)$$
avec w_1^N = w_1…w_n la séquence de mot que l'on cherche à traduire. Pour ce modèle, nous modélisons P(w_1^N) par un produit de bigramme tel que P(w_1^N) = $\prod_{i=1}^{N} p(w_i|w_{i-1})$.
Pour modéliser la probabilité de traduction, on suppose l'existence de l'alignement a_1^J tel que chaque mot f_j est produit par e_{aj} a la position a_j et à la probabilité p(f_j|e_{aj}) :
$$p(f_1^J|e_1^J) = \prod_{j=1}^{J} p(f_j|e_{aj}) \quad (9)$$

L'idée est d'utiliser a_1^J comme une variable cachée dans le modèle de traduction. Pour obtenir les paramètres on utilise l'algorithme de maximum de ressemblance et a^J_1 est l'alignement de Viterbi.

Avec ces nouvelles équations, on peut alors réécrire l'équation (8) :

$$p(f_1^J|e_1^I;\mathcal{E},\mathcal{F}) = \prod_{j=1}^{J} p\left(\mathcal{F}(f_j)\right)\left|\mathcal{E}(e_{aj})\right) * p(f_j|\mathcal{F}(f_j)) \qquad (10)$$

En utilisant une fonction $n_t(F|E)$ qui compte la fréquence les mots de la classe F sont alignés avec les mots de la classe E, on sait que :

$$P(F|E) = n_t(F|E)(\sum_F n_t(F|E))$$

$$p(f|F) = n_t(f)/(\sum_f n_t(f|F)) = \frac{n_t(f)}{\sum_E n_t(F|E)}$$

tel qu'en appliquant ces équations aux équations (10) et (7) et en appliquant un logarithme négatif et en changeant l'ordre de sommation de l'équation (7), nous obtenons le critère d'optimisation suivant :

$$-\sum_{E,E'} h\big(n(E|E')\big) - \sum_{E,F} h\big(n_t(F|E)\big) + 2\sum_E h(n(E)) + \sum_F h(\sum_E n_t(F|E)) + \sum_E h(n_t(F|E)$$

avec E' la classe prédécesseur de E. En combinant les 2 fonctions de comptes n(E|E') et n_t(F|E) en n_g(X|Y) := n(X|Y)+n_t(X|Y) comme tous les mots f, e et e' vérifient n(f|e)=0 et n_t(e|e') = 0 on a alors le critère d'optimisation suivant :

$$LP_2\left((\mathcal{E},\mathcal{F}),n_g\right) = -\sum_{X,X'} h(n_g(X|X')) + \sum_X h\left(n_{g,1}(X)\right) + \sum_X h(n_{g,2}(X)) \qquad (11)$$

tel que $\left(\tilde{\mathcal{E}},\tilde{\mathcal{F}}\right) = argmax_{\mathcal{E},\mathcal{F}} LP_2((\mathcal{E},\mathcal{F}),n_g)$ (12)

On définit $n_{g,1}$(X) = $\sum_{X'} n_g(X|X')$ et $n_{g,2}$(X) = $\sum_{X'} n_g(X'|X)$ et X traversant les classes de $\mathcal{E}$ *et* $\mathcal{F}$.

5. Compter toutes les phrases paires du corpus d'entrainement qui sont cohérents avec la matrice d'alignement déterminer à l'étape 2. On dit qu'une paire est cohérente avec la matrice si les mots sources de la paire sont seulement alignés avec les mots cible de la paire. On peut ainsi obtenir une fonction compte n(z) qui nous indique la fréquence d'occurrence du template d'alignement dans le corpus d'entrainement aligné. On estime donc la probabilité d'utiliser un template d'alignement par :

$$p\big(z = \big(\tilde{F},\tilde{E},\tilde{A}\big)\big|\tilde{e}\big) = \frac{n(z)*\delta(\tilde{E},\mathcal{E}(\tilde{e}))}{n(\mathcal{E}(\tilde{e}))} \qquad (13)$$

Cette équation nous est utile notamment pour l'équation (5).

Maintenant que nous avons vu comment réaliser l'alignement entre groupe de mots, intéressons maintenant à comment on traduit ces groupes de mots.
Pour la traduction, nous réutilisons l'équation (7) simplifier sous la forme suivante :

$$argmax_{e_1^I}\{p(e_1^I)p\big(e_1^I\big|f_1^J\big)\} \qquad (14)$$

Cependant, comme cet espace de recherches est tres large, nous appliquons 2 étapes de préprocessing :

1. Determiner tous les sets de phrases sources dans $\tilde{f}$ pour lequel il existe un template d'alignement applicable. On nomme l'*instanciation de template d'alignement* (alignment template instantiation) toutes les applications possible d'un template sur une sous-phrase de la phrase source.
2. On segmente la phrase d'entrée. On cherche la séquence de phrase $f_1^J = \tilde{f}_1 \circ \dots \circ \tilde{f}_k$ avec :

$$argmax_{\tilde{f}_1 \circ \dots \circ \tilde{f}_k} \prod_{k=1}^{K} max_z p(z|\tilde{f}_k) \qquad (15)$$

