Introduction aux modèles de langue

Philippe Langlais
felipe@iro.umontreal.ca

October 12, 2012

Plan

Quelques sources à l'origine de ces transparents

- "An Empirical Study of Smoothing Techniques for Language Modeling", (Chen and Goodman 1996)
- "A Gentle Tutorial on Information Theory and Learning", Roni Rosenfeld
- "A bit of progress in Language Modeling", Extended Version, (Goodman 2001)
- "Two decades of statistical language modeling: where do we go from here ?", (Rosenfeld 2000)

Modèles de langue

Définition

- Un modèle de langue probabiliste est un modèle qui spécifie une distribution $p(s)$ sur les chaînes s de la langue modélisée:

$$\sum_{s} Pr(s) = 1$$

- Sans perte d'information, si l'on considère que s est une séquence de N mots (phrase ?), $s \equiv w_1 \dots w_N$, alors:

$$Pr(s) \stackrel{def}{=} \prod_{i=1}^{N} Pr(w_i | \underbrace{w_1 \dots w_{i-1}}_{h})$$

où h est appelé l'historique

Exemples d'applications

Classification de documents

But: classer un texte selon plusieurs catégories (ex: sport, religion, etc.).

- $\mathscr{C}$ l'ensemble des classes possibles,
- c_i, $i \in [1, |\mathscr{C}|]$ l'une de ces classes,
- T un texte dont on veut connaître la classe c_T:

$$
\begin{aligned}
c_T &= \underset{i \in [1,|\mathscr{C}|]}{\operatorname{argmax}} p(c_i|T) \\
&= \underset{i \in [1,|\mathscr{C}|]}{\operatorname{argmax}} \frac{p(T|c_i) \times p(c_i)}{p(T)} \\
&= \underset{i \in [1,|\mathscr{C}|]}{\operatorname{argmax}} \underbrace{p(T|c_i)}_{\text{langue}} \times \underbrace{p(c_i)}_{\text{a priori}}
\end{aligned}
$$

- en pratique, un modèle unigramme donne des performances (étonnamment) raisonnables.

Exemples d'applications

Recherche d'information

But: trouver les documents D pertinents à une requête R

$$\begin{aligned}\hat{D} &= \operatorname*{argmax}_D p(D|R) \\ &= \operatorname*{argmax}_D \underbrace{p(D)}_{\text{a priori}} \times \underbrace{P(R|D)}_{\text{langue}}\end{aligned}$$

- Un modèle de langue par document dans la collection !
- Mode opératoire de base:

$$\operatorname*{argmax}_D \prod_{i=1}^{|R|} \lambda p(R_i|D) + (1-\lambda) p(R_i|Collection)$$

Exemples d'applications

Identification de la langue

- soit T un texte, T_i le i-ème caractère de T et T_a^b la séquence $T_a, T_{a+1}, \ldots, T_b$
- soit $\mathscr{L}$ l'ensemble des langues, l_i une de ces langues

But: Découvrir L_T, la langue de T

$$
\begin{aligned}
L_T &= \operatorname*{argmax}_{i \in [1,|\mathscr{L}|]} p(L_i|T) \\
&\approx \operatorname*{argmax}_{i \in [1,|\mathscr{L}|]} \underbrace{\prod_{c=1}^{|T|} p(T_c|T_{c-n+1}^{c-1}, L_i)}_{\text{n-car}} \times \underbrace{p(L_i)}_{\text{a priori}}
\end{aligned}
$$

Identification de la langue

Jag talar en litten svenska

suédois	**cp1252**	**0.076**
suédois	cp850	0.076
suédois	macintosh	0.076
norvégien	cp1252	0.056
norvégien	cp850	0.056
norvégien	macintosh	0.056
danois	cp1252	0.039
danois	cp850	0.039
danois	macintosh	0.039
néerlandais	cp1252	0.034
néerlandais	cp850	0.034
néerlandais	macintosh	0.034
allemand	cp1252	0.023
allemand	cp850	0.023
allemand	macintosh	0.023

estonien	iso-8859-4	0.016
hongrois	cp1250	0.015
hongrois	cp852	0.015
anglais	cp1252	0.014
français	cp1252	0.011
français	cp850	0.011
français	macintosh	0.011
espagnol	cp1252	0.010
espagnol	cp850	0.010
espagnol	macintosh	0.010
albanais	cp1252	0.010
albanais	cp850	0.010
albanais	macintosh	0.010
finnois	cp1252	0.010
finnois	cp850	0.010

http://www-rali.iro.umontreal.ca/SILC/

Identification de la langue

Mon char est parké au garage		
français	cp1252	0.099
allemand	cp1252	0.064
français	cp850	0.036
français	macintosh	0.036

Je parke mon char		
néerlandais	cp1252	0.046
néerlandais	cp850	0.046
néerlandais	macintosh	0.046
anglais	cp1252	0.046
allemand	cp1252	0.038

Exemples d'applications

Réaccentuation automatique de textes

Le systeme m'accentue	$\rightarrow$	Le système m'accentue.
Le systeme m'a accentue	$\rightarrow$	Le système m'a accentué.

- Soit w_1^n la phrase de n mots à réaccentuer
- Implantation possible:
  - désaccentuer tous les mots w_i
  - sélectionner pour tout mot w_i ses versions accentuées possibles, soit a_i cet ensemble
    (lorqu'un mot n'est pas accentuable, $a_i = \{w_i\}$)
  - considérer toutes les phrases que l'on peut construire à partir des a_i (en prenant un mot par a_i) et sélectionner celle de plus forte probabilité selon le modèle de langue
  - www.iro-rali.umontreal.ca

Exemples d'applications

Traduction automatique

- Soit S un document dans la langue source

But Trouver $\hat{T}$ la traduction de S

- approche à la traduction proposée au début des années 90 par une équipe d'IBM (voir les acétates sur la traduction):

$$\begin{aligned} \hat{T} &= \operatorname{argmax}_T P(T|S) \\ &= \operatorname{argmax}_T \underbrace{P(S|T)}_{\text{traduc.}} \underbrace{\times P(T)}_{\text{langue}} \end{aligned}$$

- 2 distributions que l'on sait estimer
- problème de recherche de maximum (argmax) non trivial

Modèle *n*-gramme

Décomposition par la règle de chaînage

$$
\begin{aligned}
Pr(\text{John aime Marie qui aime Paul}) = \\
Pr(\text{John} \mid \text{BOS}) \times \\
Pr(\text{aime} \mid \text{BOS John}) \times \\
Pr(\text{Marie} \mid \text{BOS John aime}) \times \\
Pr(\text{qui} \mid \text{BOS John aime Marie}) \times \\
Pr(\text{aime} \mid \text{BOS John aime Marie qui}) \times \\
Pr(\text{Paul} \mid \text{BOS John aime Marie qui aime})
\end{aligned}
$$

- approximation markovienne d'ordre $n-1$, le modèle *n*-gramme:

$$p(s = w_1^n) \approx \prod_{i=1}^{N} p(w_i | w_{i-n+1}^{i-1})$$

Modèle n-gramme

Cas du modèle trigramme

$$p(s) = \prod_{i=1}^{N} p(w_i | w_{i-2} w_{i-1})$$

$$\begin{aligned} Pr(\text{John aime Marie qui aime Paul}) = \\ Pr(\text{John} \mid \text{BOS BOS}) \times \\ Pr(\text{aime} \mid \text{BOS John}) \times \\ Pr(\text{Marie} \mid \text{John aime}) \times \\ Pr(\text{qui} \mid \text{aime Marie}) \times \\ Pr(\text{aime} \mid \text{Marie qui}) \times \\ Pr(\text{Paul} \mid \text{qui aime}) \end{aligned}$$

Estimateur à maximum de vraisemblance

Intuition

Soit $\mathscr{D} \equiv w_1 \dots w_N$, un corpus (texte) de N mots

- Cas de l'unigramme $p(s) = \prod_i p(w_i)$:

$$p(w) = \frac{|w|}{N}, \text{ avec } |w| \text{ la fréquence de } w \text{ dans } \mathscr{D}$$

- Cas du bigramme $p(s) = \prod_i p(w_i|w_{i-1})$:

$$p(w_i|w_{i-1}) = \frac{|w_{i-1}w_i|}{|w_{i-1}|} = \frac{|w_{i-1}w_i|}{\sum_w |w_{i-1}w|}$$

- Cas du n-gramme:

$$p(w_i|w_{i-n+1}^{i-1}) = \frac{|w_{i-n+1}^{i-1}w_i|}{\sum_w |w_{i-n+1}^{i-1}w|}$$

Estimateur à maximum de vraisemblance

Vincent aime Virginie
Estelle aime les fleurs
Elle aime les fleurs jaunes plus particulièrement

- $p(\text{Vincent aime les fleurs}) = \frac{1}{3} \times 1 \times \frac{2}{3} \times 1 \times \frac{1}{2} = \frac{1}{9} \approx 0.111$

$$
\begin{array}{rll}
 & p(\text{Vincent}|\text{BOS}) & |\text{BOS Vincent}|/|\text{BOS}| = 1/3 \\
\times & p(\text{aime}|\text{Vincent}) & |\text{Vincent aime}|/\text{Vincent} = 1/1 = 1 \\
\times & p(\text{les}|\text{aime}) & |\text{aime les }|/|\text{aime}| = 2/3 \\
\times & p(\text{fleurs}|\text{les}) & |\text{les fleurs}|/|\text{les}| = 2/2 = 1 \\
\times & p(\text{EOS}|\text{fleurs}) & |\text{fleurs EOS}|/|\text{fleurs}| = 1/2
\end{array}
$$

- $p(\text{Virginie aime les fleurs}) = 0$ car $|\text{BOS Virginie}| = 0$

Estimateur à maximum de vraisemblance

Augmenter le corpus d'entraînement (1/2)

- Le corpus Austen[1] contient 8762 phrases, 620968 tokens et 14274 types.
- En théorie, il existe:

$$
\begin{aligned}
n_{bigram} &= 14274 \times 14274 = 203,747,076 \\
n_{trigram} &\approx 2.9 \times 10^{12} \\
n_{4-gram} &\approx 4.1 \times 10^{16} \\
n_{5-gram} &\approx 5.9 \times 10^{20}
\end{aligned}
$$

- On observe:
  - 194 211 bigrammes $\neq$ (soit $\approx$ 0.09%), dont 69% d'hapax legomena
  - 462 615 trigrammes $\neq$ (soit $\approx 10^{-5}$%), dont 87% d'hapax

[1] disponible sur la page web de (Manning and Schütze 1999)

Estimateur à maximum de vraisemblance

Augmenter le corpus d'entraînement (2/2)

- Une partie du corpus Hansard[2] contient 1,639,250 phrases, 33,465,362 tokens et 103,830 types.
- En théorie, il existe:

$$
\begin{aligned}
n_{bigram} &= 10,780,668,900 \text{ (dix milliards !)} \\
n_{trigram} &\approx 1.1 \times 10^{15} \\
n_{4-gram} &\approx 1.2 \times 10^{20} \\
n_{5-gram} &\approx 1.2 \times 10^{25}
\end{aligned}
$$

- Lissage des probabilités, *i.e* donner une probabilité à des choses non vues à l'entraînement.

[2] http://www.parl.gc.ca/HouseChamberBusiness/ChamberSittings.aspx?Language=F

Estimation par maximum de vraisemblance

Soit $\mathscr{D} \equiv (x_1, \ldots, x_n) \sim p(x|\theta)$ un corpus d'échantillons tirés indépendamment et aléatoirement d'une distribution $p(x|\theta)$ dont les paramètres θ sont inconnus. On souhaite estimer θ.

- les observations étant indépendantes, on a:

$$p(\mathscr{D}|\theta) = \prod_{i=1}^{n} p(x_i|\theta)$$

- $p(\mathscr{D}|\theta)$ vue comme une fonction de θ est appelée la vraisemblance (likelihood) de θ respectivement aux données.
- on suppose θ fixe, on recherche $\hat{\theta}$:

$$\hat{\theta} = \hat{\theta}_{MLE} = \underset{\theta}{\operatorname{argmax}}\, p(\mathscr{D}|\theta)$$

MLE = Maximum Likelihood Estimation

Estimation à maximum de vraisemblance

- Il est souvent plus simple de maximiser la log-vraisemblance $l(\theta) = \ln p(\mathscr{D}|\theta)$:

$$\underset{\theta}{\operatorname{argmax}}\, l(\theta) = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{n} \ln p(x_i|\theta)$$

- Soit $\theta \equiv (\theta_1, \ldots, \theta_m)^t$ le vecteur de paramètres et $\nabla_\theta \equiv (\frac{\partial}{\partial\theta_1}, \ldots, \frac{\partial}{\partial\theta_m})^t$ l'opérateur gradient

- $\nabla_\theta l(\theta) = (\underbrace{0, \ldots, 0}_{m})^t$

  représente un ensemble de m conditions nécessaires que doit vérifier l'estimateur de vraisemblance.

  La solution est soit un extremum global (plutôt rare), soit un extremum local (le plus souvent), soit (plus rarement) un point d'inflexion.

MLE et bernouilli

Soit un échantillon $x = (x_1, \ldots, x_n)$ extrait de n épreuves de bernouilli de paramètre θ. On observe s événements positifs. Quelle est la valeur de θ ?

