Quelques sources qui ont inspiré 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

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 suite de N mots (phrase ?), $s = w_1 \dots w_N$, 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

Décomposition par la règle de chaînage

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

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

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

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 n-gramme MLE: intuitif

Soit $\mathcal{T} = \{w1 \ldots w_N\}$, un corpus de N mots ($N \propto 10^6$):

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

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

- 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-gram:

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

Estimateur n-gramme MLE

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

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

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

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

Augmenter le corpus d'entraînement (1/2)

Étude de cas: le corpus Austen (disponible sur la page web de Manning and Schütze [1999]) contient 8762 phrases, 620968 tokens et 14274 types.

En théorie, il existe:

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

Mais, 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

Augmenter le corpus d'entraînement (2/2)

Autre étude de cas: le corpus Hansard (une partie)

Contient 1,639,250 phrases, 33,465,362 tokens et 103,830 types.

En théorie, il existe:

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

$\Longrightarrow$ lissage des probabilités ($i.e$ donner une probabilité à des choses non vues dans $\mathcal{T}$)

Modèle n-gramme: précisions

• Pour éviter les traitements spéciaux en début de séquence, on ajoute $n-1$ codes de début de phrase (ex: DDP, BOS, <s>, etc.).

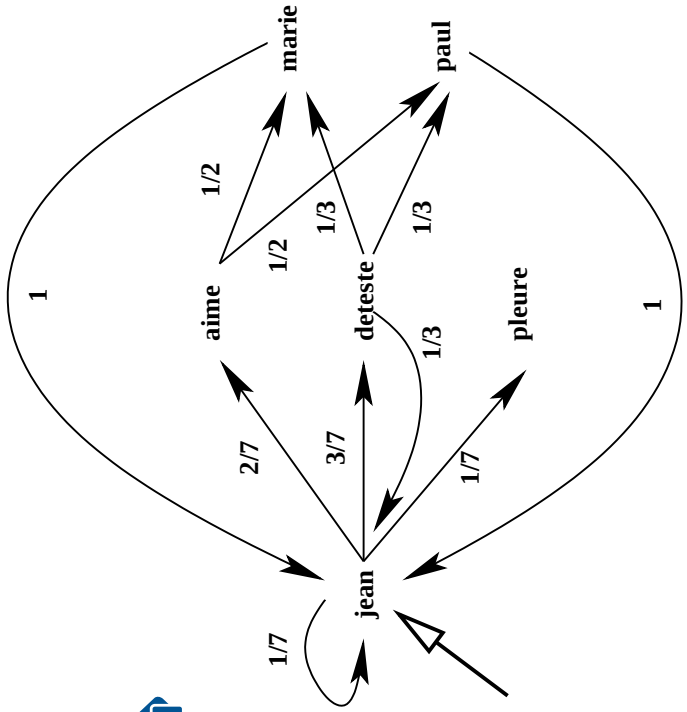
• On ajoute habituellement aussi un marqueur de fin de phrase (ex: FDP, EOS, </s>, etc.).

Soit le corpus d'entraînement $\mathcal{T}$ constitué des 6 phrases:

Jean aime Marie	(P_1)
Jean aime Paul	(P_2)
Jean déteste Marie	(P_3)
Jean déteste Paul	(P_4)
Jean déteste Jean	(P_5)
Jean pleure	(P_6)

Modèle n-gramme: précisions

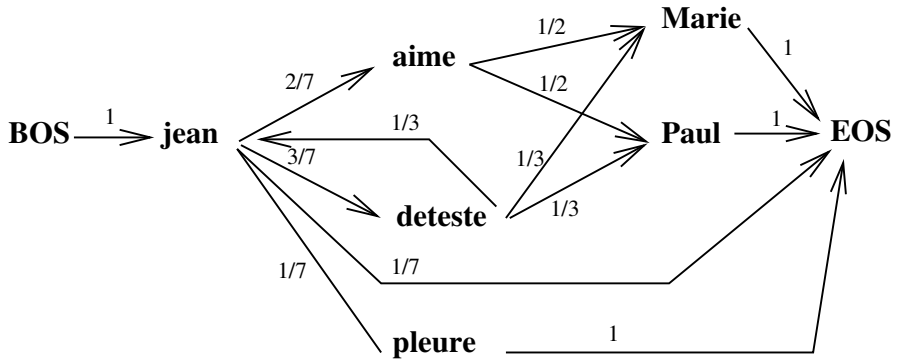
Jean aime Marie Jean aime Paul jean déteste Marie Jean déteste Paul Jean déteste Jean Jean pleure