Une fois ces deux étapes réalisées, nous pouvons commencer le processus de traduction. Pour ce faire, nous établissons des hypothèses partielles avec les informations suivantes :

1. Le dernier mot cible produit
2. L'état du modèle de langage (les classes des 4 derniers mots cibles)
3. Un vecteur de bit représentant les positions déjà prises de la phrase source
4. Une référence à l'instanciation du template d'alignement qui a produit le dernier mot cible
5. La position du dernier mot cible dans l'instanciation du template d'alignement
6. Les coûts accumulés (le logarithme négatif des probabilités) de toutes les décisions précédentes
7. Une référence à la dernière hypothèse partielle.

Pour cette partie, nous avons fait le choix de présenter l'origine du modèle par groupe de mots réaliser par Och en 1999. En ce qui concerne les systèmes actuelles de la littérature, nous avons le logiciel opensource Moses par Philipp Koehn et al,. en 2007 qui est le modèle le plus avancé en ce qui concerne les modèles par groupes de mots.

Les modèles statistiques syntaxiques

Ces approches récentes ont comme potentielle de pouvoir résoudre plusieurs problèmes de la traduction par groupes de mots. Ces erreurs proviennent de manque d'informations. En effet, pour certaines paires de langue, la plupart des réordonnements ne sont motivés que par la syntaxe. Ainsi sans ces informations, les modèles ne peuvent pas généraliser. De plus, ces systèmes peuvent conserver les dépendances sur les groupes de mots pertinents et assurer une cohérence entre les choix de traduction.
Pour permettre cela, un modèle peut prendre comme entrée un arbre tel que la phrase d'entrée a été traiter par un analyseur syntaxique (*syntactic parser*). Ainsi, soit $\mathcal{E}$ l'arbre de la phrase cible et f la phrase source. $\mathcal{E}$ contient des nœuds $\mathcal{E}_1 \dots \mathcal{E}_n$ et f contient les mots $f_1,\dots,f_m$. On a les 3 opérations applicables sur les nœuds suivants :

- **N** : Insertion, cette variable aléatoire gère les insertions de mots avant et après chaque nœud. Elle décide aussi du mot à insérer.
- **R** : Réordonner, a pour but de changer l'ordre des nœuds fils. Cette opération ne s'applique que aux nœuds intermédiaires.
- **T** : Traduction, cette opération traduit la feuille de l'arbre dans la langue source. Cette opération ne s'applique que aux nœuds finaux.

On introduit ainsi $\theta = \langle \upsilon, \rho, \tau \rangle$ qui contient les valeurs <**N**,**R**,**T**> tel que $\theta_i = \langle \upsilon_i, \rho_i, \tau_i \rangle$ le set de valeurs associé à ε_i et $\boldsymbol{\theta} = \theta_1, \dots, \theta_n$ l'ensemble des valeurs des variables aléatoires associé a $\mathcal{E}$.
On a alors la probabilité d'avoir une phrase source f en connaissant un arbre $\mathcal{E}$:

$$p(f|\mathcal{E}) = \sum_{\boldsymbol{\theta}: Str(\boldsymbol{\theta}(\mathcal{E}))=f} p(\boldsymbol{\theta}|\mathcal{E})$$

avec Str($\boldsymbol{\theta}(\mathcal{E})$ la suite de mots feuilles de l'arbre transformé par $\boldsymbol{\theta}$.

On a alors la probabilité d'avoir une suite en particulier de variables aléatoires dans l'arbre est :

$$p(\boldsymbol{\theta}|\mathcal{E}) = p(\theta_1, \dots, \theta_n|\varepsilon_1, \dots, \varepsilon_n) = \prod_{i=1}^{n} p(\theta_i|\theta_1, \dots, \theta_{i-1}, \varepsilon_1, \dots, \varepsilon_n)$$

On suppose cependant que les θ_i sont indépendants entre eux mais qu'ils dépendent tous de caractéristiques du nœud ε_i.

$$p(\theta_i|\varepsilon_i) = p(\upsilon_i, \rho_i, \tau_i|\varepsilon_i)$$
$$p(\theta_i|\varepsilon_i) = p(\upsilon_i|\varepsilon_i) * p(\rho_i|\varepsilon_i) * p(\tau_i|\varepsilon_i)$$
$$p(\theta_i|\varepsilon_i) = p(\upsilon_i|\mathcal{N}(\varepsilon_i)) * p(\rho_i|\mathcal{R}(\varepsilon_i)) * p(\tau_i|\mathcal{T}(\varepsilon_i))$$
$$p(\theta_i|\varepsilon_i) = n(\upsilon_i|\mathcal{N}(\varepsilon_i)) * r(\rho_i|\mathcal{R}(\varepsilon_i)) * t(\tau_i|\mathcal{T}(\varepsilon_i))$$