- On résoud $\frac{\partial}{\partial \theta} \ln p_\theta(x) = 0$

$$\frac{\partial}{\partial \theta} \ln \left[\binom{n}{s} \theta^s (1-\theta)^{n-s} \right] = \frac{\partial}{\partial \theta} \ln \theta^s + \frac{\partial}{\partial \theta} \ln (1-\theta)^{n-s}$$

- soit $\frac{s}{\theta} - \frac{n-s}{1-\theta} = 0$ qui a pour solution intuitive: $\theta_{MLE} = \frac{s}{n}$ appelée aussi fréquence relative

Si on fait 10 jetés d'une pièce et que 3 "piles" sont observés alors, la probabilité que la pièce tombe sur pile est estimée à 3/10.

Quel rapport avec les modèles de langue ?

Hypothèse: les mots d'un texte $\mathscr{D}$ sont générés aléatoirement de manière indépendante par une multinomiale (jetés d'un dé à $|V|$ faces, où V est le vocabulaire)

- La vraisemblance est alors:

$$p(\mathscr{D}|\theta) = p(x_1, \ldots, x_{|V|}) = \underbrace{\frac{(\sum_{i=1}^{|V|} x_i)!}{\prod_{i=1}^{|V|} x_i!}}_{\beta} \prod_{i=1}^{|V|} p_i^{x_i} \quad \text{avec} \sum_{i=1}^{|V|} p_i = 1$$

où x_i est le compte (observé) de chacun des mots w_i de V dans D; et p_i sont les paramètres de la distribution que l'on souhaite apprendre.

- On résoud $|V|$ équations (sous la contrainte $\sum_i p_i = 1$):

$$\frac{\partial}{\partial p_i} \log \left(\beta \prod_i p_i^{x_i} \right) = 0$$

Quel rapport avec les modèles de langue ?

- En introduisant λ, un coefficient de Lagrange:

$$\frac{\partial}{\partial p_i}\left[\log p(D|\theta) + \lambda(1 - \sum_{i=1}^{|V|} p_i)\right] = 0$$

- on a:

$$\frac{\partial}{\partial p_i}\left[\log \beta + \sum_i x_i \log p_i\right] - \lambda = 0$$

- soit:

$$p_i = \frac{x_i}{\lambda}$$

- et comme $\sum_i p_i = 1$, alors $\lambda = \sum_i x_i$, d'où:

$$p_i = \frac{x_i}{\sum_i x_i} = \frac{x_i}{N}$$

Une méthode de lissage simple

Le add-one smoothing (Loi de Laplace, 1814)

- Idée (simple qui marche mal en pratique): prétendre que tout événement a été vu une fois de plus dans $\mathscr{D}$

- Soit $|V|$ la taille du vocabulaire:

$$p_{add-one}(w_i|w_{i-1}) = \frac{|w_{i-1}w_i| + 1}{\sum_w (|w_{i-1}w| + 1)} = \frac{|w_{i-1}w_i| + 1}{|V| + \sum_w |w_{i-1}w|}$$

En particulier, le bigramme `fleur bleue` n'a pas été rencontré dans $\mathscr{D}$, nous prétendons cependant que nous l'avons trouvé une fois.

Le add-one smoothing, Vincent et Virginie

p(**Vincent aime les fleurs**)

Vincent aime Virginie
Estelle aime les fleurs
Elle aime les fleurs jaunes plus particulièrement

- avant: 0.111, maintenant: $1/3718 \approx 0.000269$

$$
\begin{aligned}
& p(\text{Vincent}|\text{BOS}) && \frac{|\text{BOS Vincent}|+1}{|\text{BOS}|+|V|} && = \frac{1+1}{3+10} \\
\times\ & p(\text{aime}|\text{Vincent}) && \frac{|\text{Vincent aime}|+1}{|\text{Vincent}|+|V|} && = \frac{1+1}{1+10} \\
\times\ & p(\text{les}|\text{aime}) && \frac{|\text{aime les}|+1}{|\text{aime}|+|V|} && = \frac{2+1}{3+10} \\
\times\ & p(\text{fleurs}|\text{les}) && \frac{|\text{les fleurs}|+1}{|\text{les}|+|V|} && = \frac{2+1}{2+10} \\
\times\ & p(\text{EOS}|\text{fleurs}) && \frac{|\text{fleurs EOS}|+1}{|\text{fleurs}|+|V|} && = \frac{1+1}{2+10}
\end{aligned}
$$

Le add-one smoothing, Vincent et Virginie

p(**Virginie aime les fleurs**)

Vincent aime Virginie
Estelle aime les fleurs
Elle aime les fleurs jaunes plus particulièrement

- avant: 0, maintenant $\approx 6.72\ 10^{-5}$:

$$
\begin{aligned}
& p(\text{Virginie}|\text{BOS}) && \frac{|\text{BOS Virginie}|+1}{|\text{BOS}|+|V|} && = \frac{0+1}{3+10}\ (\star) \\
\times\ & p(\text{aime}|\text{Virginie}) && \frac{|\text{Virginie aime}|+1}{|\text{Virginie}|+|V|} && = \frac{0+1}{1+10}\ (\star) \\
\times\ & p(\text{les}|\text{aime}) && \frac{|\text{aime les}|+1}{|\text{aime}|+|V|} && = \frac{2+1}{3+10} \\
\times\ & p(\text{fleurs}|\text{les}) && \frac{|\text{les fleurs}|+1}{|\text{les}|+|V|} && = \frac{2+1}{2+10} \\
\times\ & p(\text{EOS}|\text{fleurs}) && \frac{|\text{fleurs EOS}|+1}{|\text{fleurs}|+|V|} && = \frac{1+1}{2+10}
\end{aligned}
$$

$\star$ changée par rapport au calcul précédant

Problème avec le add-one smoothing

Le corpus Austen contient $N = 620,968$ tokens (14,274 types) et 194,211 bigrammes différents pour un nombre total théorique de bigrammes: $n_{th} = 203,747,076$. Le nombre de bigrammes non vus (à vocabulaire fermé) est donc $n_0 = 203,552,865$.

- Estimation (jointe) d'un bigramme:

$$p_{add-one}(w_{i-1}w_i) = \frac{|w_{i-1}w_i| + 1}{N + n_{th}}$$

- Probabilité d'un bigramme non rencontré dans $\mathscr{D}$:

$$p_0 = 1/(620,968 + 203,747,076) \approx 4.9 \times 10^{-9}$$

- Masse totale de probabilité associée à des bigrammes non vus: $n_0 \times p_0 \approx 0.996$

  99.6% de l'espace des probabilités a été distribué à des événements non vus !

Lissage *add-one* et estimateur maximum à postériori (MAP)

La formule de Bayes nous donne la relation suivante:

$$\overbrace{p(\theta|x)}^{\text{à postériori}} = \frac{\overbrace{p(x|\theta)}^{\text{vraisemblance}} \times \overbrace{p(\theta)}^{\text{à priori}}}{\underbrace{p(x)}_{\text{évidence}}}$$

- $p(\theta|x)$ est la probabilité à posteriori de θ une fois les données observées,
- $p(x|\theta)$ est la vraisemblance de θ au regard de x
- $p(\theta)$ est l'à priori

Lissage *add-one* et MAP

- Un (autre) estimateur populaire consiste à maximiser la distribution à postériori des paramètres (θ):

$$\theta_{map} = \underset{\theta}{\operatorname{argmax}}\, p(\theta|\mathscr{D}) = \underset{\theta}{\operatorname{argmax}} \log p(\mathscr{D}|\theta) + \log p(\theta)$$

On parle de maximum à postériori (estimateur MAP)

- θ est maintenant une variable aléatoire pour laquelle nous avons un à priori.
- $\mathscr{D}$ est la résultante de N tirages multinomiaux:

$$p(\mathscr{D}|\theta) = p(x_1, \ldots, x_{|V|}) = \underbrace{\frac{(\sum_{i=1}^{|V|} x_i)!}{\prod_{i=1}^{|V|} x_i!}}_{\beta} \prod_{i=1}^{|V|} p_i^{x_i} \quad \text{avec} \sum_{i=1}^{|V|} p_i = 1$$

Lissage *add-one* et MAP

- Une loi de dirichelet est une loi à priori conjuguée d'une distribution multinomiale:

$$p(\theta = (p_1, \ldots, p_{|V|}) | \alpha = (\alpha_1, \ldots, \alpha_{|V|})) = \underbrace{\frac{\Gamma\left(\sum_{i=1}^{|V|} \alpha_i\right)}{\prod_{i=1}^{|V|} \Gamma(\alpha_i)}}_{\gamma} \prod_{i=1}^{|V|} p_i^{\alpha_i - 1}$$

avec $\alpha_i > 0$

- Alors notre distribution à postériori s'exprime par:

$$\begin{aligned} p(\theta|\mathscr{D}) &\propto p(\theta) \times p(D|\theta) \\ &\propto \gamma \prod_{i=1}^{|V|} p_i^{\alpha_i - 1} \times \beta \prod_{i=1}^{|V|} p_i^{x_i} \\ &\propto \prod_{i=1}^{|V|} p_i^{x_i + \alpha_i - 1} \end{aligned}$$

Lissage *add-one* et MAP

- L'estimateur MAP consiste à trouver θ qui maximise cette quantité (ou son log) sous les contraintes stochastiques:

$$\frac{\partial}{\partial p_i}\left[\log p(\theta|\mathscr{D}) + \lambda(1 - \sum_{i=1}^{|V|} p_i)\right] = 0$$

(rappel: λ est un **coefficient de Lagrange**)

- Soit ici:

$$\frac{\partial}{\partial p_i}\left[\sum_{i=1}^{|V|}(x_i + \alpha_i - 1)\log p_i + \lambda(1 - \sum_{i=1}^{|V|} p_i)\right] = 0$$

Lissage *add-one* et MAP

- On a donc $|V|$ équations de la forme:

$$p_i = \frac{\alpha_i - 1 + x_i}{\lambda}$$

- C'est le moment de réintroduire notre contrainte:

$$\sum_{i=1}^{|V|} p_i = 1 = \sum_{i=1}^{|V|} \frac{\alpha_i - 1 + x_i}{\lambda} = \frac{1}{\lambda} \sum_{i=1}^{|V|} (\alpha_i - 1 + x_i)$$

- Donc:

$$p_i = \frac{\alpha_i - 1 + x_i}{\sum_{i=1}^{|V|} (\alpha_i - 1 + x_i)}$$

Lissage *add-one* et MAP

- Parmi les lois de Dirichlet, il est courant de choisir les α égaux (on parle alors de Dirichlet symétrique, noté D_α).

- Si nous choisissons D_2 comme a priori, alors nous retombons sur l'estimateur *add-one*:

$$p_i = \frac{x_i + 1}{\sum_{i=1}^{|V|}(x_i + 1)} = \frac{x_i + 1}{|V| + \sum_{i=1}^{|V|} x_i} = \frac{x_i + 1}{|V| + N}$$

- Si nous choisissons D_1, alors nous retombons sur notre estimateur MLE:

$$p_i = \frac{x_i}{\sum_{i=1}^{|V|}(x_i)} = \frac{x_i}{N}$$

Lissage additif

Add δ (ou loi de Lidstone)

$$p_{lid}(w_i|w_{i-1}) = \frac{|w_{i-1}w_i| + \delta}{\delta|V| + |w_{i-1}|} \qquad \delta \leq 1$$

- Si on pose:

$$\mu_{|w_{i-1}|} = \frac{1}{1 + \delta|V|/|w_{i-1}|}$$

- alors:

$$p_{lid}(w_i|w_{i-1}) = \mu_{|w_{i-1}|} \underbrace{\frac{|w_{i-1}w_i|}{|w_{i-1}|}}_{\text{MLE}} + (1 - \mu_{|w_{i-1}|}) \underbrace{\frac{1}{|V|}}_{\text{uniforme}}$$

"Poor estimate of context are worse than none" (Gale and Church 1990)

Information

$\neq$ connaissance

$=$ réduction de l'incertitude = surprise

Information(jeté de dé) $>$ Information(tirage à pile ou face)

- **Def:** Si E est un événement dont la probabilité d'apparition est $P(E)$, alors l'information associée à l'annonce que E s'est produit est:

$$I(E) = \log_2 \frac{1}{P(E)} = -\log_2 P(E)$$

quantité exprimée en nombre de bits

Information

- jeté d'une pièce non pipée: $I = \log_2 2 = 1$ bit
- jeté d'un dé non pipé: $I = \log_2 6 \approx 2.585$ bits
- choix d'un mot parmi un vocabulaire (équiprobable) de 1000 formes: $I = \log_2 1000 \approx 9.966$ bits

Note: si $P(E) = 1$ alors $I(E) = 0$

L'information est additive

L'information de deux événements indépendants est additive.

- k jetés successifs d'une pièce: $I = \log_2 \frac{1}{(1/2)^k} = k$ bits
- k jetés successifs d'un dé: $I = k \log_2 6$ bits
- un document de k mots d'un vocabulaire de 100 000 formes: $I = k \log_2 100\,000$ bits
- une image en 16 niveaux de gris 480x640, $I = 307\,200 \log_2 16 = 1\,228\,200$ bits

Entropie

- Soit S une source d'information qui émet de manière indépendante des symboles appartenant à un alphabet $s_1, \ldots, s_k$ avec les probabilités respectives $p_1, p_2, \ldots, p_k$.

