Introduction aux algorithmes de recherche

(cas de la traduction probabiliste)

But: introduction à la recherche de solutions dans un espace trop grand pour que l'on puisse le parcourir complètement.

Trois approches classiques sont illustrées dans le cadre de la traduction automatique:

- Recherche par programmation dynamique avec filtrage ($beam\ search$)
- Recherche gloutonne (par hill-climbing)
- Recherche A^*

Rappelez-vous du canal bruité

Problème: on cherche la traduction $\hat{e}$ d'une phrase source f:

$$\begin{aligned}
\hat{e} &= \text{argmax}_{e \in \mathcal{E}}\, p(e|f) \\
&= \text{argmax}_{e \in \mathcal{E}}\, \frac{p(f|e) \times p(e)}{p(f)} \\
&= \text{argmax}_{e \in \mathcal{E}}\, \underbrace{p(f|e)}_{\text{traduction}} \times \underbrace{p(e)}_{\text{langue (cible)}}
\end{aligned}$$

Préambule: Cette maximisation est une opération NP-complète (si on ne présuppose pas d'ordre à priori sur les mots) Knight [1999]

↪ **Problème:** comment trouver $\hat{e}$ sans énumérer toutes les phrases cibles e possibles ?

Décodage: l'espoir fait vivre

- **Tentative:** On peut associer à chaque mot source d'une phrase de n mots un ensemble de m mots probables. Et on construit toutes les phrases possibles avec ces mots ($\Rightarrow m^n$ "phrases" candidates) que l'on soumet au modèle probabiliste.

$\hookrightarrow$ Si $n = 10$ et $m = 3$, ça fait déjà $59\,049$ phrases à noter. . .

- Pire, une traduction ne se fait pas mot à mot, et l'ordre des mots dans la traduction peut ne pas suivre celui des mots sources.

$\hookrightarrow$ Il faut autoriser des permutations de ces phrases ($\Rightarrow (m^n)!$).

Bref, ça ne marchera pas...

Décodage = maximisation

- Cas d'un modèle de langue bigramme et d'un modèle de traduction de type IBM2, pour la traduction d'une phrase source f_1^J:

$$\hat{e}_1^{\hat{I}} = \underset{I}{\operatorname{argmax}} \underbrace{p(J|I)}_{\text{longueur}} \underset{e_1^I}{\operatorname{argmax}} \underbrace{\prod_{i=1}^{I} p(e_i|e_{i-1})}_{p(e_1^I)} \underbrace{\sum_{a_1^J} \prod_{j=1}^{J} p(a_j|j,J,I)p(f_j|e_{a_j})}_{p(f_1^J|e_1^I)}$$

- Le processus correspond au choix d'une longueur cible optimale $\hat{I}$, puis d'une série de choix d'alignements a_j et de choix lexicaux e_{a_j}.

Notez les dépendances dans cette équation, et le modèle de traduction inversé.

Décodage à la Tillmann et al. [1997]

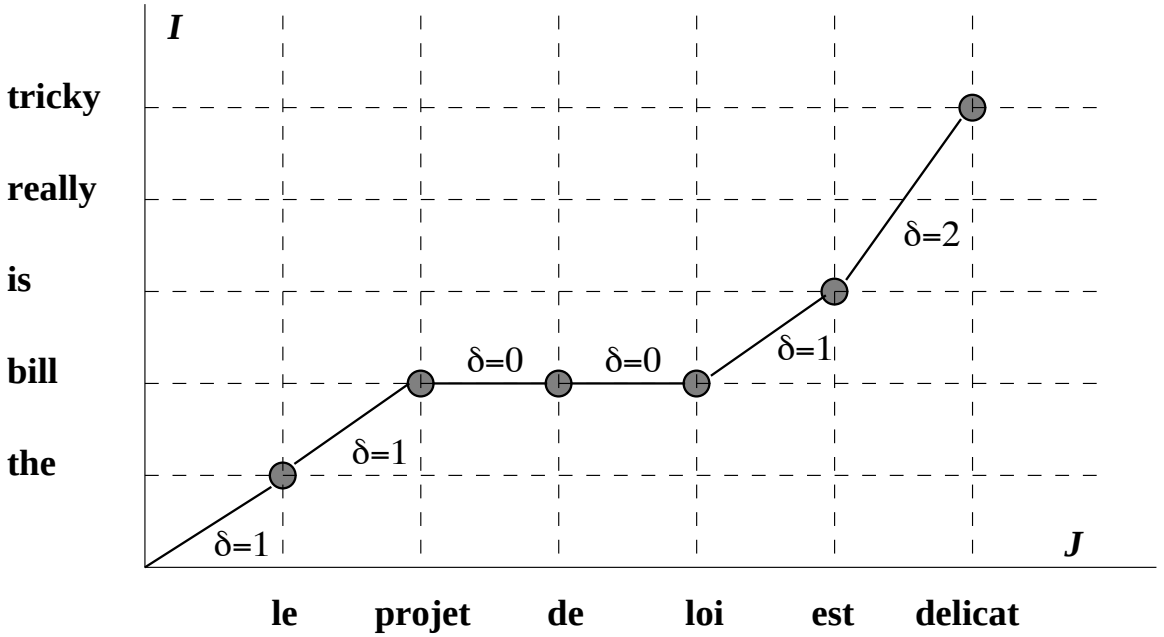
- **Choix:** le modèle d'alignement est un modèle markovien d'ordre 1[1]; les alignements produits sont monotones.
- Plus précisément, ils imposent que $a_j = a_{j-1} + \delta$, avec $\delta \in \{0, 1, 2\}$. Ceci est une approximation, mais des opérations de prétraitement sont possibles pour en minimiser l'impact.
- Le critère que l'on cherche à maximiser est:

$$\hat{e}_1^{\hat{I}} = \underset{I, e_1^I}{\operatorname{argmax}} \left\{ \prod_{i=1}^{I} p(e_i|e_{i-1}) \max_{a_1^J} \prod_{j=1}^{J} \Big[p(a_j|a_{j-1}) p(f_j|e_{a_j}) \Big] \right\}$$

- **Notez** que la somme sur tous les alignements a été remplacée par une maximisation. On parle souvent de *maximum approximation*.

[1] Rappelez-vous que le modèle d'alignement de IBM2 est un modèle markovien d'ordre 0).