avec $\mathcal{N}, \mathcal{R}, \mathcal{T}$ les caractéristiques intéressantes de **N**, **R**, et **T**. Pour la dernière ligne, on note un changement de notation avec $n(\upsilon|\nu), r(\rho|\mathrm{r}) et\ t(\tau|\mathrm{t})$ qui représentent les paramètres du modèle et où $\nu, \mathrm{r}\ et\ \mathrm{t}$ sont des valeurs possibles pour $\mathcal{N}, \mathcal{R}\ et\ \mathcal{T}$.
On arrive donc à la forme finale :

$$P(f|\mathcal{E}) = \sum_{\boldsymbol{\theta}:Str(\boldsymbol{\theta}(\mathcal{E}))=f} \prod_{i=1}^{n} n(\upsilon_i|\mathcal{N}(\varepsilon_i)) * r(\rho_i|\mathcal{R}(\varepsilon_i)) * t(\tau_i|\mathcal{T}(\varepsilon_i))$$

Maintenant que nous avons identifié les paramètres, examinons comment les entrainer. La méthode est similaire qu'aux autres modèles puisqu'on utilise toujours l'algorithme de maximum de ressemblance. On peut résumer la méthode de la manière suivante :

1. Initialiser toutes les tables de probabilités : $n(\upsilon|\nu), r(\rho|\mathrm{r}) et\ t(\tau|\mathrm{t})$. La méthode de distribution semble libre.
2. Réinitialiser les compteurs $c(\upsilon|\nu), c(\rho|\mathrm{r}) et\ c(\tau|\mathrm{t})$. On considère qu'un événement est la paire d'une valeur de variable aléatoire et d'une valeur de caractéristique.
3. Pour chaque pair $\langle \mathcal{E}, f \rangle$, pour tout **θ** tel que f = Str(**θ**($\mathcal{E}$) et pour i allant de 1 à n, on met à jour les fonctions de comptes avec cnt = $p(\boldsymbol{\theta}|\mathcal{E}) / \sum_{\boldsymbol{\theta}:Str(\boldsymbol{\theta}(\mathcal{E}))=f} p(\boldsymbol{\theta}|\mathcal{E})$
4. Pour chaque $\langle \upsilon, \mathcal{N} \rangle, \langle \rho, \mathcal{R} \rangle\ et\ \langle \tau, \mathcal{T} \rangle$:

$$n(\upsilon|\mathcal{N}) = \frac{c(\nu, \mathcal{N})}{\sum_{\upsilon} c(\upsilon, \mathcal{N})}$$

$$r(\rho|\mathcal{R}) = \frac{c(\rho, \mathcal{R})}{\sum_{\rho} c(\rho, \mathcal{R})}$$
$$t(\tau|\mathcal{T}) = \frac{c(\tau, \mathcal{T})}{\sum_{\tau} c(\tau, \mathcal{T})}$$

5. Réitérer les étapes 2-4.

Conclusion

En conclusion, il serait intéressant de comparer les résultats des différentes approches. Selon les différentes littératures existantes, les modèles à groupes de mots sont à favoriser de par leur taux de réussite supérieur au modèle a mots. En effet, les modèles a mots ont généralement un taux de succès de l'ordre de 42%. Les écarts entre les 2 approches en utilisant les mêmes données d'entrainements et d'évaluation et les mêmes modèles de langages on constate une différence d'environ 8 points. En ce qui concerne les modèles syntaxiques, on constate que ces modèles se débrouillent mieux que les modèles a groupes de mots de 8 points aussi. Je n'ai cependant pas trouvé d'article comparant les modèles syntaxiques aux modèles a groupes de mots permettant de départager les deux. Ainsi, même si les approches présentées sont prometteuses, nous sommes loin d'une utilisation grand public. Des approches ont été explorées pour essayer de mieux adapter les systèmes de reconnaissance à la traduction.

Bibliographie

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

Franz Josef Och. 1999. An efficient method to determine bilingual word classes. In *EACL'99: Ninth Conf. of the Europ. Chapter of the Association for Computational Linguistics*, Bergen, Norway, June.

Kenji Yamada and Kevin Knight. A syntax-based statistical translation model. In *Proc. of the Meeting of the Association for Computational Linguistics*, pages 523–530, Toulouse, France, 2001.

Daniel Déchelotte. Traduction automatique de la parole par méthodes statistiques. Université de Paris-Sud, 17 décembre 2007

Lambros CRANIAS, Harris PAPAGEORGIOU, Stelios PIPERIDIS. A matching technique in example-based machine translation. Institute for Language and Speech Processing, GREECE 10 aug 1995.

Toshiaki Nakazawa, Kun Yu, Daisuke Kawahara, Sadao Kurohashi. Example-based Machine Translation based on Deeper NLP. Graduate School of Information Science and Technology University of Tokyo. January 2006