- **Def:** la quantité moyenne d'information obtenue en observant la sortie de S est appelée l'entropie de S et est définie par:

$$H(S) = \sum_{i=1}^{k} p_i I(s_i) = \sum_{i=1}^{k} p_i \log_2 \frac{1}{p_i} = E\left[\log_2 \frac{1}{p(s)}\right]$$

C'est le nombre moyen de bits qu'il faut pour communiquer chaque symbole de la source

Propriétés de $H(P) = \sum_{i=1}^{k} p_i \log_2 \frac{1}{p_i}$

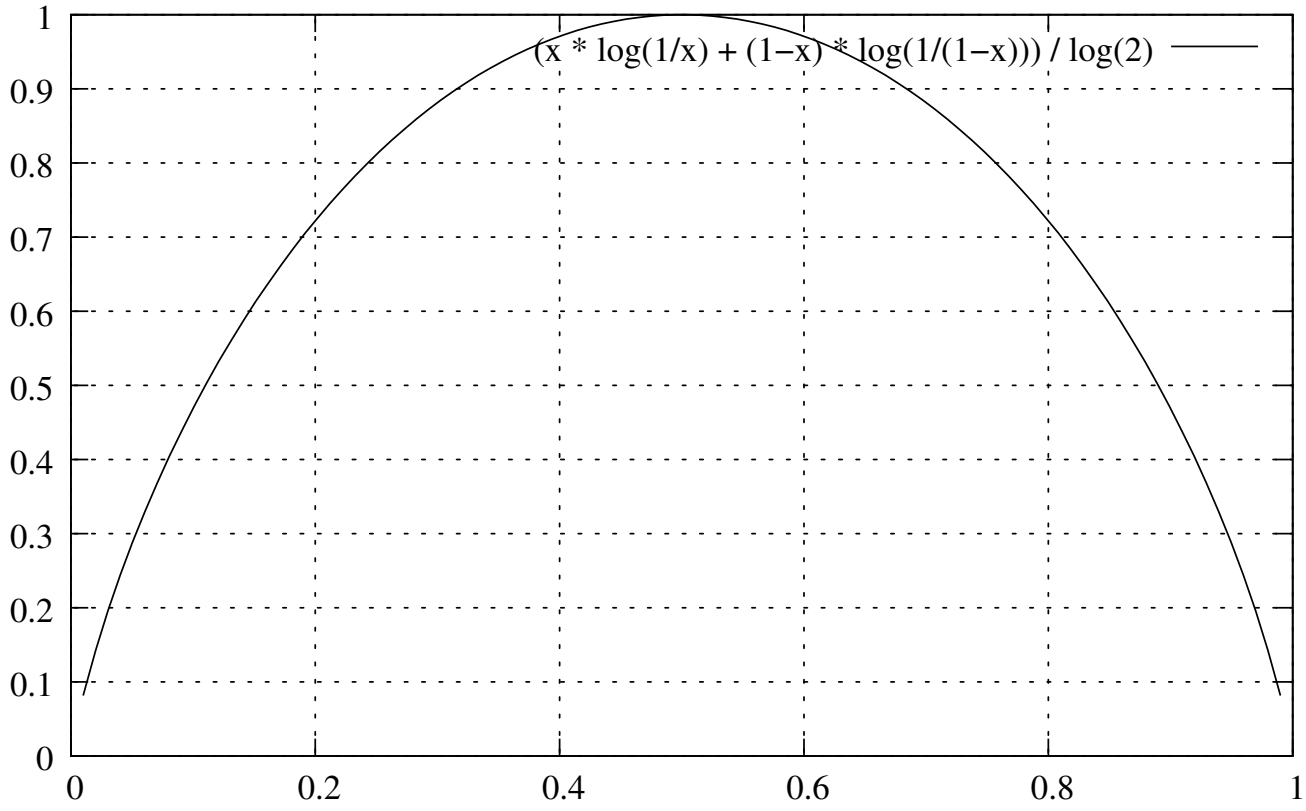
- $H(P) \geq 0$
- Pour toute distribution $q = q_1, q_2, \ldots, q_k$, sous le régime P:

$$H(P) \leq H(P, Q)$$

où $H(P, Q) = E_P[-\log_2 Q] = -\sum_x p(x) \log_2 q(x)$ est appelée l'entropie croisée

- $H(P) \leq log_2\, k$ avec égalité ssi $p_i = 1/k \quad \forall i$
- Plus P s'écarte de l'uniforme, plus l'entropie est petite (entropie = surprise moyenne)

Entropie d'une pièce plus ou moins biaisée



L'entropie sur un exemple[3]

- On veut transmettre le plus efficacement possible (au sens de la quantité d'information) le cheval gagnant d'une course de 8 chevaux, et ce à chaque course.
- Sans à priori sur la compétence des chevaux, on peut transmettre le code (sur 3 bits) du numéro du cheval gagnant:

$$1 = 001, 2 = 010, 3 = 011, \ldots, 8 = 100$$

Cela revient à dire qu'on ne se soucie pas des informations sur les chevaux: tous ont la même chance de gagner:

$$\log_2(\frac{1}{1/8}) = \log_2(8) = 3$$

[3] Extrait de (Jurafsky and Martin 2000) pp. 225

L'entropie sur un exemple

- Supposons maintenant que l'on connaît la probabilité que chaque cheval a de gagner:

$$\{\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64}\}$$

- Alors:

$$\begin{aligned} H(x) &= -\sum_{i=1}^{8} p(i) \log_2 p(i) \\ &= -\frac{1}{2}\log_2 \frac{1}{2} + \frac{1}{4}\log_2 \frac{1}{4} - \ldots - 4 \times \frac{1}{64}\log_2 \frac{1}{64} \\ &= 2 \text{ bits} \end{aligned}$$

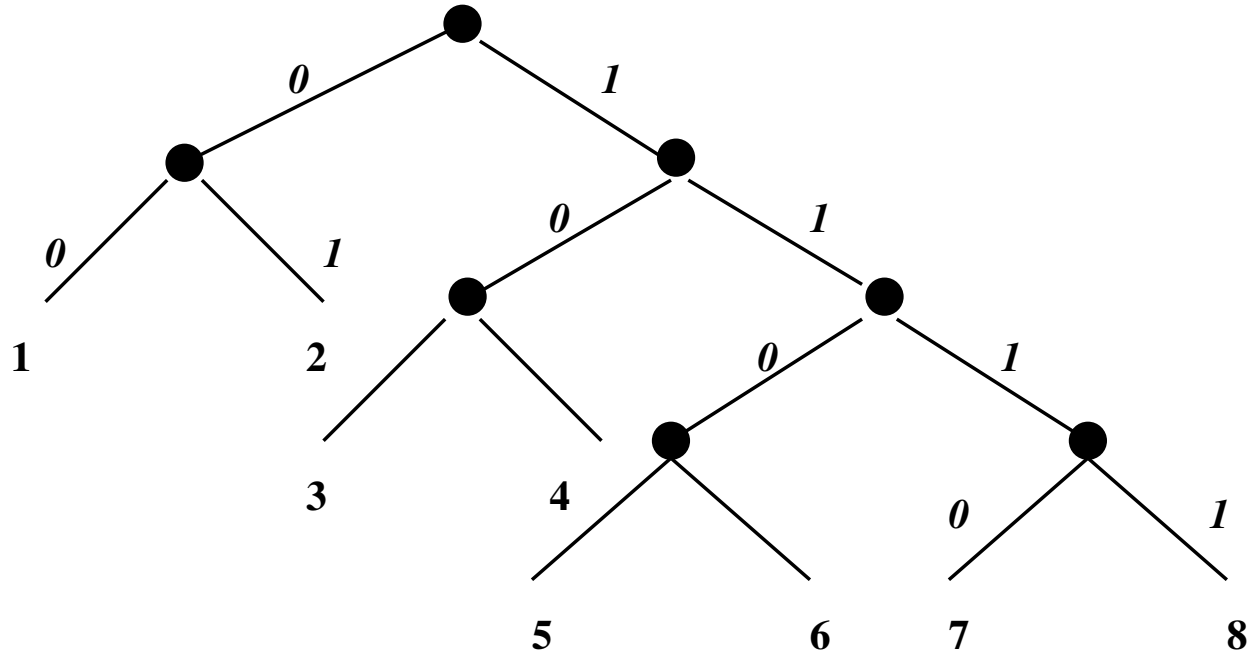
On peut coder la même chose en 2 bits **en moyenne**

- **Idée:** plus un événement est probable, moins on utilise de bits pour l'encoder.

Entropie chevaline

Exemple d'encodage

- Exemple (non optimal) de code à longueur variable pour transmettre les gagnants des courses:

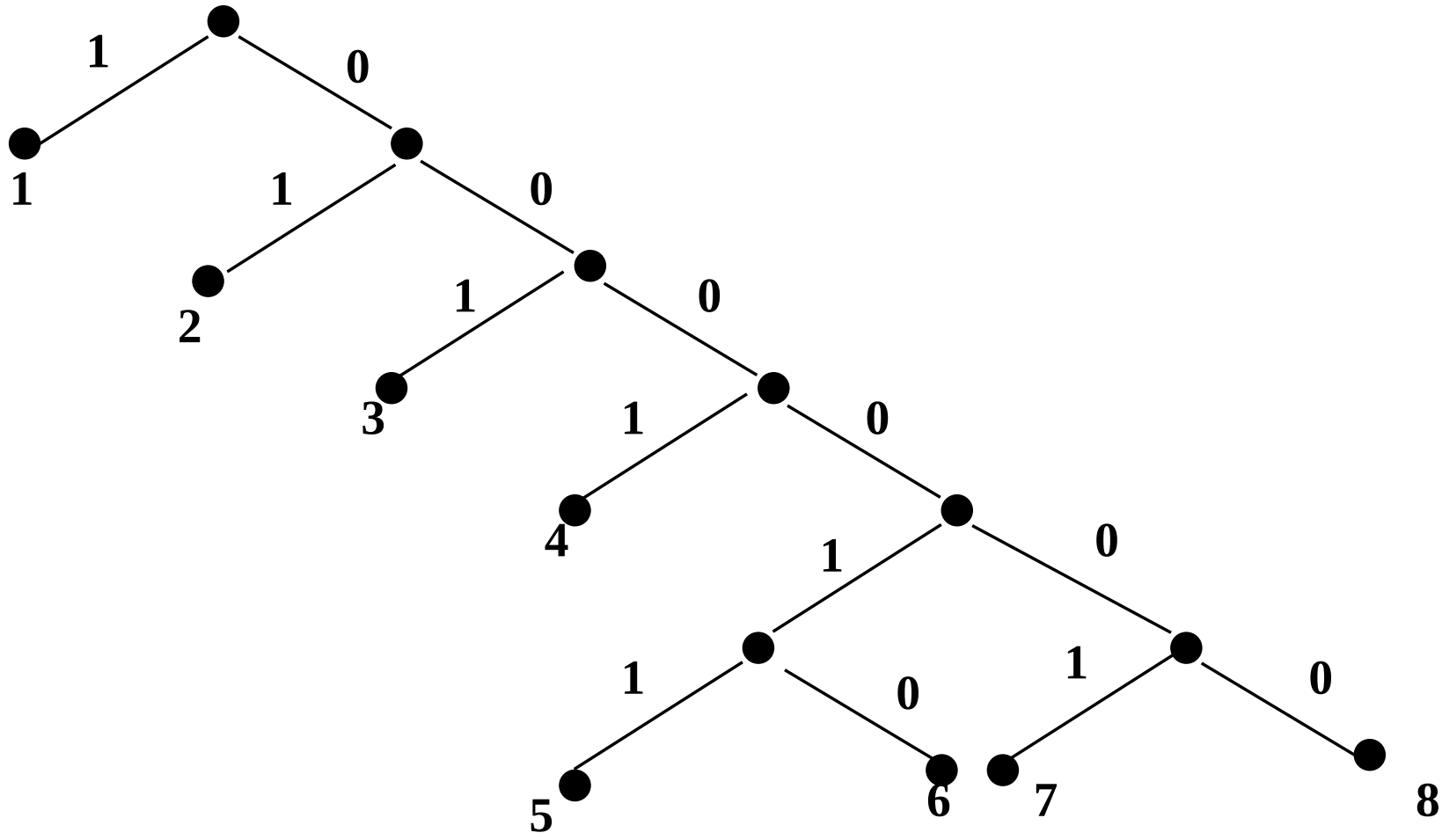


$$\frac{1}{2}\times 2+\frac{1}{4}\times 2+\frac{1}{8}\times 3+\frac{1}{16}\times 4+4\times\frac{1}{64}\times 4=2.1875$$

Un théorème de Shanon nous dit que l'on peut faire mieux en moyenne

Entropie chevaline

Codage de huffman



$$\frac{1}{2} \times 1 + \frac{1}{4} \times 2 + \frac{1}{8} \times 3 + \frac{1}{16} \times 4 + \frac{1}{64} \times 6 \times 4 = 2$$

Shannon nous dit qu'on ne peut pas faire mieux en moyenne

Mesure de la qualité d'un modèle de langue

- soit S une source sans mémoire qui émet des symboles $s_i, i \in [1,k]$ avec une probabilité p_i:

$$H(S) = -\sum_{i \in [1,k]} p_k \log p_k$$

- si on considère un message comme une instance d'une variable aléatoire W alors:

$$H(W) = -\sum_{W} p(W) \log p(W)$$

- pour les documents de n mots:

$$H(W_1^n) = -\sum_{w_1^n} p(w_1^n) \log p(w_1^n)$$

Mesure de la qualité d'un modèle de langue

- Pour ne pas dépendre de la longueur du message, on considère souvent l'entropie par mot:

$$\frac{1}{n}H(W_1^n) = -\frac{1}{n}\sum_{w_1^n} p(w_1^n)\log p(w_1^n)$$

- On parle également de l'entropie d'un langage L:

$$H(L) = -\lim_{n\to\infty}\frac{1}{n}\sum_{w_1^n} p(w_1^n)\log p(w_1^n)$$

Mesure de la qualité d'un modèle

- Pour une source quelconque:

$$H = -\lim_{n->\infty} \frac{1}{n} \sum_{x_1 \ldots x_n} p(x_1 \ldots x_n) \log p(x_1 \ldots x_n)$$