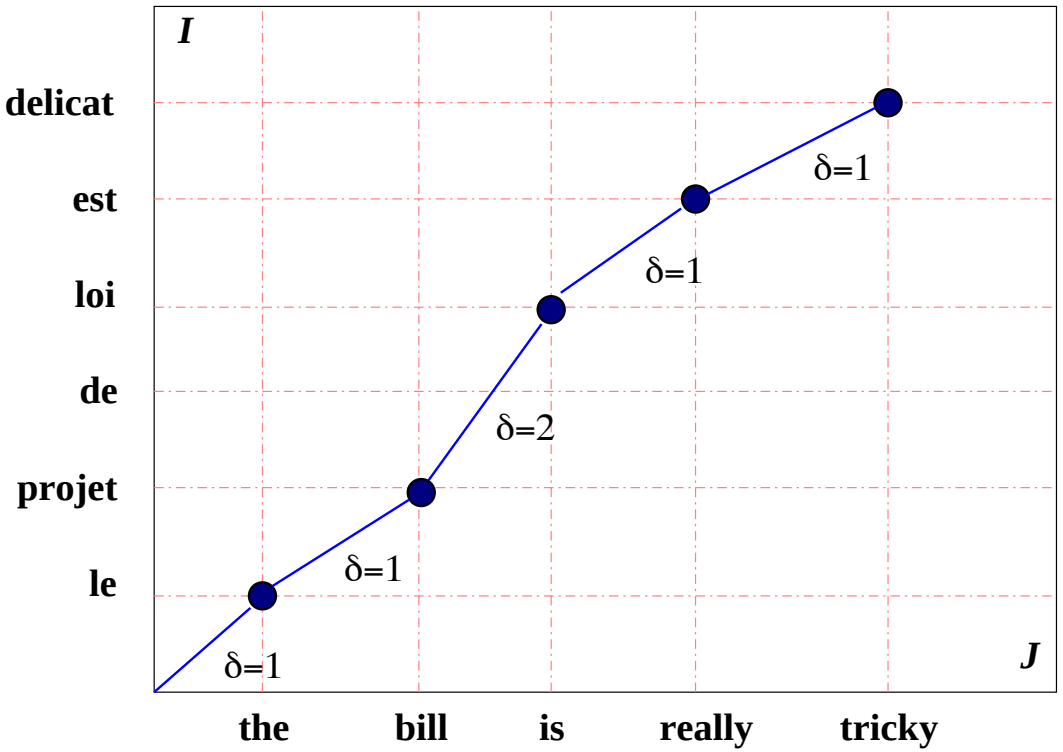
Exemple d'alignement monotone



Alignement sous-jacent: $a = \{1, 2, 2, 2, 3, 5\}$, cad:

(the,le) (bill,projet de loi) (is,est) (really,ϵ) (tricky,délicat)

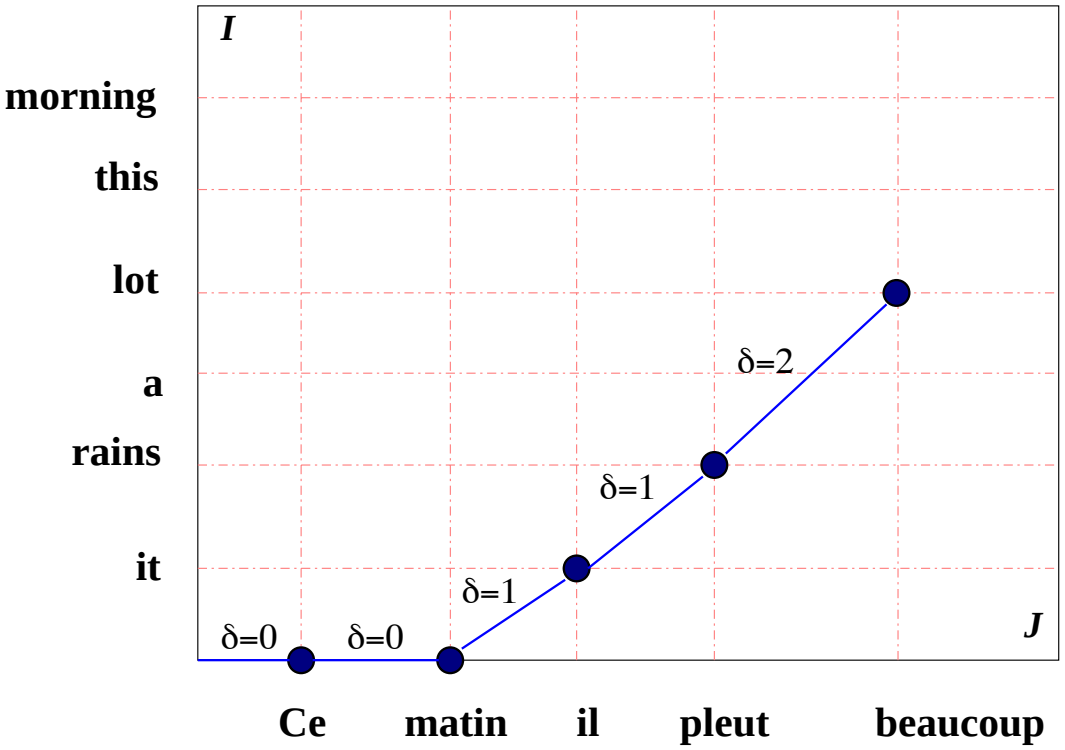
Changeons la direction (traduction vers le français)



Alignement sous-jacent: $a = \{1, 2, 4, 5, 6\}$, cad:

(le,the) (projet,bill) (de,ϵ) (loi,is) (est,really) (délicat,tricky)

Et si les deux phrases n'étaient pas synchronisées...



Alignement sous-jacent: $a = \{0, 0, 1, 2, 4\}$, cad:
(ϵ,ce),(ϵ,matin),(it,il),(rains,pleut),(a,ϵ),(lot,ϵ)

Décodage Tillmann et al. [1997]

- Synchronisation monotone $\Rightarrow$ décodage très rapide. **Mais:** nécessite des prétraitements:

Il parle depuis la maison blanche	phrase initiale
Il parle depuis la blanche maison	pré-traitement
He speaks from the white house	traduction

- Reformulons notre modèle de langue de manière à exprimer $\prod_{i=1}^{I} p(e_i|e_{i-1})$ comme un produit $\prod_{j=1}^{J} p_\delta(e_{a_j}|e_{a_{j-1}})$ où $\delta = a_j - a_{j-1}$

Décodage Tillmann et al. [1997]

$\delta = 1$, (cas le plus "normal"), $a_j = a_{j-1} + 1$:
alors $p_{\delta=1}(e|e') = p_\delta(e_{a_j}|e_{a_{j-1}}) = p(e|e')$

$\delta = 2$ (un mot cible sans mot source associé), $a_j = a_{j-1} + 2$:
alors $p_{\delta=2}(e|e') = p_\delta(e_{a_j}|e_{a_{j-1}}) = \max_{\bar{e}} p(\bar{e}|e')p(e|\bar{e})$
Note: la maximisation est *a priori* à faire sur l'ensemble du vocabulaire cible ! ($\rightarrow$ précalcul)

$\delta = 0$ (un mot cible est associé à deux ou plusieurs mots sources):
alors $p_{\delta=0}(e|e') = \begin{cases} 1 & \text{si } e = e' \\ 0 & \text{sinon} \end{cases}$

Le critère de recherche devient donc:

$$\hat{e}_1^{\hat{I}} = \underset{I, e_1^I, a_1^J}{\operatorname{argmax}} \prod_{j=1}^{J} \left[p(a_j|a_{j-1}) p_{[a_j - a_{j-1}]}(e_{a_j}|e_{a_{j-1}}) p(f_j|e_{a_j}) \right]$$

Décodage Tillmann et al. [1997]

- En chaque position de l'axe vertical (cible), tout mot e est possible (a $priori$). Le choix d'un alignement monotone et d'un modèle de langue bigramme introduit une dépendance d'ordre 1 entre les différents points du treillis de recherche.

- Soit $Q(i, j, e)$ la probabilité du meilleur chemin arrivant au point (i, j, e). Notre dépendance d'ordre 1 nous permet d'exprimer la récurrence:

$$Q(i, j, e) = p(f_j|e) \max_{\delta} p(i|i - \delta) \max_{e'} p_\delta(e|e')Q(i - \delta, j - 1, e')$$

- La traduction est alors donnée par $\max_{i,e} Q(i, J, e)$ en retournant un pointeur arrière (idem à viterbi).