p(J) = 1	p(Ja) = 2/7	p(JaM)=1/7
	p(Jd) = 3/7	p(JaP)=1/7
	p(Jp) = 1/7	p(JdM)=1/7
	p(JJ) = 1/7	p(JdP)=1/7
		p(JdJ)=1/7
		p(JJa)=2/49
		p(JJd)=3/49
		p(JJJ) = 1/49
		p(JJp)= 1/49

Modèle n-gramme: précisions

Jean aime Marie EOS Jean aime Paul EOS Jean déteste Marie EOS Jean déteste Paul EOS Jean déteste Jean EOS Jean pleure EOS



p(Jean)=1/7
p(Jean pleure)=1/7

p(Jean déteste Jean pleure) = 1/49
. . .

p(Jean déteste Marie) = 1/7
p(Jean déteste Paul) = 1/7
p(Jean déteste Jean) = 1/49
p(Jean aime Marie) = 1/7
p(Jean aime Paul) = 1/7

Une première 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 $\mathcal{T}$ (soit $|V|$ est 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 $\mathcal{T}$, nous prétendons cependant que nous l'avons trouvé une fois.

Note: c'est l'estimateur qu'on obtient si l'on assume une distribution uniforme *a priori* sur les événements dans une logique d'**inférence bayésienne**.

Le add-one smoothing, Vincent et Virginie

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

Maintenant, $p(\text{Vincent aime les fleurs}) = 1/3718 \approx 0.000269$ (avant 0.111) car:

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

Le add-one smoothing, Vincent et Virginie

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

Maintenant, $p(\text{Virginie aime les fleurs})$ vaut $\approx 6.72\ 10^{-5}$, car:

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

$\star$ probabilité changée par rapport au calcul précédant.

Problème avec le add-one smoothing

Retour au corpus $Austen$ qui contient $N = 620,968$ tokens 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 $n_0 = 203,552,865$.

Faisons pour changer de l'estimation jointe ($p(w_{i-1}w_i)$ au lieu de $p(w_i|w_{i-1})$), alors:

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

La probabilité d'un bigramme non rencontré dans $\mathcal{T}$ est:

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

La masse totale de probabilité associée à des bigrammes non vus est donc $n_0 \times p_0 \approx 0.996$

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

Lissage additif ++: add δ: (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:

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

$\Longrightarrow$ "Poor estimate of context are worse than none" Gale and Church [1990]

Éléments de théorie de l'information[1]

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.

[1] D'après un tutoriel de Rosenfeld.

Éléments de théorie de l'information

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 $I(E) = 0$

Éléments de théorie de l'information

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

Éléments de théorie de l'information

Entropie

Soit une source d'information S 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:

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

Nombre moyen de bits qu'il faut pour communiquer chaque symbole de la source

Éléments de théorie de l'information

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

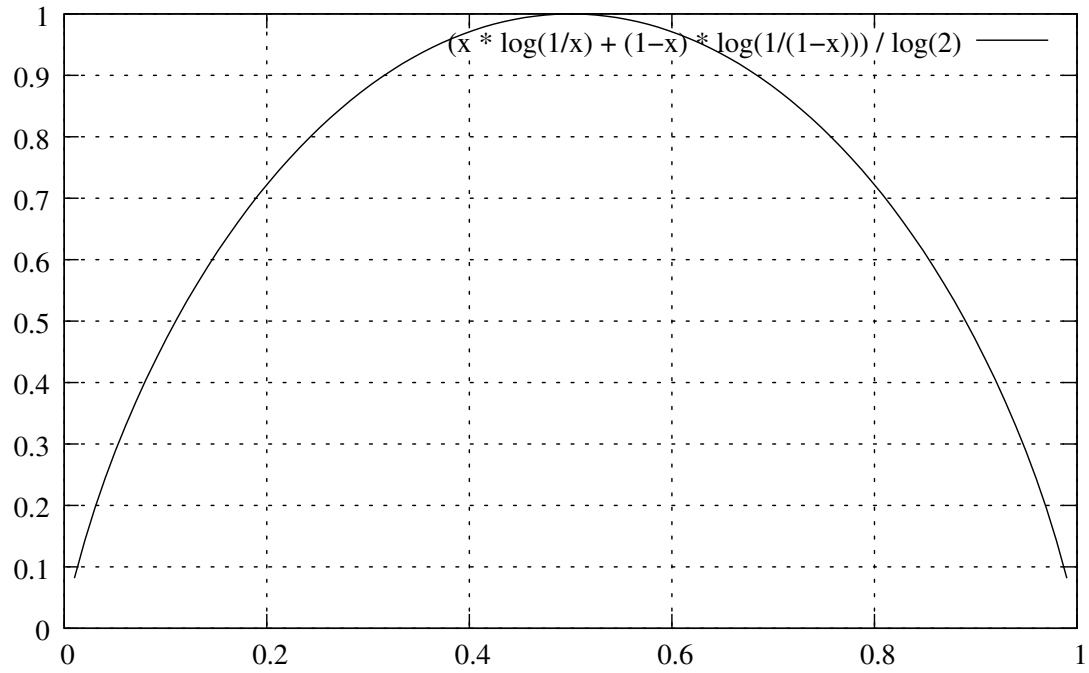
- $H(P) \geq 0$
- pour toute distribution $q = q_1, q_2, \ldots, q_k$, sous le régime P

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

  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)

Éléments de théorie de l'information

Entropie d'une pièce plus ou moins biaisée



L'entropie sur un exemple[2]

On veut transmettre le plus efficacement possible (au sens quantité d'information) le cheval gagnant d'une course de 8 chevaux, et ce à chaque course.

• Sans $a\ priori$ 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 $(\log_2(\frac{1}{1/8}) = \log_2(8) = 3)$

[2]Extrait de Jurafsky and Martin [2000] pp. 225

L'entropie sur un exemple

- On connaît la probabilité que chaque cheval a de gagner:

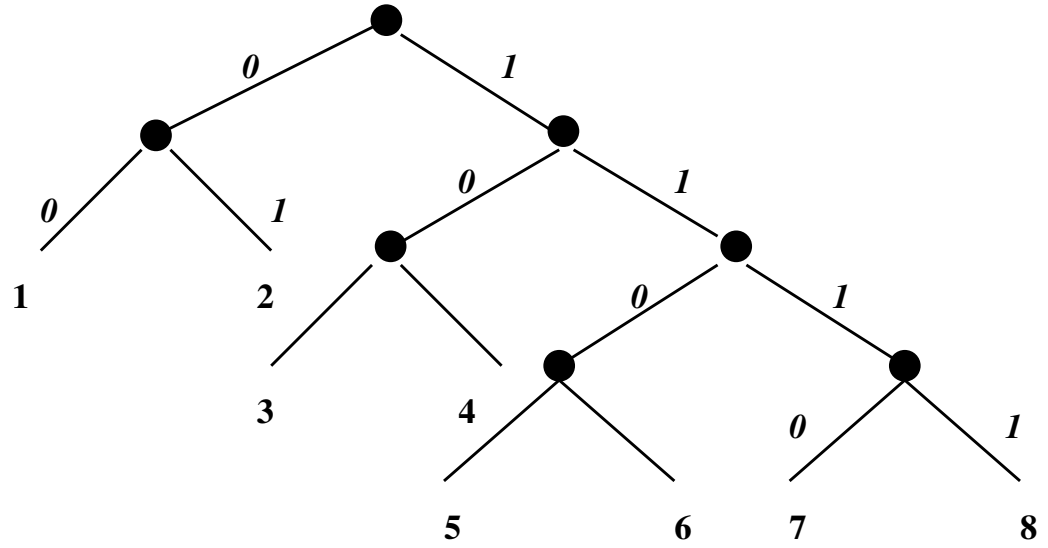
$$\{\tfrac{1}{2}, \tfrac{1}{4}, \tfrac{1}{8}, \tfrac{1}{16}, \tfrac{1}{64}, \tfrac{1}{64}, \tfrac{1}{64}, \tfrac{1}{64}\}$$

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

$\Longrightarrow$ On peut coder la même chose en 2 bits **en moyenne**. Idée: plus qqchose est fréquent, moins on utilise de bits pour le coder.