- Si la source est ergodique, alors:

$$H = -\lim_{n->\infty} \frac{1}{n} \log p(x_1 \ldots x_n)$$

- Si de plus, le jeu de test est assez grand:

$$H = -\frac{1}{n} \log p(x_1 \ldots x_n)$$

- Mais $p(x_1 \ldots x_n)$ est inconnue (c'est ce que l'on cherche)

$$LP = -\frac{1}{n} \log \hat{p}(x_1 \ldots x_n) \qquad \text{(note: } LP \geq H\text{)}$$

Cas du trigramme

- Soit un corpus de test de n phrases $\mathscr{T} = \{s_1, \ldots, s_n\}$, où $s_i = \{w_1^i \ldots w_{n_{s_i}}^i\}$ est une phrase de n_{s_i} mots. Alors:

$$
\begin{aligned}
H &= -\frac{1}{\sum_{i=1}^{n} n_{s_i}} \log p(s_1, \ldots, s_n) \\
&= -\frac{1}{\sum_{i=1}^{n} n_{s_i}} \log \prod_{i=1}^{n} p(s_i) \\
&= -\frac{1}{\sum_{i=1}^{n} n_{s_i}} \sum_{i=1}^{n} \log p(s_i) \\
&= -\frac{1}{\sum_{i=1}^{n} n_{s_i}} \sum_{i=1}^{n} \log \prod_{j=1}^{n_{s_i}} p(w_j^i | w_{j-2}^i w_{j-1}^i) \\
&= -\underbrace{\frac{1}{\sum_{i=1}^{n} n_{s_i}}}_{N} \sum_{i=1}^{n} \underbrace{\sum_{j=1}^{n_{s_i}} \log p(w_j^i | w_{j-2}^i w_{j-1}^i)}_{\log p(s_i)}
\end{aligned}
$$

Mesure de la qualité d'un modèle

- On préfère souvent présenter la qualité d'un modèle en terme de perplexité, qui représente en gros le nombre moyen d'hésitations (prédictions équiprobables) lors d'une prédiction (typiquement entre 70 et 400):

$$PP = 2^H = \hat{p}(x_1 \dots x_n)^{-\frac{1}{n}}$$

- Relation entre la réduction de l'entropie et de la perplexité (Goodman 2001):

entropie	.01	.1	.16	.2	.3	.4	.5	.75	1
perplexité	0.7%	6.7%	10%	13%	19%	24%	29%	41%	50%

Ex: si $H = 8, H' = 7.9$, alors la perplexité passe de 256 à 238.8 soit une réduction relative de la perplexité de 6.7%

!! Une réduction de perplexité de 5% (ou moins) n'est habituellement pas significative, une réduction entre 10% et 20% est souvent intéressante, enfin une réduction au delà de 30% est intéressante mais rare (Rosenfeld 2000)...

Le jeu de Shannon

- **protocole:** On demande à un humain de deviner lettre après lettre un texte. Le sujet propose en premier la lettre qu'il pense la meilleure, puis ensuite la seconde, etc... jusqu'à trouver la bonne. Il passe ensuite à la lettre suivante.
  - le sujet ne change pas son degré de connaissance au cours de l'expérience (sujet stationnaire)
  - on conserve le rang r_i de la bonne réponse pour la i-ème lettre
- $H^+ = -\sum_j Q(j) \log Q(j)$
  - où $Q(j)$ est la fréquence relative d'une prédiction au rang j.
  - H^+ est une borne supérieure de l'entropie de l'anglais (si le test est assez long).

Entropie de l'anglais

- Shannon a montré que sur un texte donné (100,000 mots) avec des mots d'en moyenne 5.5 caractères:
  - entropie par caractère de 1.3 bits (26 + 1 caractères).
  - estimée largement sous-estimée (texte trop petit)
  - modèle aveugle: $\log(27) = 4.75$ bits /caractère
- (Brown, Pietra, Pietra, Lai, and Mercer 1992) ont entraîné un gros modèle (583 millions de tokens), puis en testant sur un grand corpus de test.
  - une estimée de 1.75 bits/car.
  - surestimé (seulement un modèle)

Le lissage Good/Turing (1953)

- Pour tout n-gramme apparu r fois, on prétend qu'il est apparu r^* fois, avec:

$$r^* = (r+1)\frac{n_{r+1}}{n_r}$$

où n_r est le nombre de n-grammes apparus r fois dans $\mathscr{T}$.

- Ainsi pour tout n-gramme α on a:

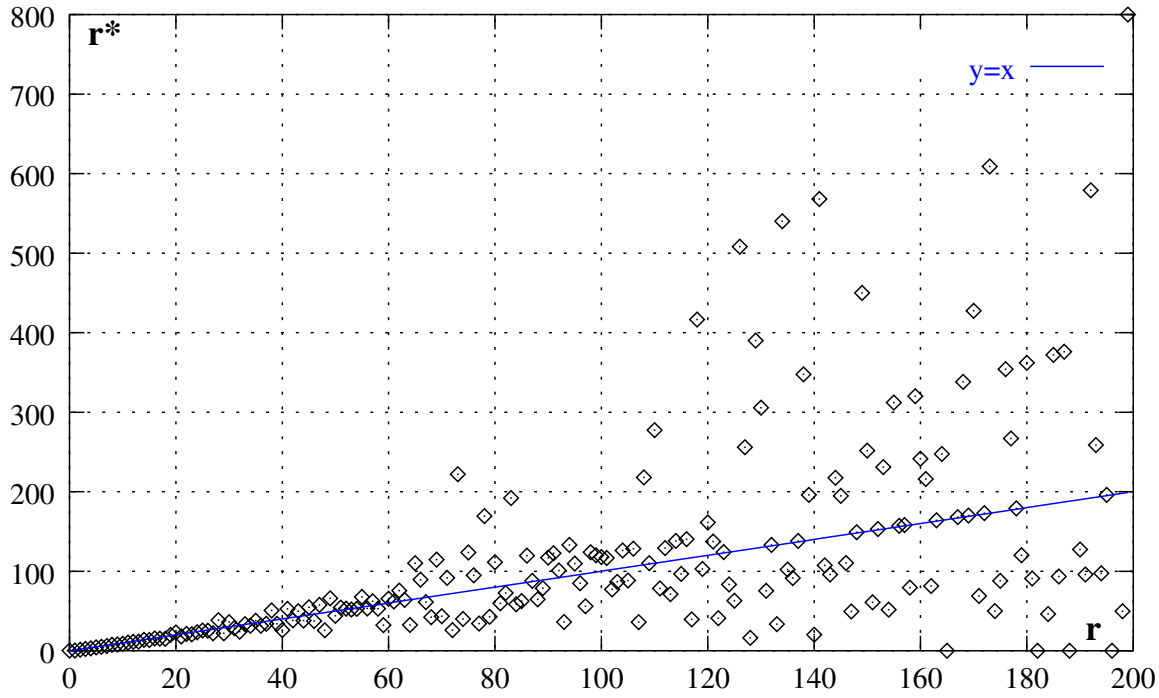
$$p_{GT}(\alpha) = \frac{r^*}{N'} \quad \text{avec } N' = \sum_{r=0}^{\infty} n_r r^*$$

!! N' est bien le compte original (N) observé dans $\mathscr{T}$, car:

$$N' = \sum_{r=0}^{\infty} n_r r^* = \sum_{r=0}^{\infty} n_r (r+1)\frac{n_{r+1}}{n_r} = \sum_{r=0}^{\infty} (r+1) n_{r+1} = \sum_{r=1}^{\infty} r n_r = N$$

Le lissage Good/Turing (1953)

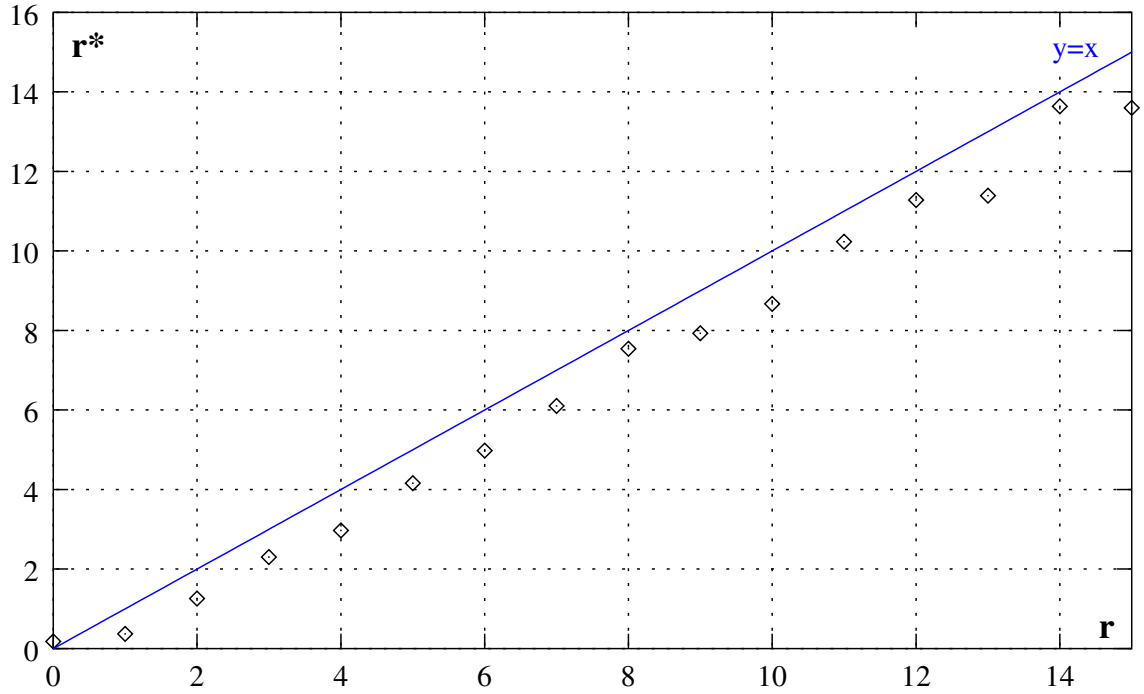
- Le nombre de n-grammes (différents) de fréquence r, lorsque r est grand, est faible (n_r est faible) $\Rightarrow$ les estimées de fréquence (r^*) ont alors tendance à être bruitées.
- Lorsqu'un n-gramme est fréquent, l'estimateur MLE est raisonnable.



Texte=Austen, $|V| = 14274$, $N = 620969$, $N1 = 68\%$ N

Le lissage Good/Turing (1953)

- !! Le n-gramme le plus fréquent (soit r sa fréquence) aurait une estimée nulle par la formule GT, car n_{r+1} est nul.
- On ne peut pas appliquer GT si l'un des comptes n_r est nul (ou alors il faut lisser ...)



- On applique Good/Turing sur les n-grammes dont la fréquence est petite.

Le lissage Good/Turing (1953)

r	n_r	r^*	r	n_r	r^*
0	203,552,865	.00065	6	2524	4.98
1	133420	.37	7	1796	6.10
2	25201	1.26	8	1371	7.54
3	10585	2.30	9	1150	7.93
4	6099	2.97	10	912	8.67

- La probabilité associée à un n-gramme non vu est $p_0 = \frac{n_1}{N \times n_0}$ où n_0 est le nombre de n-grammes non vus et N le nombre total de n-grammes dans le corpus.
- Dans l'exemple du corpus Austen, $p_0 = 1.05 \times 10^{-9}$, et la masse totale de probabilité associée à des bigrammes non vus est: 0.21

Lissage Good-Turing

- **Hyp:** un texte est composé par K unités s_k (unigrammes, bigrammes ou autres) tirées indépendamment les unes des autres avec une probabilité p_k (inconnue).
- Soit une unité α dont le compte en corpus est r. Quelle est notre estimée de p_α sur la simple information de r ?

$$\hat{p}_i = E_j[p(j=i)|r] = \sum_{j=1}^{K} p_j p(j=i|r)$$

- posons:

$$p(j=i|r) = \frac{\binom{N}{r} p_i^r (1-p_i)^{N-r}}{\sum_j \binom{N}{r} p_j^r (1-p_j)^{N-r}}$$

Lissage Good-Turing

Or, le nombre estimé d'unités ayant un compte r dans le corpus de N mots est:

$$E_N(n_r) = \sum_{j=1}^{K} \binom{N}{r} p_j^r (1-p_j)^{N-r}$$

de même:

$$E_{N+1}(n_{r+1}) = \sum_{j=1}^{K} \binom{N+1}{r+1} p_j^{r+1} (1-p_j)^{N-r} = \frac{N+1}{r+1} \sum_{j=1}^{K} \binom{N}{r} p_j^{r+1} (1-p_j)^{N-r}$$

car $\binom{N+1}{r+1} = \binom{N}{r} \times \frac{N+1}{r+1}$, donc:

$$\hat{p}_i = \frac{\sum_j \binom{N}{r} p_j^{r+1} (1-p_j)^{N-r}}{E_N(n_r)} = \frac{r+1}{N+1} \frac{E_{N+1}(n_{r+1})}{E_N(n_r)}$$

Lissage Good-Turing

$$\hat{p}_i = \frac{r+1}{N+1} \frac{E_{N+1}(n_{r+1})}{E_N(n_r)}$$

donc

$$r^\star = \frac{N \times (r+1)}{N+1} \frac{E_{N+1}(n_{r+1})}{E_N(n_r)}$$

d'où:

$$r^\star \approx (r+1) \times \frac{E_{N+1}(n_{r+1})}{E_N(n_r)} \approx (r+1) \times \frac{n_{r+1}}{n_r}$$

Lissage de Katz (1987): le modèle backoff

Idée: étend l'intuition de GT, mais en introduisant la combinaison de modèles d'ordre inférieur.