Décodage Tillmann et al. [1997]

Input: $f_1 \dots f_j \dots f_J$
Initialisation du treillis de décodage

for all position source $j = 1, 2, \dots, J$ **do**
  for all position cible $i = 1, 2, \dots, I_{max}$ **do**
    for all mot cible e du vocabulaire **do**

$$Q(i,j,e) = \log p(f_j|e) + \max_{\delta,e'} \begin{cases} \log p(i|i-\delta) + \\ \log p_\delta(e|e') + \\ Q(i-\delta, j-1, e') \end{cases}$$

Retour de la meilleure solution depuis: $\max_{i,e} Q(i, J, e)$
Output: $e_1 \dots e_i \dots e_{\hat{I}}$

Complexité: $J \times I_{max} \times |\mathcal{E}|^2$, où $\mathcal{E}$ est la taille du vocabulaire cible considéré, I_{max} est la taille cible maximale considérée, et J la longueur de la phrase source.

Est-ce que ça marche ?

- Difficile de le dire... sans le coder !
- Quelques exemples (probablement choisis :-) où O indique la phrase source, R la traduction de référence et A la traduction automatique:

O	Por favor, reservamos dos habitaciones dobles con cuarto de baño
R	We booked two double rooms with a bathroom
A	We booked two double rooms with a bathroom, please
O	Quisiera que nos despertaran mañana a las dos y cuarto, por favor
R	I would like you to wake up us tomorrow at a quarter past two, please
A	I want you to wake up us up tomorrow at a quarter past two, please
O	Repáseme la cuenta de la habitación ochocientos vientiuno
R	Could you check the bill for room number eight two one for me, please ?
A	Check the bill for room number eight two one.

Décodage Nießen et al. [1998]

- Même modèle, mais pas d'hypothèse de monotonie (plus long)
- Search inversé: on avance le long du texte cible en couvrant progressivement les positions sources.
- Choix: un mot cible e_i est aligné avec au plus l mots sources consécutifs (sorte de fertilité non modélisée).
- Le critère à optimiser devient alors (sous approximation maximale):

$$\underset{I}{\operatorname{argmax}} \left[p(J|I) \underset{e_1^I}{\operatorname{argmax}} \prod_{i=1}^{I} \left[p(e_i|e_{i-1}) \max_{j,l} \prod_{\bar{j}=j-l+1}^{j} p(i|\bar{j}, J, I) p(f_{\bar{j}}|e_i) \right] \right]$$

Décodage Nießen et al. [1998]

Soit $Q_I(c, i, j, e)$ la probabilité du meilleur chemin arrivant en position i dans e_1^I, en position j (dans f_1^J) et tel que $(e_i = e)$ et c positions sources ont été couvertes.

Définissons cette quantité récursivement:

- **Cas simple:** le mot e_i n'a pas de correspondant dans le texte source $(skip)$.

$$Q_I^S(c, i, j, e) = \max_{e'} \{p(e|e')Q_I(c, i-1, j, e')\}$$

↪ On recherche dans la table la meilleure façon d'arriver à (c,i,j,e) depuis une cellule précédente.

Décodage Nießen et al. [1998]

- **Cas général:** le mot e_i est associé à l mots sources ($l > 0$).

$$Q_I^N(c,i,j,e) = \begin{array}{l} \max_{l>0} \left[\prod_{\bar{j}=j-l+1}^{j} \left\{ p(i|\bar{j},J,I).p(f_{\bar{j}}|e_i) \right\} \right. \\ \max_{e'} \left\{ p(e|e'). \right. \\ \left. \left. \max_{j'} \left[Q_I(c-l,i-1,j',e').v(c,l,j',j,e') \right] \right\} \right] \end{array}$$

où $v(c,l,j',j,e')$ est un prédicat qui retourne 1 si les l positions de $\bar{j}$ à f_j sont libres dans la chaîne source; 0 sinon.

↪ En gros, on cherche la meilleure fertilité, et la meilleure position source libre (via v).

Note: les choix l, j' et e' sont mémorisés pour pouvoir à la fin reconstruire la traduction "optimale".

Décodage Nießen et al. [1998]

- Nous avons alors tous les ingrédients de notre récursion:

$$Q_I(c,i,j,e) = \max\left\{Q_I^S(c,i,j,e), Q_I^N(c,i,j,e)\right\}$$

- et la traduction est donnée par:

$$\max_I \left\{ p(J|I). \max_{j,e} Q_I(J,I,j,e) \right\}$$

- Compléxité $\mathcal{O}(I_{max}^2.J^3.|\mathcal{E}|^2)$ où $|\mathcal{E}|$ est la taille du vocabulaire cible.

- Rappel: le décodeur de Tillmann et al. [1997] est en $\mathcal{O}(J \times I_{max} \times |\mathcal{E}|^2)$

Décodage Nießen et al. [1998]: algorithme simplifié

Input: $f_1 \ldots f_j \ldots f_J$; f_{max}, fertilité maximale
Initialize $Space$; Select I_{max}; Compute the active vocabulary

for all target position $i = 1, 2, \ldots, I_{max}$ **do**
  prune$(i-1)$;
  for all alive hyp. $h = Space(i, j, c, e)$ **do**
    $uv \leftarrow$ History(h);
    $zones \leftarrow$ FreeSrcPositions(h);
    $bestWords \leftarrow$ NBestTgtWords(uv);
    for all w in $bestWords$ **do**
      $prob \leftarrow$ Score$(h) + \log p(w|uv)$;
      setIfBetter$(i, j, c, b, prob, 0, j, v)$;
      for all free source position d **do**
        $s \leftarrow prob$;
        for all $f \in [1, f_{max}]$ / $d + f - 1$ is free **do**
          $s+ = \log a(i|d, J) + \log t(f_d|e_i)$;
          setIfBetter$(i, d, c + f, w, s, f, j, v)$;

Décodage Nießen et al. [1998]: algorithme

// recherche de la meilleure solution

$max_s \leftarrow -\infty$

for all $i \in [1, I_{max}]$ **do**

  for all alive hyp. $h = Space(i, j, c, e)$ **do**

    $s \leftarrow Score(h) + p(J|i)$;

    if $((c == J)$ and $(s > max_s))$ **then**

      $max_s \leftarrow s$

      $\langle max_i, max_j, max_e \rangle \leftarrow \langle i, j, e \rangle$

// Backtrack si une hypothèse a survécu

if $(max_s! = \infty)$ **then**

  Return $Space(max_i, max_j, J, max_e)$;

else

  Failure

Output: $e_1 \ldots e_i \ldots e_{max_i}$

Décodage Nießen et al. [1998]: légende

- Pour l'algorithme précédent:

  $Space(i, j, c, e)$ contient les hypothèses qui alignent $e_i = e$ à f_j et qui couvrent c mots sources

  $setIfBetter(i, j, c, e, p, f, bj, be)$ réalise l'opération de maximisation sur l'hypothèse $< i, j, c, e >$ si p est meilleur que le score précédant

  $< b_j, b_e >$ information de retour-arrière (de quel mot f_{b_j} vient-on ?, quel est le mot b_e anglais précédent ?).

- Pour l'illustration qui suit:
  - Une case de la table d'analyse représente toutes les hypothèses alignant la position cible i à la position source j. Un item
  - $\langle e_i, \text{c, s, j', e'} \rangle \equiv$ le mot cible e_i posé à la position i, la couverture source c, le score s associé à l'hypothèse (ici -logprob); j' est la position source de retour, e' le mot duquel on vient.

Décodage Nießen et al. [1998]: exemple !

Source sentence: la loi anti-tabac

la → [the] [of] [to] [house] [in] [it] [on] [for] [a] [motion] ['s] [madam] . . . (944)

---

loi → [bill] [act] [legislation] [law] [bills] [clause] [it] [.] [passed] [this] [statute] [legislative]. . . (341)

---

anti-tabac → [anti-smoking] [non-smoking] [tobacco] [no-smoking] [anti-tobacco] (5)

Calcul du vocabulaire actif (les mots cibles les plus pertinents selon $\text{argmax}_e\, p(f|e)p(e)$):

$\Longrightarrow$'s . a act anti-smoking anti-tobacco bill bills clause for house in it law legislation legislative madam motion no-smoking non-smoking of on passed statute the this to tobacco (28)

Décodage Nießen et al. [1998]: le premier mot

⟨anti-smoking, 1, 151, 0, bos⟩ ⟨anti-tobacco, 1, 281, 0, bos⟩ ⋮ ⟨legislation, 1, 23, 0, bos⟩ ⟨the, 1, 21, 0, bos⟩		
la	loi	anti-tabac

$$\begin{array}{lll}
\log p(\text{anti-smoking}|BOS) & + \log p(\text{la}|\text{anti-smoking}) & + \log p(1|1,3) \\
\log p(\text{anti-tobacco}|BOS) & + \log p(\text{la}|\text{anti-tobacco}) & + \log p(1|1,3) \\
 & \vdots & \\
\log p(\text{legislation}|BOS) & + \log p(\text{la}|\text{legislation}) & + \log p(1|1,3) \\
\log p(\text{the}|BOS) & + \log p(\text{la}|\text{the}) & + \log p(1|1,3)
\end{array}$$

$$(BOS = Begin\ Of\ Sentence)$$

Décodage Nießen et al. [1998]: fertilité n

⟨anti-smoking, 1, 151, 0, bos⟩ ⟨anti-smoking, 2, 281, 0, bos⟩ ⋮ ⟨the, 1, 21, 0, bos⟩ ⟨the, 2, 51, 0, bos⟩ ⟨the, 3, 127, 0, bos⟩		
la	loi	anti-tabac

$$\log p(\text{the}|BOS) + \overbrace{\log p(\text{la}|\text{the}) + \log p(1|1,3)}^{a}$$

$$\log p(\text{the}|BOS) + a + \overbrace{\log p(\text{loi}|\text{the}) + \log p(1|2,3)}^{b}$$

$$\log p(\text{the}|BOS) + a + b + \log p(\text{anti-tabac}|\text{the}) + \log p(1|3,3)$$

Décodage Nießen et al. [1998]: La première ligne

⟨anti-smoking, 1, 151, 0, bos⟩ ⟨anti-smoking, 2, 281, 0, bos⟩ ⟨anti-smoking, 3, 286, 0, bos⟩ ⋮ ⟨anti-tobacco, 1, 151, 0, bos⟩ ⟨anti-tobacco, 2, 281, 0, bos⟩ ⟨anti-tobacco, 3, 287, 0, bos⟩ ⟨legislation, 1, 23, 0, bos⟩ ⟨legislation, 2, 34, 0, bos⟩ ⟨legislation, 3, 154, 0, bos⟩ ⟨the, 1, 21, 0, bos⟩ ⟨the, 2, 51, 0, bos⟩ ⟨the, 3, 127, 0, bos⟩	⟨anti-smoking, 1, 153, 0, bos⟩ ⟨anti-smoking, 2, 159, 0, bos⟩ ⟨anti-tobacco, 1, 154, 0, bos⟩ ⋮ ⟨anti-tobacco, 2, 159, 0, bos⟩ ⟨legislation, 1, 17, 0, bos⟩ ⟨legislation, 2, 147, 0, bos⟩ ⟨the, 1, 18, 0, bos⟩ ⟨the, 2, 148, 0, bos⟩ ⟨bill, 1, 11, 0, bos⟩	⟨anti-smoking, 1, 29, 0, bos⟩ ⟨anti-tobacco, 30, 1, 0, bos⟩ ⟨legislation, 1, 143, 0, bos⟩ ⋮ ⟨the, 1, 133, 0, bos⟩ ⟨no-smoking, 1, 26, 0, bos⟩
la	loi	anti-tabac

$$\log p(\text{the}|BOS) + \log p(\text{anti-tabac}|\text{the}) + \log p(1|3,3)$$

$$\log p(\text{legislation}|BOS) + \log p(\text{loi}|\text{legislation}) + \log p(1|2,3)$$

Décodage Nießen et al. [1998]: Le deuxième mot

	⟨anti-smoking, 2, 153, 1, the⟩ ⟨anti-tobacco, 2, 157, 1, the⟩ ⟨legislation, 2, 18, 1, the⟩	
⟨anti-smoking, 1, 151, 0, bos⟩ ⟨anti-smoking, 2, 281, 0, bos⟩ ⟨anti-smoking, 3, 286, 0, bos⟩ ⟨anti-tobacco, 1, 151, 0, bos⟩ ⟨anti-tobacco, 2, 281, 0, bos⟩ ⟨anti-tobacco, 3, 287, 0, bos⟩ ⟨legislation, 1, 23, 0, bos⟩ ⟨legislation, 2, 34, 0, bos⟩ ⟨legislation, 3, 154, 0, bos⟩ ⟨the, 1, 21, 0, bos⟩ ⟨the, 2, 51, 0, bos⟩ ⟨the, 3, 127, 0, bos⟩ . . .	⟨anti-smoking, 1, 153, 0, bos⟩ ⟨anti-smoking, 2, 159, 0, bos⟩ ⟨anti-tobacco, 1, 154, 0, bos⟩ ⟨anti-tobacco, 2, 159, 0, bos⟩ ⟨legislation, 1, 17, 0, bos⟩ ⟨legislation, 2, 147, 0, bos⟩ ⟨the, 1, 18, 0, bos⟩ ⟨the, 2, 148, 0, bos⟩ ⟨bill, 1, 11, 0, bos⟩ . . .	⟨anti-smoking, 1, 29, 0, bos⟩ ⟨anti-tobacco, 30, 1, 0, bos⟩ ⟨legislation, 1, 143, 0, bos⟩ ⟨the, 1, 133, 0, bos⟩ ⟨no-smoking, 1, 26, 0, bos⟩ . . .
la	loi	anti-tabac

$21 + \log p(\text{anti-smoking}|\text{the}, BOS) + \log p(\text{loi}|\text{anti-smoking}) + \log p(2|2,3)$

$\log p(\text{anti-tabacco}|\text{the}, BOS) + \log p(\text{loi}|\text{anti-tabacco}) + \log p(2|2,3)$

$\log p(\text{legislation}|\text{the}, BOS) + \log p(\text{loi}|\text{legislation}) + \log p(2|2,3)$

Décodage Nießen et al. [1998]: . . .

⟨anti-smoking, 2, 169, 2, the⟩ ⟨anti-tobacco, 2, 173, 2, the⟩ ⟨legisl., 2, 38, 2, anti-smo.⟩ ⟨the, 2, 31, 2, legislation⟩	⟨anti-smoking, 2, 153, 1, the⟩ ⟨anti-tobacco, 2, 157, 1, the⟩ ⟨legislation, 2, 18, 1, the⟩ ⟨the, 2, 49, 3, anti-smoking⟩	⟨anti-smoking, 2, 31, 1, the⟩ ⟨anti-tobacco, 2, 33, 1, the⟩ ⟨legislation, 2, 143, 1, the⟩ ⟨the, 2, 147, 1, the⟩
⟨anti-smoking, 1, 151, 0, bos⟩ ⟨anti-smoking, 2, 281, 0, bos⟩ ⟨anti-smoking, 3, 286, 0, bos⟩ ⟨anti-tobacco, 1, 151, 0, bos⟩ ⟨anti-tobacco, 2, 281, 0, bos⟩ ⟨anti-tobacco, 3, 287, 0, bos⟩ ⟨legislation, 1, 23, 0, bos⟩ ⟨legislation, 2, 34, 0, bos⟩ ⟨legislation, 3, 154, 0, bos⟩ ⟨the, 1, 21, 0, bos⟩ ⟨the, 2, 51, 0, bos⟩ ⟨the, 3, 127, 0, bos⟩	⟨anti-smoking, 1, 153, 0, bos⟩ ⟨anti-smoking, 2, 159, 0, bos⟩ ⟨anti-tobacco, 1, 154, 0, bos⟩ ⟨anti-tobacco, 2, 159, 0, bos⟩ ⟨legislation, 1, 17, 0, bos⟩ ⟨legislation, 2, 147, 0, bos⟩ ⟨the, 1, 18, 0, bos⟩ ⟨the, 2, 148, 0, bos⟩ ⟨bill, 1, 11, 0, bos⟩	⟨anti-smoking, 1, 29, 0, bos⟩ ⟨anti-tobacco, 30, 1, 0, bos⟩ ⟨legislation, 1, 143, 0, bos⟩ ⟨the, 1, 133, 0, bos⟩ ⟨no-smoking, 1, 26, 0, bos⟩
la	loi	anti-tabac