- Soit un corpus $\mathscr{T}$ tel que:
  $|\text{journal du}| = |\text{journal de}| = |\text{journal humidifiant}| = 0$
  (par exemple parce que `journal` n'est pas dans $\mathscr{T}$).
- Selon GT (et également add-one), ces bigrammes vont recevoir la même probabilité. Intuitivement, le troisième devrait être moins probable.
- En consultant un modèle unigramme on peut éventuellement parvenir à rendre compte de cette intuition qui devrait donner `humidifiant` comme moins probable (car moins fréquent).

backoff de Katz

$$p_{katz}(w_i|w_{i-1}) = \begin{cases} \hat{p}(w_i|w_{i-1}) & \text{si } |w_{i-1}w_i| > 0 \\ \alpha(w_{i-1})p_{katz}(w_i) & \text{sinon} \end{cases}$$

où $\hat{p}$ est une distribution lissée selon GT. De manière plus générale:

$$p_{katz}(w_i|w_{i-n+1}^{i-1}) = \hat{p}(w_i|w_{i-n+1}^{i-1}) + \theta(|w_{i-n+1}^{i}|)\alpha(w_{i-n+1}^{i-1})p_{katz}(w_i|w_{i-n+2}^{i-1})$$

où:

$$\theta(x) = \begin{cases} 1 & \text{si } x = 0 \\ 0 & \text{sinon} \end{cases}$$

$\hat{p}$ est obtenu en appliquant GT, *cad* un discount de l'ordre de $\frac{r^*}{r}$.

Katz (1987)

$$\forall w_{i-1} \ , \ \sum_{w_i} p_{katz}(w_i|w_{i-1}) = 1$$

$$= \sum_{w_i:|w_{i-1}^i|>0} \hat{p}(w_i|w_{i-1}) + \sum_{w_i:|w_{i-1}^i|=0} \alpha(w_{i-1}) p_{ML}(w_i)$$

donc:

$$1 - \sum_{w_i:|w_{i-1}w_i|>0} \hat{p}(w_i|w_{i-1}) = \alpha(w_{i-1}) \sum_{w_i:|w_{i-1}w_i|=0} p_{ML}(w_i)$$

cad:

$$\alpha(w_{i-1}) = \frac{1 - \sum_{w_i:|w_{i-1}w_i|>0} \hat{p}(w_i|w_{i-1})}{1 - \sum_{w_i:|w_{i-1}w_i|>0} p_{ML}(w_i)}$$

Katz (1987)

Some boring details

- Pour calculer $\hat{p}$, on applique un discount $d_r \approx \frac{r^*}{r}$ pour les comptes faibles: $d_r = 1$ pour $\forall r > k$. Katz suggère $k = 5$.
- Dans le cas d'un bigramme, les discounts sont choisis tel que:
  - les discounts sont proportionnels à ceux de GT: $1 - d_r = \mu(1 - \frac{r^*}{r})$
  - le nombre total de counts discountés de la distribution globale est égal au nombre total de counts que GT assigne aux bigrammes non vus: $\sum_{r=1}^{k} n_r(1 - d_r)r = n_1$

- En résolvant ces $k+1$ équations, on trouve (merci Good):

$$d_r = \frac{\frac{r^*}{r} - \frac{(k+1)n_{k+1}}{n1}}{1 - \frac{(k+1)n_{k+1}}{n1}}$$

Katz (1987)

k | r	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
3	.24	.55	.72																	
4	.28	.57	.73	.70																
5	.29	.58	.74	.71	.81															
6	.31	.59	.75	.72	.82	.81														
7	.32	.59	.75	.72	.82	.81	.86													
8	.32	.59	.75	.72	.82	.81	.86	.94												
9	.33	.60	.75	.72	.82	.82	.86	.94	.87											
10	.34	.61	.75	.73	.82	.82	.86	.94	.87	.86										
11	.34	.61	.75	.73	.82	.82	.86	.94	.87	.86	.93									
12	.34	.61	.76	.73	.82	.82	.86	.94	.87	.86	.93	.94								
13	.35	.61	.76	.73	.83	.82	.87	.94	.88	.86	.93	.94	.87							
14	.35	.61	.76	.73	.83	.82	.87	.94	.88	.86	.93	.94	.87	.97						
15	.35	.61	.76	.73	.83	.82	.87	.94	.88	.86	.93	.94	.87	.97	.90					
16	.35	.61	.76	.73	.83	.82	.87	.94	.88	.86	.93	.94	.87	.97	.90	.96				
17	.36	.62	.76	.73	.83	.82	.87	.94	.88	.86	.93	.94	.87	.97	.90	.96	.89			
18	.36	.62	.76	.74	.83	.82	.87	.94	.88	.86	.93	.94	.87	.97	.90	.96	.89	.82		
19	.36	.62	.76	.74	.83	.82	.87	.94	.88	.86	.93	.94	.87	.97	.90	.96	.89	.82	1.06	
20	.36	.62	.76	.74	.83	.82	.87	.94	.88	.86	.93	.94	.87	.97	.90	.96	.89	.82	1.06	1.15

- Discounts accordés en fonction de k sur le corpus Austen pour les bigrammes.
- Plus r est grand, plus d_r se rapproche de 1 ($d_1 \approx 0.3, d_8 \approx 0.9$)

Le lissage Jelinek-Mercer (1980)

- **Idée:** proche du backoff mais ici, on consulte les modèles d'ordre inférieur tout le temps.

$$p_{interp}(w_i|w_{i-n+1}^{i-1}) = \lambda_{w_{i-n+1}^{i-1}} p_{ML}(w_i|w_{i-n+1}^{i-1}) +$$
$$(1 - \lambda_{w_{i-n+1}^{i-1}}) p_{interp}(w_i|w_{i-n+2}^{i-1})$$

- Pour arrêter la récursion, on peut combiner avec un modèle 0-gram ($p_{unif}(x) = \frac{1}{|V|}$) ou un modèle unigramme.

Jelinek-Mercer

Estimer λ – take 1

- **Intuitivement:** si un historique h est fréquent dans $\mathscr{T}$, alors λ_h devrait être grand; faible sinon.

Ex on rencontre souvent `Monsieur le` dans le Hansard. On peut à priori donner du poids au modèle trigramme. En revanche, on ne rencontre jamais `pédaler dans`. On peut donc dans ce cas ci donner plus de poids sur les modèles d'ordre inférieur.

Jelinek-Mercer

Estimer λ – take 1

- Les λ ne dépendent pas de l'historique:

$$p_{interp}(w_i|w_{i-2}w_{i-1}) = \begin{cases} \alpha p_{ML}(w_i|w_{i-2}w_{i-1}) + \\ \beta p_{ML}(w_i|w_{i-1}) + \\ \gamma p_{ML}(w_i) + \\ \lambda \frac{1}{|V|} \end{cases}$$

avec $\alpha + \beta + \gamma + \lambda = 1$ car:

$\sum_w p_{interp}(w|w_{i-2}w_{i-1}) = 1 =$

$$\alpha \overbrace{\sum_w p_{ML}(w|w_{i-2}w_{i-1})}^{1} + \beta \overbrace{\sum_w p_{ML}(w|w_{i-1})}^{1} + \gamma \overbrace{\sum_w p_{ML}(w)}^{1} + \lambda \overbrace{\sum_w \frac{1}{|V|}}^{1} =$$

$$\alpha + \beta + \gamma + \lambda$$

Jelinek-Mercer

Estimer λ – take 2

Partitionner l'espace des historiques

- (Jelinek and Lafferty 1991) suggère de partitionner en fonction de $\sum_{w_i} |w_{i-n+1}^{i}|$, cad, la fréquence de l'historique. Par exemple: $\lambda(h) = \lambda(\log(|h|))$

- (Chen 1996), suggère de partitionner selon: $\frac{\sum_{w_i} |w_{i-n+1}^{i}|}{|\{w_i : |w_{i-n+1}^{i}| > 0\}|}$, cad de prendre en compte le nombre de mots différents qui peuvent suivre un historique donné.

Ex sur une tranche du Hansard, 2 mots suivent le bigramme `projet de`: `loi` et `modification`. En revanche, 31 mots différents suivent le bigramme `la chambre`: `des`, `a`, `aujourd'hui`, `comprend`, etc.

Jelinek-Mercer

Estimer λ – take 2

- Un algorithme dérivé de Baum-Welch (1972) permet d'apprendre les λ à partir d'un corpus. C'est un cas particulier de l'algorithme EM (Expectation, Maximization).
- **Important:** Il faut prendre un autre corpus que $\mathscr{T}$ pour estimer les λ. Sinon l'algorithme va trouver la réponse optimale qui est de donner le poids maximal au modèle le meilleur.
  - un corpus held-out ou encore de développement de 10% à 25% de la taille de $\mathscr{T}$.
  - Que faire si on manque de corpus?
    - validation croisée:

    Ex: On divise $\mathscr{T}$ en deux parties (ex: 80%/20%). Sur le 1er corpus on fait l'estimée des modèles, sur le 2nd, on fait l'estimée des λ. On recommence pour une autre division de $\mathscr{T}$. Les estimées finales pondèrent les estimées partielles.

Lissage Witten-Bell (1990)

- Peut être vu comme une instance de Jelinek-Mercer:

$$p_{WB}(w_i|w_{i-n+1}^{i-1}) = \lambda_{w_{i-n+1}^{i-1}} P_{ML}(w_i|w_{i-n+1}^{i-1}) + (1-\lambda_{w_{i-n+1}^{i-1}}) p_{WB}(w_i|w_{i-n+2}^{i-1})$$

- mais au lieu de partionner l'espace des λ, on calcule la quantité:

$$1-\lambda_h = \frac{N_{1+}(h)}{N_{1+}(h)+\sum_w |hw|}$$

où $N_{1+}(h) = |\{w : |hw| > 0\}|$ est le nombre de mots différents qui peuvent suivre l'historique.

Lissage Witten-Bell (1990)

- **Intuitivement:** $1 - \lambda$ est une approximation de la probabilité avec laquelle on devrait écouter le(s) modèle(s) d'ordre inférieur.
  - c'est une estimée de la probabilité qu'un mot w suive un historique h alors qu'on a jamais vu hw dans $\mathscr{T}$. C'est le MLE obtenu sur un corpus étendu avec un événement qui dit "je n'ai encore jamais été vu après h"
  - Imaginez que pour chaque historique on passe de gauche à droite sur le corpus $\mathscr{T}$ et qu'à chaque fois qu'on rencontre un mot w qui suit cet historique, alors on regarde si oui ou non ce mot a déjà été vu après l'historique.
- Donc au final:

$$p_{WB}(w_i|w_{i-n+1}^{i-1}) = \frac{|w_{i-n+1}^{i}|+N_{1+}(w_{i-n+1}^{i-1})p_{WB}(w_i|w_{i-n+2}^{i-1})}{\sum_{w_i}|w_{i-n+1}^{i}|+N_{1+}(w_{i-n+1}^{i-1})}$$

Note Méthode de lissage proposée initialement pour la compression de textes.

Lissage Absolute-Discounting (Ney, Essen, and Kneser 1994)

- Encore une méthode d'interpolation, mais on retire de chaque compte positif une quantité fixe D ($D \leq 1$).

$$p_{abs}(w_i|w_{i-n+1}^{i-1}) = \frac{max\{|w_{i-n+1}^{i}|-D,0\}}{\sum_{w_i}|w_{i-n+1}^{i}|} + (1-\lambda_{w_{i-n+1}^{i-1}})p_{abs}(w_i|w_{i-n+2}^{i-1})$$

- Pour obtenir une distribution de probabilité, on a: $\sum_{w_i} p_{abs}(w_i|w_{i-n+1}^{i-1}) = 1$, ce qui entraîne:

$$1-\lambda_{w_{i-n+1}^{i-1}} = \frac{D \times N_{1+}(w_{i-n+1}^{i-1})}{\sum_{w_i}|w_{i-n+1}^{i}|}$$

- Ney et al. suggèrent d'utiliser: $D = \frac{n_1}{n_1+2n_2}$

Note: Ca ressemble pas mal à Witten-Bell, avec une autre estimée du poids que l'on devrait attribuer au modèle d'ordre inférieur.

(Kneser and Ney 1995)

Vous commencez à être tannés ?

- Une extension du lissage par absolute discounting, où le modèle d'ordre inférieur est estimé de façon plus adaptée:

$$p_{KN}(w_i|w_{i-n+1}^{i-1}) = \frac{max\{|w_{i-n+1}^{i}|-D,0\}}{\sum_{w_i}|w_{i-n+1}^{i}|} + \frac{D}{\sum_{w_i}|w_{i-n+1}^{i}|} \times N_{1+}(w_{i-n+1}^{i-1}) p_{KN}(w_i|w_{i-n+2}^{i-1})$$

- avec:

$$p_{KN}(w_i|w_{i-n+2}^{i-1}) = \frac{|\{w_{i-n+1}: \ |w_{i-n+1}^{i}|>0\}|}{\sum_{w_i}|\{w_{i-n+1}: \ |w_{i-n+1}^{i}|>0\}|}$$

- En quoi est-ce plus adapté ?
  - Le nombre de contextes différents dans lesquels se produit w_{i-n+2}^{i} est consulté; avec l'idée que si ce nombre est faible, alors on devrait accorder une petite probabilité au modèle $(n-1)$-gram, et ce, même si w_{i-n+2}^{i} est fréquent.