$$29 + \log p(\text{the}|\text{anti-smoking}, BOS) + \log p(\text{loi}|\text{the}) + \log p(2|2,3)$$
$$\log p(\text{legislation}|\text{anti-smoking}, BOS) + \log p(\text{la}|\text{legislation}) + \log p(2|1,3)$$

Décodage Nießen et al. [1998]: éventuellement . . .

. . .	. . .	. . .
⟨anti-smoking, 3, 287, 0, the⟩	⟨anti-smoking, 3, 163, 1, the⟩	⟨anti-smoking, 3, 52, 1, legis.⟩
⟨anti-tobacco, 3, 291, 0, 's⟩	⟨anti-tobacco, 3, 168, 1, the⟩	⟨anti-tobacco, 3, 46, 2, legis.⟩
⟨legis., 3, 40, 3, anti-smoking⟩	⟨legis., 3, 34, 3, anti-smoking⟩	⟨legislation, 3, 160, 2, legis.⟩
⟨the, 3, 51, 3, anti-smoking⟩	⟨the, 3, 48, 3, anti-smoking⟩	⟨the, 3, 153, 2, legis.⟩
⟨anti-smoking, 2, 169, 2, the⟩	⟨anti-smoking, 2, 153, 1, the⟩	⟨anti-smoking, 2, 31, 1, the⟩
⟨anti-tobacco, 2, 173, 2, the⟩	⟨anti-tobacco, 2, 157, 1, the⟩	⟨anti-tobacco, 2, 33, 1, the⟩
⟨legisl., 2, 38, 2, anti-smo.⟩	⟨legislation, 2, 18, 1, the⟩	⟨legislation, 2, 143, 1, the⟩
⟨the, 2, 31, 2, legislation⟩	⟨the, 2, 49, 3, anti-smoking⟩	⟨the, 2, 147, 1, the⟩
⟨anti-smoking, 1, 151, 0, bos⟩	⟨anti-smoking, 1, 153, 0, bos⟩	⟨anti-smoking, 1, 29, 0, bos⟩
⟨anti-smoking, 2, 281, 0, bos⟩	⟨anti-smoking, 2, 159, 0, bos⟩	⟨anti-tobacco, 30, 1, 0, bos⟩
⟨anti-smoking, 3, 286, 0, bos⟩	⟨anti-tobacco, 1, 154, 0, bos⟩	⟨legislation, 1, 143, 0, bos⟩
⟨anti-tobacco, 1, 151, 0, bos⟩	⟨anti-tobacco, 2, 159, 0, bos⟩	⟨the, 1, 133, 0, bos⟩
⟨anti-tobacco, 2, 281, 0, bos⟩	⟨legislation, 1, 17, 0, bos⟩	⟨no-smoking, 1, 26, 0, bos⟩
⟨anti-tobacco, 3, 287, 0, bos⟩	⟨legislation, 2, 147, 0, bos⟩	
⟨legislation, 1, 20, 0, bos⟩	⟨the, 1, 18, 0, bos⟩	
⟨legislation, 2, 24, 0, bos⟩	⟨the, 2, 148, 0, bos⟩	
⟨legislation, 3, 154, 0, bos⟩	⟨bill, 1, 11, 0, bos⟩	
⟨the, 1, 21, 0, bos⟩		
⟨the, 2, 51, 0, bos⟩		
la	loi	anti-tabac

$31 + \log p(\text{lesgislation}|\text{anti-smoking, the}) + \log p(\text{loi}|\text{legislation}) + \log p(3|2,3)$

Nießen et al. [1998]: Maximise sur la longueur cible

. . .	. . .	. . .
⟨anti-smoking, 3, 287, 0, the⟩	⟨anti-smoking, 3, 163, 1, the⟩	⟨anti-smoking, 3, 52, 1, legis.⟩
⟨anti-tobacco, 3, 291, 0, 's⟩	⟨anti-tobacco, 3, 168, 1, the⟩	⟨anti-tobacco, 3, 46, 2, legis.⟩
⟨legis., 3, 40, 3, anti-smoking⟩	⟨legis., 3, 34, 3, anti-smoking⟩	⟨legislation, 3, 160, 2, legis.⟩
⟨the, 3, 51, 3, anti-smoking⟩	⟨the, 3, 48, 3, anti-smoking⟩	⟨the, 3, 153, 2, legis.⟩
⟨anti-smoking, 2, 169, 2, the⟩	⟨anti-smoking, 2, 153, 1, the⟩	⟨anti-smoking, 2, 31, 1, the⟩
⟨anti-tobacco, 2, 173, 2, the⟩	⟨anti-tobacco, 2, 157, 1, the⟩	⟨anti-tobacco, 2, 33, 1, the⟩
⟨legislation, 2, 38, 2, the⟩	⟨legislation, 2, 18, 1, the⟩	⟨legislation, 2, 143, 1, the⟩
⟨the, 2, 31, 2, legislation⟩	⟨the, 2, 49, 3, anti-smoking⟩	⟨the, 2, 147, 1, the⟩
⟨anti-smoking, 1, 151, 0, bos⟩	⟨anti-smoking, 1, 153, 0, bos⟩	⟨anti-smoking, 1, 29, 0, bos⟩
⟨anti-smoking, 2, 281, 0, bos⟩	⟨anti-smoking, 2, 159, 0, bos⟩	⟨anti-tobacco, 30, 1, 0, bos⟩
⟨anti-smoking, 3, 286, 0, bos⟩	⟨anti-tobacco, 1, 154, 0, bos⟩	⟨legislation, 1, 143, 0, bos⟩
⟨anti-tobacco, 1, 151, 0, bos⟩	⟨anti-tobacco, 2, 159, 0, bos⟩	⟨the, 1, 133, 0, bos⟩
⟨anti-tobacco, 2, 281, 0, bos⟩	⟨legislation, 1, 17, 0, bos⟩	⟨no-smoking, 1, 26, 0, bos⟩
⟨anti-tobacco, 3, 287, 0, bos⟩	⟨legislation, 2, 147, 0, bos⟩	
⟨legislation, 1, 20, 0, bos⟩	⟨the, 1, 18, 0, bos⟩	
⟨legislation, 2, 24, 0, bos⟩	⟨the, 2, 148, 0, bos⟩	
⟨legislation, 3, 154, 0, bos⟩	⟨bill, 1, 11, 0, bos⟩	
⟨the, 1, 21, 0, bos⟩		
⟨the, 2, 51, 0, bos⟩		
⟨the, 3, 127, 0, bos⟩		
la	loi	anti-tabac

Nießen et al. [1998]:

length 1	it
length 2	the anti-smoking
length 3	the anti-smoking legislation
length 4	the anti-smoking legislation .
length 5	the anti-smoking legislation in the

DP et beam search

recherche en faisceau: beam search = DP + filtrage:

beam pruning: garder pour tout nœud n du treillis de recherche toutes hypothèses H telles que $\frac{p(H)}{p(H_{max})} \geq \beta$;

où β détermine la largeur du faisceau (beam) et H_{max} est l'hypothèse de probabilité maximale (trouvée)

$\hookrightarrow$ $\beta = 1$ on ne garde que les hypothèses de score maximal,

$\hookrightarrow$ $\beta = 0.5$ on garde toutes les hypothèses dont la probabilité est au moins la moitié de p(H_{max}). etc.

histogram pruning on garde les n-meilleures hypothèses de tout nœud

Pro et cons de la recherche DP

- Récursion spécifique à un jeu de modèles
- Permet assez facilement de générer des n-bests (on garde un retour arrière sur toutes les hypothèses qui amènent à un maximum). Voir à ce sujet Ueffing et al. [2002].

  idée populaire en RAP: générer un n-best à l'aide d'un modèle simple (rapide à calculer), puis renoter les n-meilleures hypothèses à l'aide d'un modèle plus complexe (c'est ce qu'on appelle le rescoring)
- Voir Garciá and Casacuberta [2001], Ueffing et al. [2002] pour des implémentations DP des modèles IBM3 et/ou 4.

Extrait d'un n-best: house of common debates

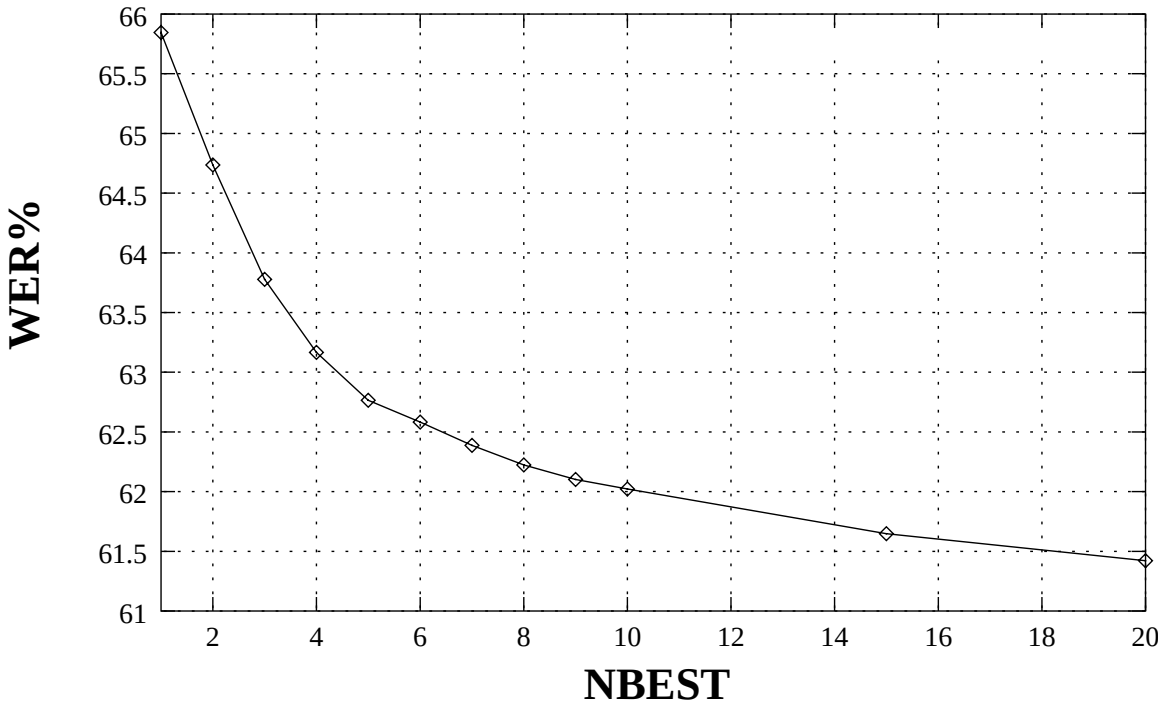
$\log p(f\|e)$	$\log lg(f)$	traduction
-22.6132	-2.33009	chambre des communes débats
-24.6132	-2.33009	travaux de la chambre
-27.4214	-2.694	travaux de la chambre .
-27.5228	-2.694	travaux de la chambre des
-27.6009	-3.10813	travaux de la chambre des communes
-28.9184	-2.33009	travaux de la séance
-29.5336	-4.0121	travaux de la chambre des communes .
-29.6181	-2.33009	travaux de la motion
-29.737	-2.694	travaux de la chambre de
-30.4074	-2.694	les travaux de la chambre
-31.3376	-2.33009	travaux de la chambre
-31.9783	-2.32975	travaux de la
. . .	. . .	. . .

Exemples de 5-best hansardiens

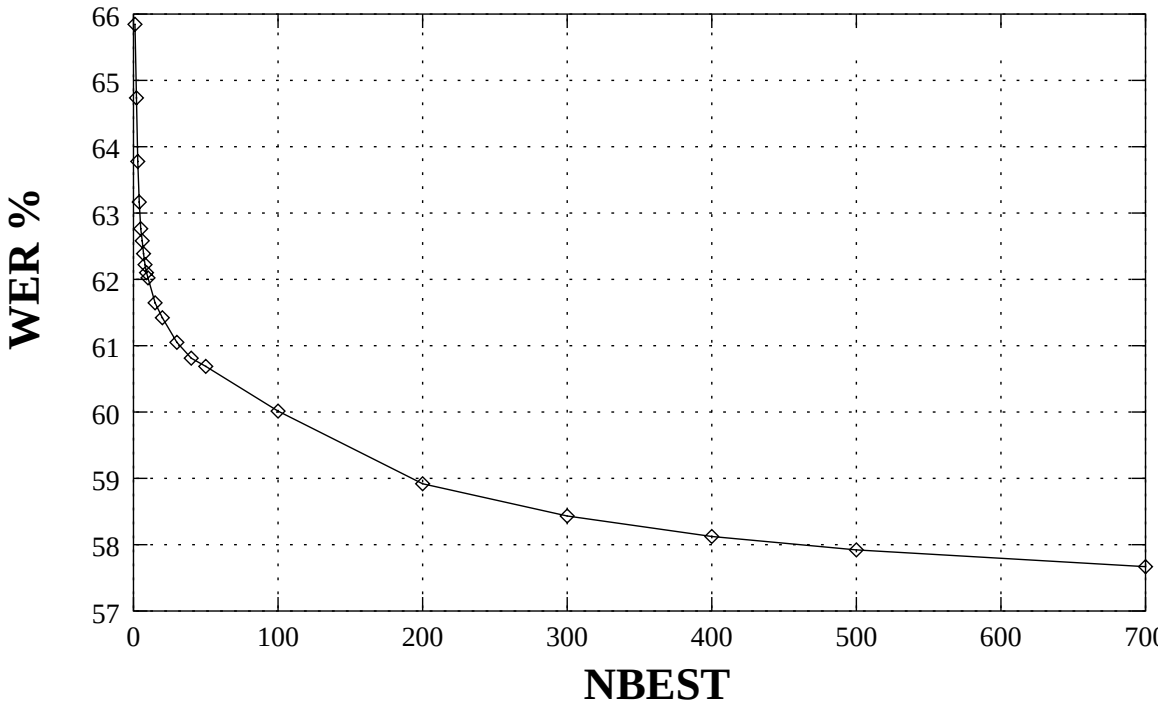
le canada est un acteur actif et dynamique sur la scène de l' économie mondiale .	
-171.58	the country is an active and dynamic on the scene of the world economy .
-177.954	the country is an active and dynamic on the scene of the world economy . .
-180.073	the country is an active and dynamic on the scene of the world economy of canada
-181.302	the country is an active and dynamic and internationally on the economy world .
-182.17	the country is an active and dynamic on the scene of the world 's economy

il y a même eu des peines d' incarcération .	
-106.895	there has been the same penalties and fines and imprisonment .
-108.596	there has been the same penalties as an incarceration .
-110.014	there has been the same penalties for incarceration .
-112.351	there has been the same penalties incarceration .
-115.546	there has been in the same penalties and fines and imprisonment .