Lissage Kneser-Ney

Exemple (p_{kn} unigramme)

- Dans un sous-corpus du Hansard de $N = 153,213$ tokens ($|V| = 7986$ types), on rencontre 28 fois le bigramme `présidente suppléante`, mais jamais le bigramme `souveraineté suppléante`. Le nombre de bigrammes différents dans le corpus est 16 057.
  - "Habituellement", $p(\text{suppléante}|\text{souveraineté}) \propto p(\text{suppléante}) = \frac{28}{153213} (\approx 0.00018)$
    28 étant la fréquence de *suppléante*.
  - Dans Kneser-Ney, elle est plutôt proportionnelle à: $\frac{|\{\text{présidente}\}|}{16057} (\approx 6.22 \times 10^{-5})$,

Lissage Kneser-Ney

Exemple (p_{kn} bigramme)

- Toujours dans ce même corpus, on ne voit jamais le trigramme `stupide le président`, alors qu'on rencontre 524 fois le trigramme monsieur le président. Le bigramme le président a été vu 738 fois.
  - Avec Kesner-Ney:

    $$p(\text{président}|\text{stupide le}) \propto \frac{|\{\bullet\text{le président}\}|}{|\{\bullet\text{le}\bullet\}|} = \frac{7}{2421} (\approx 0.0029)$$

  - alors que dans d'autres approches, c'est plutôt proportionnel à $p(\text{président}|\text{le}) = \frac{738}{|\text{le}|=4425} (\approx 0.166)$

Quelle technique choisir ?

- Lire (Chen and Goodman 1996) et (Goodman 2001).
  - Kneser & Ney parmi les meilleures techniques de lissage
  - Katz est une méthode qui marche d'autant mieux que les comptes initiaux sont grands.
  - Jelinek & Mercer est une valeur sure adaptée aux comptes plus faibles.
- La réponse à cette question dépend beaucoup de l'application visée: de sa langue, de son registre, de l'évolution du vocabulaire à travers le temps, des ressources mémoire disponibles, etc.

Quelques facteurs à prendre en compte

- Taille du corpus: The more the better...
  mais les techniques de lissage plafonnent.
  - un modèle bigramme sature au delà de quelques centaines de millions de mots.

  Ex (Rosenfeld 2000): en comptant les trigrammes présents dans un corpus de 38 millions de mots d'articles de journaux, il a observé que sur de nouveaux textes de la même source, plus d'un tiers des trigrammes étaient nouveaux.
- Nature des corpus (adaptation)

  Ex (Rosenfeld 2000): un ML entraîné sur les textes du "Dow-Jones news" voit sa perplexité doubler s'il est testé sur des textes (pourtant assez semblables pour un humain) de l' "Associated Press newswire" de la même époque.

Augmenter *n*

- augmenter *n* augmente le problème de sous-représentation des données.
- la traduction statistique utilise (souvent) $n = 5$ (résultats très proches avec *n*=4). Dans ce cas, la technique de lissage prend de l'importance:
  - Katz semble peu adaptée (meilleure pour les gros comptes).
  - Kneser & Ney est plus adaptée: les performances ne chutent pas lorsque *n* augmente.
- si $|\mathscr{T}|$ n'est pas très grand, alors un modèle bigramme et un modèle trigramme ont à peu près la même performance.

Augmenter la taille des données

Stupid Backoff (Brants, Popat, Xu, Och, and Dean 2007)

- un modèle backoff sans normalisation !

$$s(w_i|w_{i-n+1}^{i-1}) = \begin{cases} \frac{|w_{i-n+1}^{i}|}{|w_{i-n+1}^{i-1}|} & si \quad |w_{i-n+1}^{i}| > 0 \\ \alpha S(w_i|w_{i-n+2}^{i-1}) & sinon \end{cases}$$

- S indique un score et non une probabilité (simplement des comptes), α fixé (à 0.4)
- approche map-reduce pour calculer efficacement les fréquences et pour y accéder rapidement lors des tests
  ▷ modèle distribué (sur plusieurs machines)

Augmenter la taille des données

Stupid Backoff (Brants, Popat, Xu, Och, and Dean 2007)

- 5-gramme entraîné sur des corpus de 13M à 2T (10^{12}) de mots . . . (google !)
- training sets (anglais):

target	237M mots (LDC), partie anglaise de corpus parallèles En-Ar (mixte de différents types de textes)
ldcnews	5G mots (LDC), news
webnews	31G mots collectés sur des pages de news
web	2T mots collectés du Web en janvier 2006

- test set: 1797 phrases postérieures à la période sur laquelle les données d'entraînement ont été collectées. Mélange de différents genres (news, newsgroups, etc.) (tâche: traduction)

Augmenter la taille des données

Stupid Backoff (Brants, Popat, Xu, Och, and Dean 2007)

- temps vs taille:

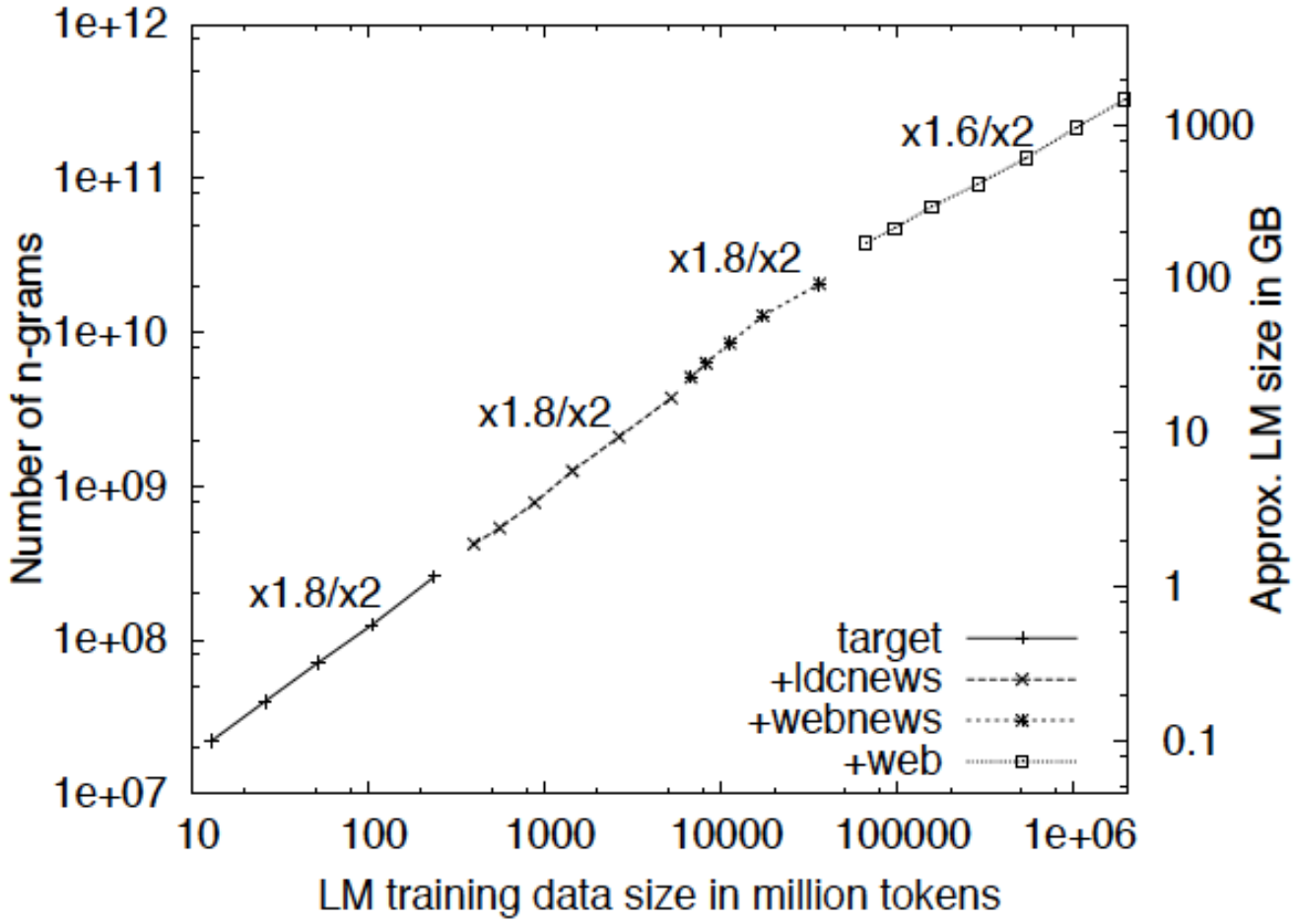
	target	webnews	web
$\|tokens\|$	237M	31G	1.8T
$\|V\|$	200k	5M	16M
$\|n-gram\|$	257M	21G	300G
$\|LM\|$ (SB)	2G	89G	1.8T
time (SB)	20 min	8h	1 day
time (KN)	2.5h	2 days	1 week[4]
# machines	100	400	1500

- cutoff sur le vocabulaire de 2 pour target et ldcnews et 200 pour webnews.
- $|n-gram|$ nombre de n-grammes différents pour $n \in [1,5]$ (pas de cutoff pour les n-grammes).

[4] temps estimé

Augmenter la taille des données

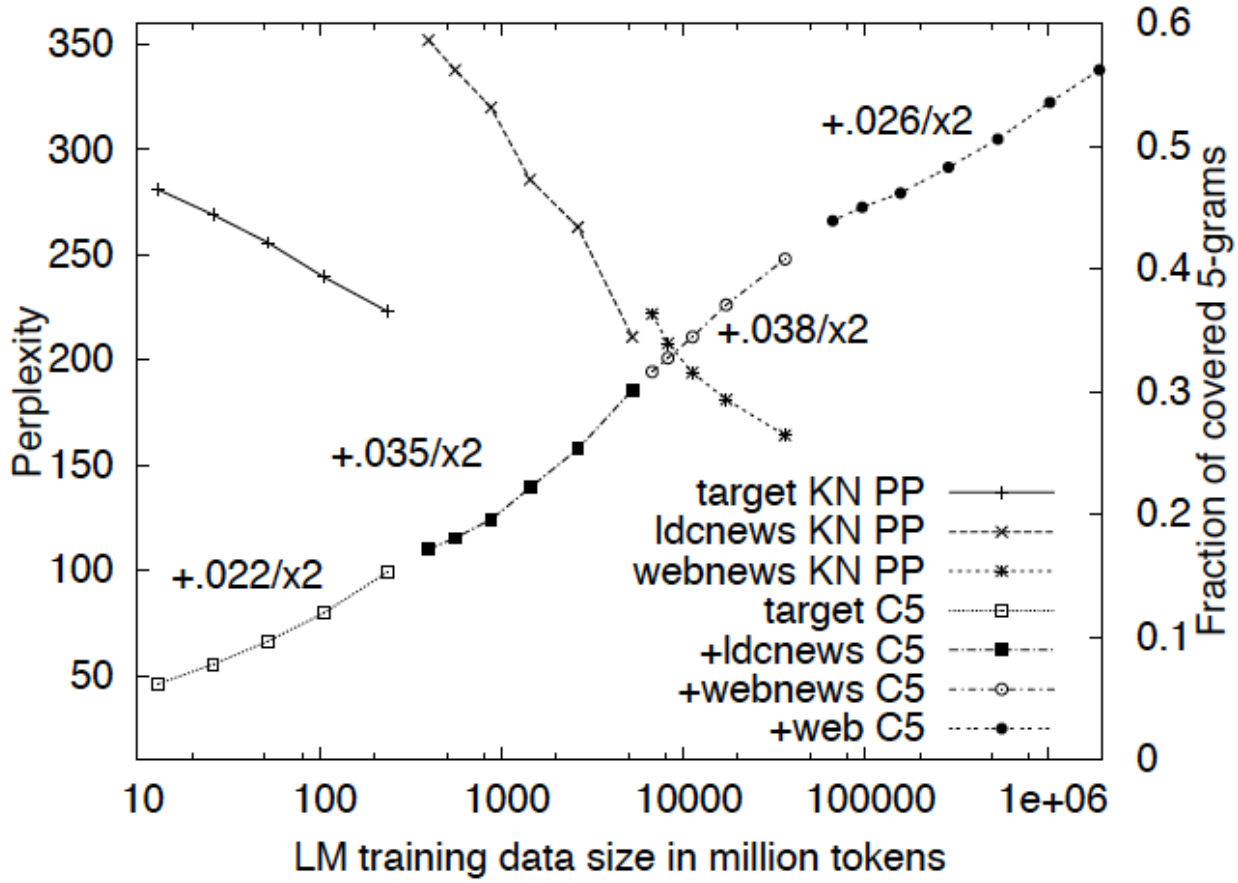
Stupid Backoff (Brants, Popat, Xu, Och, and Dean 2007)