Impact d'un n-best: WER vs. N-best



Impact d'un n-best: WER vs. N-best



Algorithme vorace (greedy)

Idée: on part d'une hypothèse (simple à obtenir) que l'on essaye d'améliorer à l'aide d'opérations élémentaires Marcu and Germann [2000], Germann et al. [2001]. Ex. fourni par les auteurs:

NULL_0 well_1 heard_2 $,_3$ it_4 talking_5 a_6 beautiful_7 victory_8 $._9$ $\text{bien}_{(1)}$ $\text{entendu}_{(2)}$ $,_{(3)}$ $\text{il}_{(4)}$ $\text{parle}_{(5)}$ $\text{de}_{(0)}$ $\text{une}_{(6)}$ $\text{belle}_{(7)}$ $\text{victoire}_{(8)}$ $._{(9)}$
NULL_0 well_1 heard_2 $,_3$ it_4 talks_5 a_6 great_7 victory_8 $._9$ $\text{bien}_{(1)}$ $\text{entendu}_{(2)}$ $,_{(3)}$ $\text{il}_{(4)}$ $\text{parle}_{(5)}$ $\text{de}_{(0)}$ $\text{une}_{(6)}$ $\text{belle}_{(7)}$ $\text{victoire}_{(8)}$ $._{(9)}$
NULL_0 well_1 understood_2 $,_3$ it_4 talks_5 about_6 a_7 great_8 victory_9 $._{10}$ $\text{bien}_{(1)}$ $\text{entendu}_{(2)}$ $,_{(3)}$ $\text{il}_{(4)}$ $\text{parle}_{(5)}$ $\text{de}_{(6)}$ $\text{une}_{(7)}$ $\text{belle}_{(8)}$ $\text{victoire}_{(9)}$ $._{(10)}$
NULL_0 well_1 understood_2 $,_3$ he_4 talks_5 about_6 a_7 great_8 victory_9 $._{10}$ $\text{bien}_{(1)}$ $\text{entendu}_{(2)}$ $,_{(3)}$ $\text{il}_{(4)}$ $\text{parle}_{(5)}$ $\text{de}_{(6)}$ $\text{une}_{(7)}$ $\text{belle}_{(8)}$ $\text{victoire}_{(9)}$ $._{(10)}$
NULL_0 quite_1 naturally_2 $,_3$ he_4 talks_5 about_6 a_7 great_8 victory_9 $._{10}$ $\text{bien}_{(1)}$ $\text{entendu}_{(2)}$ $,_{(3)}$ $\text{il}_{(4)}$ $\text{parle}_{(5)}$ $\text{de}_{(6)}$ $\text{une}_{(7)}$ $\text{belle}_{(8)}$ $\text{victoire}_{(9)}$ $._{(10)}$

Algorithme greedy

Avantage (table prise dans Germann et al. [2001])

Décodage avec un modèle IBM4 et un trigramme:

	sec./sent		erreurs	
$\|sent\|$	stack	greedy	stack	greedy
6	13.72	1.58	42	46
8	45.45	2.75	59	68
10	105.15	3.83	57	63
15	>2000	12.06	74	75
20	—	49.23	86	93
505 phrases (101 de chaque longueur)				

Algorithme A^* (d'après Jelinek [1998])

- Nous cherchons le chemin w_1^n qui maximise un critère, cad une fonction g définie pour tout chemin et toute longueur $k = 1, 2, \ldots, n$.

- Soit une fonction $F(w_1^k) = g(w_1^k) + d(w_1^k)$ où d est une fonction telle que:

$$d(w_1^n) = 0$$
$$d(w_1^k) \geq \underbrace{\underbrace{g(w_1^k||z_{k+1}^n)}_{\text{score total}} - \underbrace{g(w_1^k)}_{\text{score partiel}}}_{\text{mesure optimiste d'une complétion de } k+1 \text{ à } n} \qquad \forall z_{k+1}^n$$

où $a||b$ désigne la séquence formée par la concaténation des deux séquences a et b.

Algorithme A^*

De ce qui précède, si $F(w_1^n) \geq F(\tilde{w}_1^k)$, alors $g(w_1^n) \geq g(\tilde{w}_1^k||z_{k+1}^n)$ pour toute séquence z_{k+1}^n.

En clair, si $F(w_1^n) \geq F(\tilde{w}_1^k)$, il est inutile de considérer les complétions de $\tilde{w}_1^k$ car elle mèneront à des chemins moins bien notés par g.

C'est l'idée exploitée dans l'algorithme A^* (stack-decoder) qui repose sur une structure de donnée de type *file de priorité* $\mathcal{S}$:

1. Mettre dans $\mathcal{S}$ les hypothèses singletons w_i (constituées d'un seul mot) avec leur score associé $F(w_i)$.
2. Prendre l'entrée au sommet de $\mathcal{S}$ (c'est l'hypothèse actuelle la plus prometteuse selon F). Soit w_1^k cette hypothèse.
   Si $k = n$ alors nous avons la meilleure solution (selon g) et son score est $g(w_1^n)$
   Sinon, placer dans $\mathcal{S}$ toutes les extensions $w_1^k||v$ d'un mot v.
3. Aller en 2.

Algorithme A^*

Note: *file de priorité* = "pile" qui contient en son sommet l'élément de plus haut score.

- Dans l'algorithme A^* que nous venons de voir, la pile (ou file de priorité) contient *toutes* les hypothèses à développer. Les hypothèses les plus prometteuses sont développées en premier.
- **Pré-requis:** définir une fonction auxiliaire "optimiste" d , cad une fonction qui surestime toujours le score d'une complétion. On parle habituellement de fonction heuristique.

  $\hookrightarrow$ Le choix de cette fonction heuristique a un impact important sur les performances de l'algorithme (pas "optimiste" $\longrightarrow$ la solution n'est pas nécessairement optimale; trop "optimiste", l'algorithme est ralenti.)
- **Note:** nécessite que la fonction que l'on souhaite maximiser soit cumulative

A^* et traduction: Wang and Waibel [1997] pour IBM 2

- $f = f_1^J$ donné, on cherche:

$$\hat{e} = \hat{e}_1^{\hat{I}} = \underset{I, e_1^I}{\operatorname{argmax}}\, p(e_1^I)\epsilon \prod_{j=1}^{J} \sum_{i=0}^{I} t(f_j|e_i)a(i|j,l,m)$$

- **Note:** dans la suite nous appelons e la phrase source (c'est la phrase qui rentre dans le canal) et f la phrase cible (c'est la phrase qui en ressort).
- Une hypothèse H est notée: $I : e_1 e_2 \ldots e_k$ et signifie que la traduction aura I mots et que l'on connait déjà les k premiers.
- À chaque hypothèse H est associée un score f_H constitué de deux parties: le score (exact) g_H du préfixe $e_1 e_2 \ldots e_k$ et le score heuristique (optimiste) h_H du suffixe $e_{k+1} \ldots e_I$ à déterminer.

A^* et traduction: Wang and Waibel [1997]

Score du préfixe

- Soit $S_H(j)$ la contribution des k premiers mots sources à la masse de probabilité du mot cible f_j:

$$S_H(j) = \epsilon \sum_{i=0}^{k} t(f_j|e_i)a(i|j,I,J)$$

  $\hookrightarrow$ Étendre H d'un mot augmentera donc $S_H(j)$, pour tout $j \in [1,J]$.

- Pour l'ensemble des mots cibles, la contribution du modèle de traduction du préfixe est donc $\prod_{j=1}^{J} S_H(j)$, soit en logarithme: $\sum_{j=1}^{J} \log S_H(j)$.
- S'ajoute la contribution du modèle de langue (n-gram d'ordre N):

$$g_H = \underbrace{\sum_{j=1}^{J} \log S_H(j)}_{\text{traduction}} + \underbrace{\sum_{i=1}^{k} \log p(e_i|e_{i-N+1}^{i-1})}_{\text{langue}}$$

A^* et traduction: Wang and Waibel [1997]

Score du préfixe

On peut calculer g_H à partir du score d'une hypothèse parente: $P = I$: $e_1 e_2 \dots e_{k-1}$:

$$
\begin{aligned}
g_H &= \sum_{j=1}^{J} \log \left[\epsilon \sum_{i=0}^{k-1} t(f_j|e_i) a(i|j,I,L) + \epsilon t(f_j|e_k) a(k|j,I,J) \right] + \\
&\quad \sum_{i=1}^{k-1} \log p(e_i|e_{i-N+1}^{i-1}) + \log p(e_k|e_{k-N+1}^{k-1}) \\
&= \sum_{j=1}^{J} \log \left[S_P(j) + \epsilon t(f_j|e_k) a(k|j,I,J) \right] + \\
&\quad \sum_{i=1}^{k-1} \log p(e_i|e_{i-N+1}^{i-1}) + \log p(e_k|e_{k-N+1}^{k-1}) \\
&= \sum_{j=1}^{J} \log \left[S_P(j) \left(1 + \frac{\epsilon t(f_j|e_k) a(k|j,I,J)}{S_P(j)} \right) \right] + \\
&\quad \sum_{i=1}^{k-1} \log p(e_i|e_{i-N+1}^{i-1}) + \log p(e_k|e_{k-N+1}^{k-1}) \\
&= g_P + \log p(e_k|e_{k-N+1}^{k-1}) + \sum_{j=1}^{J} \log \left[1 + \frac{\epsilon t(f_j|e_k) a(k|j,I,J)}{S_P(j)} \right]
\end{aligned}
$$

A^* et traduction: Wang and Waibel [1997]

Score du suffixe: Contribution du modèle de langue:

- soit PP_{train} est la perplexité du modèle de langue obtenue sur le **corpus d'entraînement**:

$$h_H^{LM} = -(I-k)PP_{train} + C$$

$\hookrightarrow$ l'idée derrière ce score est que la perplexité sur-estime la vraisemblance d'un nouveau segment (en moyenne).

- C'est d'autant plus vrai que I est grand devant k. Lorsqu'il reste peu de mots à étendre C prend de l'importance et sert à ne pas sous-estimer la complétion:

$$C = PP_{train} + \log(P_{max})$$

où P_{max} est la probabilité maximale attribuée à un n-gramme dans le modèle.

A^* et traduction: Wang and Waibel [1997]

Score du suffixe: Contribution du modèle de traduction:

- Soit $v_{iI}(j)$ la contribution maximale d'un mot source à la masse de probabilité associée à f_j, le mot source étant entre les positions i et I incluses (L_E est le lexique source):

$$v_{iI}(j) = \max_{k \in [i,I], e_k \in L_E} t(f_j|e_k) a(k|j,I,M)$$

- La contribution du modèle de traduction est alors:

$$h_H^{TM} = \sum_{j=1}^{J} max\{0, \log v_{(k+1)I}(j) - \log S_H(j)\}$$

$\hookrightarrow$ c'est l'amélioration maximale qu'un mot source ajouté peut apporter à la vraisemblance d'un mot f_j.

Le score heuristique final est donné par: $h_H = h_H^{LM} + h_H^{TM}$ (pour $k < I$, 0 sinon)

A^* et traduction: détails pratiques

mémoire : en général, on ne peut pas maintenir dans une pile toutes les hypothèses à développer. On se retreint aux meilleures d'entre-elles.

plus de garantie d'optimalité

problème intrinsèque à A* : saturation de la pile avec des hypothèses proches

on maintient plusieurs piles (multi-stack search)

Lire Och et al. [2001], Garciá and Casacuberta [2001] pour des variantes DP *beam search* applicables aux modèles IBM3 et IBM4.

Quelques exemples pouvant aider à apprécier la situation

SRCE	j' ai parlé plus tôt du déséquilibre financier .
IBM4	i have talked about more than ever before for the fiscal imbalance .
IBM2	i mentioned earlier , the fiscal imbalance .
SRCE	nous devons revenir sur la question de la péréquation au canada .
IBM4	we must go back to the question of the equalization in canada .
IBM2	we must address the issue of the equalization of canada .
SRCE	la péréquation constitue un programme social très important .
IBM4	the equalization is a social program is very important .
IBM2	on equalization program is extremely important social
SRCE	ce n' est pas le comité qui n' en a pas tenu compte .
IBM4	this is the committee that has to be held to account .
IBM2	this is not the committee , which has been ignored .
SRCE	nous avons mis fin à une période de 28 ans de déficit .
IBM4	we have to put an end to a period of 28 years of deficit .
IBM2	we have developed an end of seven years $ 28 billion .

Références

ACL-35. *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL)*, Madrid, Spain, July 1997.

Ismael Garciá and Fransisco Casacuberta. Search algorithms for statistical machine translation based on dynamic programming and pruning techniques. In *Proceedings of the 8th Machine Translation Summit*, pages 115–120, Santiago de Compostela, Galicia, Spain, September 2001.

Ulrich Germann, Micheal Jahr, Kevin Knight, Daniel Marcu, and Kenji Yamada. Fast decoding and optimal decoding for machine translation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 228–235, Toulouse, France, July 2001.

Frederick Jelinek. *Statistical Methods for Speech Recognition*. The MIT Press, Cambridge, Massachusetts, 1998.

Kevin Knight. Decoding complexity in word-replacement translation models. *Computational*

Linguistics, 25(4), 1999.

Daniel Marcu and Ulrich Germann. The isi rewrite decoder release 0.7.0b. http://www.isi.edu/ germann/software/ReWrite-Decoder/index.html, 2000.

S. Nießen, S. Vogel, H. Ney, and C. Tillmann. A dp based search algorithm for statistical machine translation. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics (ACL) and 17th International Conference on Computational Linguistics (COLING) 1998*, pages 960–966, Montréal, Canada, August 1998.

Franz Josef Och, Nicola Ueffing, and Hermann Ney. An efficient a* search algorithm for statistical machine translation. In *Proceedings of the Workshop on Data Driven Machine Translation yielded at the 39th Annual Meeting of the ACL*, pages 55–62, Toulouse, France, July 2001.

C. Tillmann, S. Vogel, H. Ney, and A. Zubiaga. A dp-based search using monotone alignments in statistical translation. In ACL-35 ACL-35, pages 289–296.

Nicola Ueffing, Franz Joseph Och, and Hermann Ney. Generation of word graphs in statistical machine translation. In *Proceedings of the 7th Conference on Empirical Methods*

in Natural Language Processing (EMNLP), pages 156–163, Philadelphia, PA, USA, 2002.

Ye-Yi Wang and Alex Waibel. Decoding algorithm in statistical machine translation. In ACL-35 ACL-35, pages 366–372.