- données x 2 ▷ nombre de n-grammes x 1.8

Augmenter la taille des données

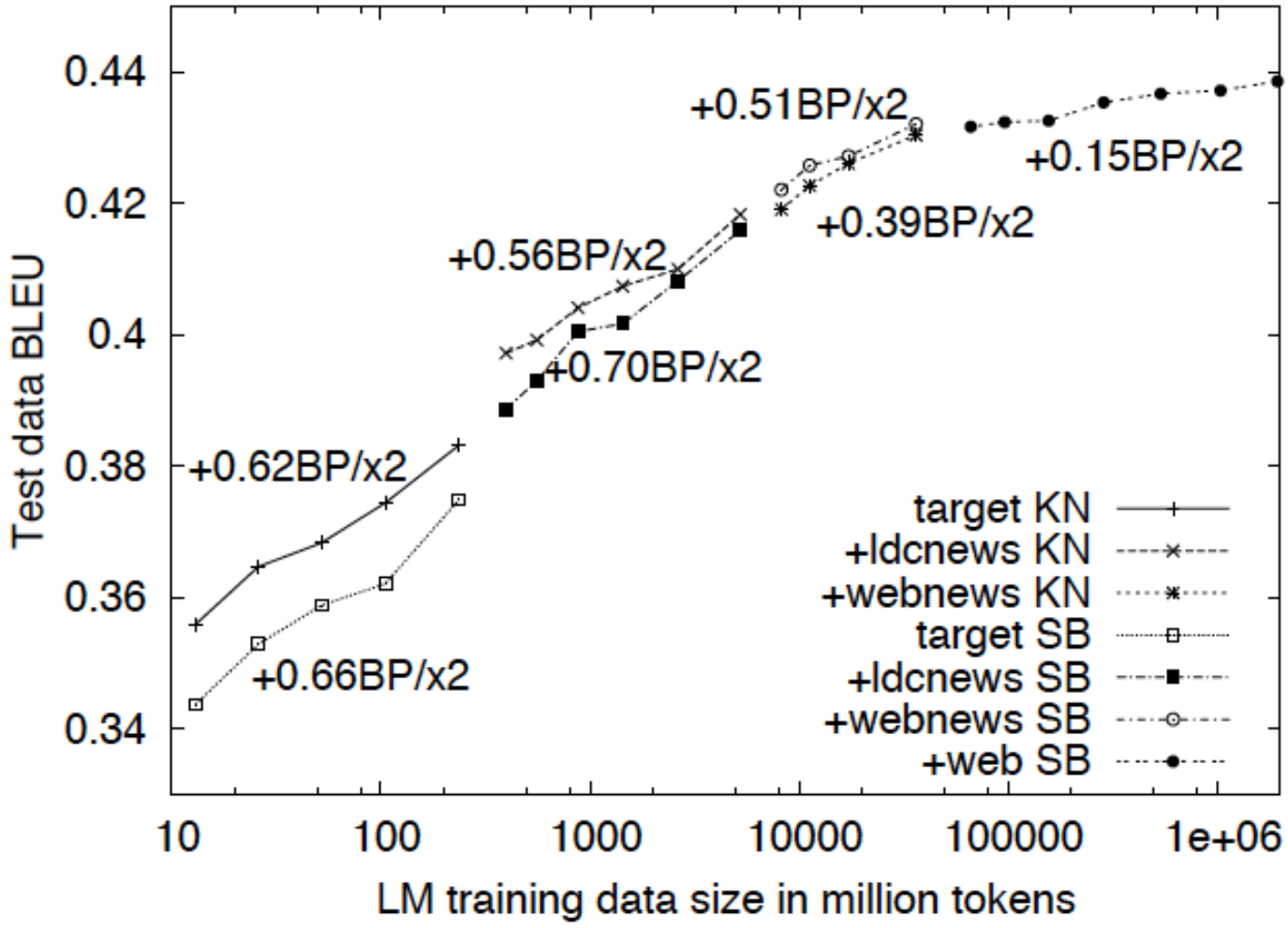
Stupid Backoff (Brants, Popat, Xu, Och, and Dean 2007)



- Perplexité (pour KN) et couverture (fraction des 5-grammes du test présents)
- Les perplexités ne sont pas directement comparables (les modèles n'ont pas le même vocabulaire)

Augmenter la taille des données

Stupid Backoff (Brants, Popat, Xu, Och, and Dean 2007)



- BLEU $\sim$ distance entre une traduction produite et une (ici 4) traduction de référence (1 signifie l'identité).

Retour à la vie de tous les jours

- Google Search par la voix:
  - modèle 3-gramme, 13.5 M n-grammes
  - 1.0/8.2/4.3M 1/2/3-grammes
  - 4bytes/n-gramme à une vitesse raisonnable (3 fois le temps d'accès à un modèle non compressé)

- Googler: *Back-Off Language Model Compression*
  http://static.googleusercontent.com/external_content/untrusted_dlcp/research.google.com/fr//archive/papers/chelba-talk.pdf

Le skipping

- **Idée:** Éliminer des éléments du contexte conditionnant
  - Si $|\texttt{il parle avec JEAN de lui}| > 0$,
  - mais $|\texttt{il parle avec PIERRE de lui}| = 0$,
  - alors on estime: $p(\texttt{lui} \mid \texttt{il parle avec --- de})$
- En pratique on combine ensemble différents modèles skippés avec un modèle non skippé, donc on augmente la complexité du modèle.
- C'est une technique qui peut servir à créer également des modèles n-gramme (n grand) du pauvre:

$$p_4(w_i|w_{i-3}w_{i-2}w_{i-1}) =$$
$$\lambda_1 p_2(w_i|w_{i-2}w_{i-1}) + \lambda_2 p_2(w_i|w_{i-3}w_{i-1}) + \lambda_3 p_2(w_i|w_{i-3}w_{i-2})$$

Le skipping

- cette technique marche assez mal ou n'apporte pas grand chose.
- Pourquoi ?

$$\text{manger une} \left\{ \begin{array}{l} \text{pomme} \\ \text{poire} \\ \text{pêche} \end{array} \right\} \text{avec} \left\{ \begin{array}{l} \text{délice} \\ \text{délectation} \\ \text{gourmandise} \end{array} \right\}$$

Mais:

$$\text{manger une} \left\{ \begin{array}{l} \text{raclée} \\ \text{volée} \\ \text{araignée} \end{array} \right\} \text{avec} \quad ???$$

Clustering (regroupement)

Voir (Brown:92)

- Séduisant par son double but (en fait ils sont liés):
  - Réduire le problème de sous représentation des données
  - Introduire de l'information (linguistique)

Ex on observe dans $\mathscr{T}$ des occurrences de `[départ {mardi, mercredi} à 15 h.]`, mais pas celle de `[départ jeudi à 18 h.]` $\Longrightarrow$ `[départ` *JOUR* `à` *NOMBRE* `h.]`

- Si un mot est associé à une seule classe, alors on parle de clustering dur (hard clustering); sinon on parle de clustering mou (soft clustering).

  [missile, rocket, bullet, gun] [officer, aide,chief, manager] [lundi, mardi, mercredi, jeudi, vendredi, samedi,dimanche]

Clustering

- Plusieurs possibilités; entre autres:

$$
\begin{aligned}
p(w_3|w_1 w_2) &\approx p(w_3|C_3)p(C_3|w_1 w_2) \\
&\approx p(w_3|C_3)p(C_3|w_1 C_2) \\
&\approx p(w_3|C_3)p(C_3|C_1 C_2) \\
&\approx p(w_3|C_1 C_2)
\end{aligned}
$$

- Pour des domaines restreints (ex: réservation de billets d'avions), on peut obtenir de bonnes performances si on fait des clusters à la main.
- Dans des domaines plus ouverts, seule une approche automatique permet d'obtenir des améliorations (de l'ordre de 10%), pour autant que l'on combine avec un modèle entraîné sur les mots (on ne réduit pas la taille des modèles).

Modèles adaptatifs

- Situation typique:
  - Un gros corpus $\mathscr{T}_h$ (hors-data) pour entraîner un modèle “général”
  - Un plus petit corpus $\mathscr{T}_a$ plus représentatif de l’application.
    En fait, la combinaison des deux modèles donne habituellement des performances décevantes en comparaison du modèle entraîné directement sur $\mathscr{T}_a$ . . .

Ex (Rosenfeld 2000):

- $\mathscr{T}_a =$ Switchboard (25 millions de mots);
- $\mathscr{T}_g = \{$WSJ (40 millions de mots), BN (140 millions de mots)$\}$.

- perf$(M_g + M_a) \approx$ perf(M_a) !

  En fait, 1 million de mots supplémentaires dans switchboard aide mieux que 30 millions de mots hors-domaine (généraux).

Modèles adaptatifs

Le modèle cache (Kuhn and Mori 1990)

- l'historique des données de test — qui grandit au fur et à mesure — est utilisé pour créer (en ligne) un modèle n-gramme $p_{cache}(w|h)$ qui est combiné au modèle statique:

$$p_{adapt}(w|h) = \lambda p_{stat}(w|h) + (1-\lambda)p_{cache}(w|h)$$

- (Goodman 2001) observe pour des corpus assez grands, une réduction d'entropie de 13% avec des modèles caches bigrammes et trigrammes.

Le modèle cache (Kuhn and Mori 1990)

- Dans une véritable application (ex: reconnaissance de la parole), l'historique est bruité (contient des erreurs de reconnaissance).
- Le système peut se bloquer sur des erreurs[5]:

USER	donne-moi l'heure
RECO	donne-moi le beurre
USER	non je veux avoir l'heure
RECO	non je veux avoir le beurre
USER	l'heure est grave
RECO	le beurre est gras
...	...

[5] Exemple factice bien que probable.

Modèles adaptatifs

Adaptation intra-domaine

- Textes d'entraînement hétérogènes (ex: dépêches AFP).
- **Idée:** découper $\mathscr{T}$ en N sujets (topics), et entraîner individuellement N modèles "spécialisés". Lors de la reconnaissance, chercher en permanence le topic actuel et (par exemple) combiner le modèle spécialisé, avec un modèle générique.

Modèles adaptatifs

(Iyer and Ostendorf 1999)

- Faisons l'hypothèse qu'il y a S types de phrases différents dans un corpus. Le type de la phrase est une variable cachée et on a un à priori sur le type de phrase δ_j (avec $\sum_{j=1}^{S} \delta_j = 1$).

$$p(w_i|h) = \sum_{j=0}^{S} \delta_j \prod_{i=1}^{N} \lambda_j \underbrace{p(w_i|h;j)}_{\text{un type}} + (1-\lambda_j) \underbrace{p(w_i|h)}_{\text{optionnel}}$$

- En pratique le type de la phrase est appris au moment de l'entraînement
- Les auteurs montrent une implantation avec 8 types de phrases. (Goodman 2001) rapporte des améliorations si on prend S grand (64 ou 128).
- Entraîne une division du corpus d'entraînement.

Maximum Entropy Modeling

- Au début de l'histoire, il y a la famille exponentielle:

$$p(w|h) = \frac{1}{\mathscr{Z}(h)} \exp \sum_i \lambda_i f_i(h, w)$$

où

  - $\mathscr{Z}(h)$ est un terme de normalisation, ou fonction de partition:
    $(\mathscr{Z}(h) = \sum_w \exp \sum_i \lambda_i f_i(h, w))$;
  - les λ_i sont des réels (paramètres) qui sont appris et qui pondèrent les fonctions f_i que l'on appelle généralement les traits (features).

- Il existe des algorithmes pour apprendre les poids des traits (λ_i) .

Maximum Entropy Modelling

- Exemple de trait:

$$f_x(w_1^i) = \begin{cases} 1 & \text{si } \exists (j,j') \in [1,i] / \begin{cases} 1 \leq j < j' \leq i \\ j' - j < 10 \\ w_j = \text{microsoft} \\ w_{j'} = \text{bug} \end{cases} \\ 0 & \text{sinon} \end{cases}$$

si *microsoft* apparaît dans l'historique, suivi dans une fenêtre de 10 mots (au plus), par *bug*, alors le feature est activé (il s'agit d'un **trigger**)

- Chaque trait impose des contraintes sur les modèles possibles. L'apprentissage (coûteux en temps) cherche le modèle le plus lisse (entropie maximale) parmi l'ensemble des modèles qui vérifient les contraintes imposées par les traits.

Maximum Entropy Modeling

- (Rosenfeld 2000) rapporte un gain en perplexité de 39% (énorme) comparé à un modèle cache et un trigramme interpolé **mais** il utilise l'information des triggers dans le modèle gagnant seulement.
- (Goodman 2001) rapporte qu'avec le même type d'information, ME ne fonctionne pas mieux que la combinaison "classique" de modèles.
- Lire (Berger, Pietra, and Pietra 1996) pour une bonne exposition à cette méthode d'apprentissage.

Sentence Maximum Entropy models

(Rosenfeld 1997)

- On ne décompose plus la distribution en sous distributions (rappelez-vous l'équation exacte: $p(w_{i=1}^n) = \prod_{i=1}^n p(w_i|h_i)$), mais on modélise directement la distribution jointe:

$$p(s) = \frac{1}{\mathscr{Z}} p_0(s) \exp \sum_k \lambda_k f_k(s)$$

- $\mathscr{Z}$ est cette fois-ci constant et $p_0(s)$ est un modèle de départ (par exemple votre meilleur n-gramme)

Pro Les traits f_k peuvent exploiter au mieux la cohérence de s.

Cons Les algorithmes d'entraînement sont lourds. On déplace finalement le problème sur le problème non trivial de la découverte des traits pertinents.

Modèle connexioniste

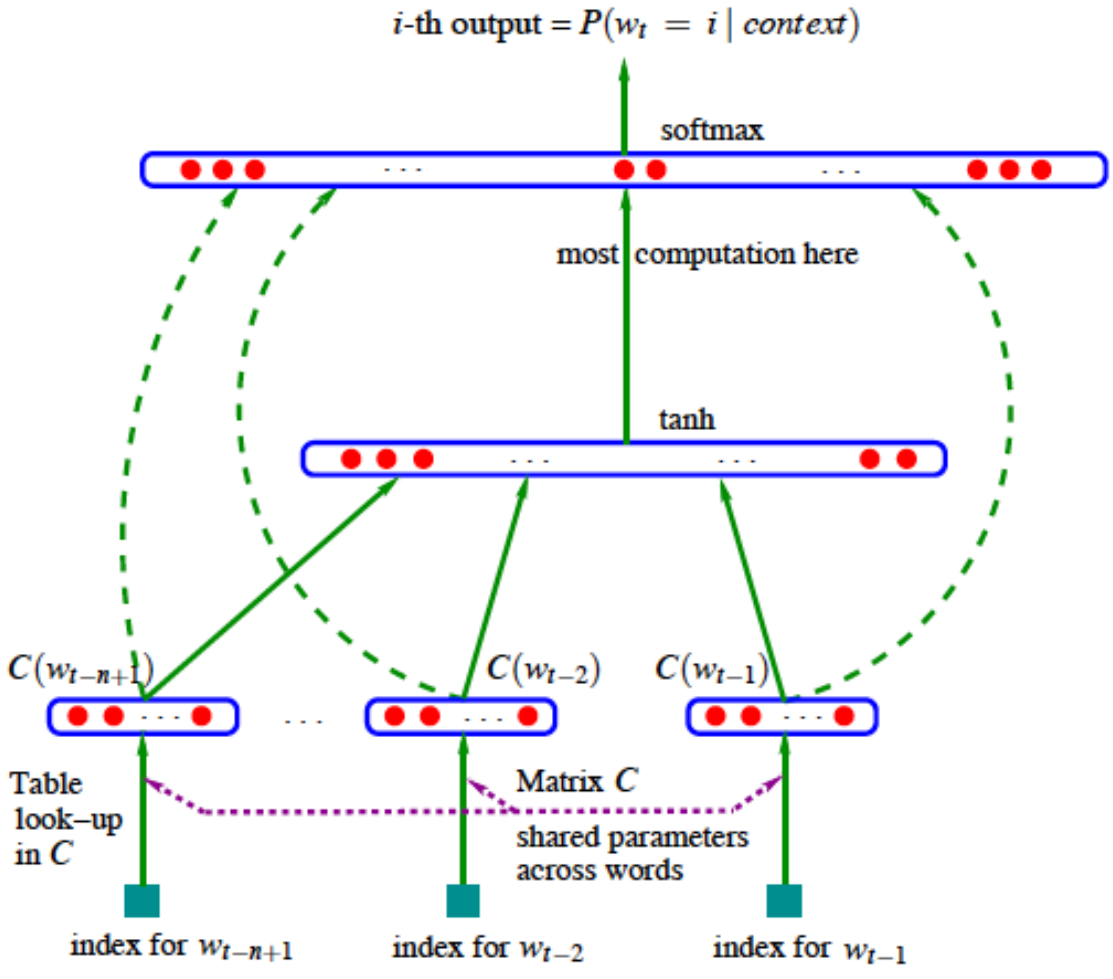
(Bengio, Ducharme, and Vincent 2001)

The cat is walking in the bedroom
A dog was running in a room
The cat is running in a room
A dog is walking in a bedroom

- Chaque mot est projeté dans un espace vectoriel; la projection est **apprise** sur le corpus d'entraînement:
  mot = vecteur
- La probabilité jointe de séquences de mots est estimée à partir de cette projection.
- Généralise mieux, car des petits changements dans les vecteurs de traits associés à chaque mot entraînent un petit changement en probabilité.
- De bonnes améliorations lorsque combiné à un trigramme interpolé.

Modèle connexioniste

(Bengio, Ducharme, and Vincent 2001)



Modèle connexioniste

(Bengio, Ducharme, and Vincent 2001)

- pro: dépend faiblement de la taille de l'historique, généralise mieux
- cons: la couche de sortie (nécessite le calcul de toutes les probabilités $p(w|h)$)
- Beaucoup de travaux sur ces modèles. Lire (Le, Allauzen, and Yvon 2012) pour un bon tour d'horizion.

Modèle connexioniste

(Le, Allauzen, and Yvon 2012)

- testé en traduction
- baseline: Moses avec modèle 4-gramme Kneser-Ney
- résultats:

modèle	ppx	BLEU		
		alone	replace	add
baseline	90	29.4	31.3	–
4-gramme	92	29.8	31.1	31.5
6-gramme	82	30.2	31.6	31.8
8-gramme	78	30.6	31.6	31.8
10-gramme	77	30.5	31.7	31.8
récurrent	81	30.4	31.6	31.8

alone pas de score de traduction (seulement LM)
replace scores LM venant du réseau
add scores LM venant du réseau et de KN

Modèle analogique

(Gosme and Lepage 2011)

- Travail encore préliminaire
  - testé sur des corpus d'entraînement de taille modeste (moins de 400 000 phrases)
  - pour des modèles 3-gramme seulement
  - pro: beaucoup de potentiel pour amélioration

- **Idée**: les *n*-grammes inconnus du jeu de test peuvent être reconstruits par analogie à partir des 3-grammes hapax du corpus d'entraînement

  [opportunité de servir : opportunité de modifier :: qui pourrait servir : qui pourrait modifier]

Modèle analogique (Gosme and Lepage 2011)

Pourcentage de 3grams inconnus reconstructibles par analogie

- $|\text{TRAIN}|$ = 90 000 phrases, $|\text{TEST}|$ = 10 000 phrases (corpus Europarl)

	3grams inconnus du test		
		λ	reconstruits μ
anglais	114 566	60.04%	83.67%
français	116 922	57.81%	81.87%
allemand	140 226	68.97%	72.14%
finnois	132 931	83,33%	44.93%

- μ = % des types de 3grams inconnus du test reconstructibles par analogie
- λ proportion des types de 3grams inconnus du test par rapport au nombre de types de 3grams du corpus d'entraînement

Modèle analogique (Gosme and Lepage 2011)

Patrons analogiques pour les 3grams

- Calculé sur 10 000 phrases par langue:

n	A	:	B	::	C	:	D	%
1	abc		abd		efc		efd	12.6
2	abc		ade		bcf		def	9.1
3	abc		dbc		efa		efd	3.1
4	abc		aec		bcd		ecd	2.7
5	abc		abd		bce		bde	2.6
⋮				⋮				

- Ex de patron 2:

[opportunité de servir : opportunité pour dire :: de servir le : pour dire le]

a b c a d e b c f d e f

- Les 5 premiers patrons sont suffisants, les 2 premiers couvrent $\geq$90% des reconstructions.

Modèle analogique (Gosme and Lepage 2011)

Effectifs des patrons

- **Intuition**: les 3grams d'effectifs semblables ont tendance à apparaître dans les mêmes analogies

- Les analogies d'effectifs 1 sont suffisantes:

	1	2	3
anglais	96.24	98.53	99.14
français	94.07	97.52	98.49
allemand	94.01	97.36	98.42
finnois	91.02	96.18	97.80

effectif d'une analogie: fréquence (dans TRAIN*) du mot le plus fréquent dans l'analogie*

Modèle analogique (Gosme and Lepage 2011)

Lissage proposé (N = taille du TRAIN, V = vocabulaire de 3grams)

- 3grams connus t: $\frac{|t|+1}{N+\delta\times|V|}$
- 3grams inconnus reconstructibles : $\frac{1-\alpha}{N+\delta\times|V|}$, α proche de 0 (effectif proche 1)
- 3grams inconnus non reconstructible: $\frac{\alpha}{N+\delta\times|V|}$

Donc:

- un lissage "à la add-one" en distinguant les 3grams inconnus reconstructibles (favorisés) des autres.
- α fixé à 10^{-6} dans les expériences
- δ déterminé de façon à ce que $\sum_t p(t) = 1$, fonction de:
  - λ, μ (qui sont calculés à partir d'un 1/10 du TRAIN non utilisé pour calculer $|\bullet|$)
  - et α (fixé)

Modèle analogique (Gosme and Lepage 2011)

Performance

- comparé à *add-one*, witten-bell, good-turing, et kneser-ney
- 11 langues (da, de, el, en, es, fi, fr, it, nl, pt, sv)
- gains en perplexité allant de modestes à importants sur toutes les langues, sauf le finnois (langue où le nombre de 3grams reconstructibles par analogie sur les mots est faible)

Bémol:

- 383 000 phrases par langue pour l'entraînement
  (pas très grand)
- le temps de lissage dépend (faiblement) de la taille du TRAIN
  ($\sim$ 100 phrases traitées/sec.; 10 fois moins que SRILM)

Packages

- SRILM - The SRI Language Modeling Toolkit
  (Stolcke 2002)
  http://www.speech.sri.com/projects/srilm/

- The CMU-Cambridge Statistical Language Modeling Toolkit
  (Clarkson and Rosenfeld 1997)
  http://mi.eng.cam.ac.uk/~prc14/toolkit.html

- Voir aussi (Pauls and Klein 2011)
  http://code.google.com/p/berkeleylm/
  ($\sim$ speed, 1/4 de la mémoire requise par SRILM)

SRILM

Exemple

- création d'un modèle 3-gramme de type KN

```
$path/ngram-count -interpolate
        -kndiscount -order 3 -unk
        -text $train
        -lm $out/mykn.3g.lm
        -write-vocab $out/mykn.vcb
```

où (par exemple):

  - $path vaut `/u/pift6010/bin/64/srilm/`
  - $train vaut `/home/www-perso/usagers/felipe/HTML/IFT6010-Automne2011/resources/tp1/train.fr`
  - $out vaut `/u/chezmoi/tp1/model/`

- consultation des probabilités

```
$path/ngram -lm $out/mykn.3g.lm -ppl -debug 1 $dev
```

où (par exemple):

  - $dev vaut `/home/www-perso/usagers/felipe/HTML/IFT6010-Automne2011/resources/tp1/train.fr`

Références I

Bengio, Yoshua, Rejean Ducharme, and Pascal Vincent (2001). “A Neural Probabilistic Language Model”. In: *Advances in Neural Information Processing Systems*.

Berger, Adam, Stpehn Della Pietra, and Vincent Della Pietra (1996). “A maximum entropy Approach to Natural Language Processing”. In: *Computational Linguistics* 22(1), pp. 39–71.

Brants, Thorsten, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean (2007). “Large Language Models in Machine Translation”. In: *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Prague, Czech Republic: Association for Computational Linguistics, pp. 858–867.

Brown, P.F., S.A. Della Pietra, V.J. Della Pietra, J.C. Lai, and R.L. Mercer (1992). “An estimate of an upper bound for the entropy of English”. In: *Computational Linguistics* 18(1), pp. 31–40.

Chen, Stanley F. (1996). *Building Probabilistic Models for Natural Language*. URL: `citeseer.nj.nec.com/chen96building.html`.

Références II

Chen, Stanley F. and Joshua Goodman (1996). “An Empirical Study of Smoothing Techniques for Language Modeling”. In: *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*. Ed. by Arivind Joshi and Martha Palmer. San Francisco: Morgan Kaufmann Publishers, pp. 310–318. URL: citeseer.nj.nec.com/chen96empirical.html.

Clarkson, Philip and Ronald Rosenfeld (1997). “Statistical Language Modeling Using the CMU-Cambridge Toolkit”. In: *Proc. Eurospeech ’97*. Rhodes, Greece, pp. 2707–2710.

Gale, W. and K. W. Church (1990). “Poor estimates are worse than none”. In: *proceedings of DARPA Speech and Natural Language Workshop*. Hidden Valley, PA, pp. 283–287.

Goodman, Joshua (2001). “A Bit of Progress in Language Modeling”. In: *Computer Speech and Language*, pp. 403–434.

Gosme, Julien and Yves Lepage (2011). “Structure des trigrammes inconnus et lissage par analogie”. In: *TALN*. Montpellier, France.

Iyer, Rukmini and Mari Ostendorf (1999). “Modeling long distance dependence in language: Topic mixture vs. dynamic cache models”. In: *IEEE Transactions on Speech and Audio Processing* 7:30, pp. 30–39.

Références III

Jelinek, Fred and John D. Lafferty (1991). “Computation of the Probability of Initial Substring Generation by Stochastic Context-Free Grammars”. In: *Computational Linguistics* 17, pp. 315–324.

Jurafsky, Daniel and James H. Martin (2000). *Speech and Language Processing*. Prentice Hall.

Kneser, Reinhard and Hermann Ney (1995). “Improved backing-off for m-gram language modeling”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing*. Vol. 1, pp. 181–184.

Kuhn, Roland and Renato De Mori (1990). “A cache-based natural language model for speech reproduction”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-12(6), pp. 570–583.

Le, Hai-Son, Alexandre Allauzen, and François Yvon (2012). “Measuring the Influence of Long Range Dependencies with Neural Network Language Models”. In: *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*. Montréal, Canada, pp. 1–10.

Manning, Christopher D. and Hinrich Schütze (1999). *Foundations of Statistical Natural Language Processing*. MIT Press.

Références IV


Ney, Hermann, Ute Essen, and Reinhard Kneser (1994). “On structuring probabilistic dependences in stochastic language modeling”. In: *Computer, Speech, and Language* 8, pp. 1–38.


Pauls, Adam and Dan Klein (2011). “Faster and Smaller N-Gram Language Models”. In: *Proceedings of the 49th ACL/HLT*. Portland, Oregon, USA, pp. 258–267.


Rosenfeld, R. (1997). “A Whole Sentence Maximum Entropy Language Model”. In: *Proceedings of the IEEE Workshop on Speech Recognition and Understanding*.


— (2000). *Two decades of statistical language modeling: Where do we go from here*. URL: `citeseer.nj.nec.com/rosenfeld00two.html`.


Stolcke, Andreas (2002). “SRILM - An Extensible Language Modeling Toolkit”. In: *Intl. Conf. Spoken Language Processing*. Denver, Colorado.