Entropie chevaline: exemple de codage

Voici un exemple (non optimal) de code à longueur variable pour transmettre les gagnants des courses:

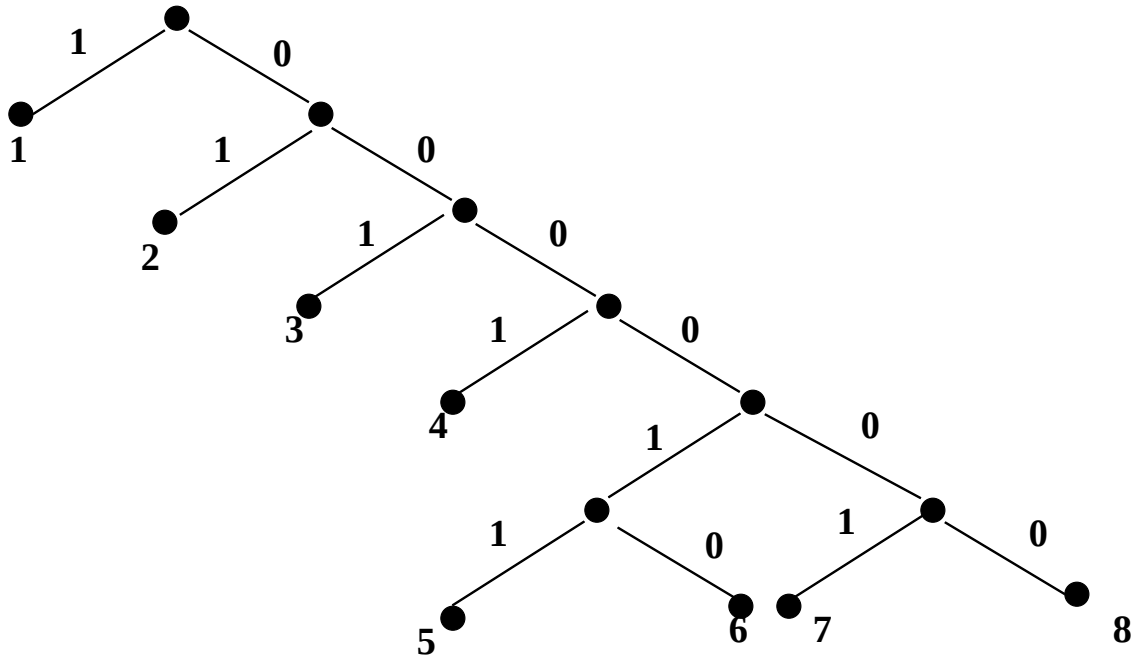


$$\frac{1}{2} \times 2 + \frac{1}{4} \times 2 + \frac{1}{8} \times 3 + \frac{1}{16} \times 4 + 4 \times \frac{1}{64} \times 4 = 2.1875$$

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

Entropie chevaline: un exemple de codage optimal

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

Pour une source sans mémoire (S) caractérisée par k probabilités indépendantes d'émettre k symboles s_k, on a:

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

Appliqué au langage, si on considère un message comme une instance d'une variable aléatoire W alors:

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

Ce que l'on peut reformuler par (où $p(w_1^n)$ signifie $p(W_1^n = w_1^n)$):

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

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 $\mathcal{T} = \{s_1, \ldots, s_n\}$, un corpus de n phrases, avec $s_i = \{w_1^i \ldots w_{n_{s_i}}^i\}$ une phrase de n_{s_i} mots. Alors:

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

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 de mots auquel doit faire face un modèle 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%

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

Note: 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 lissage Good/Turing (1953)

Idée: Pour tout n-gram 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-grams apparus r fois dans $\mathcal{T}$.

Ainsi pour tout n-gram α on a:

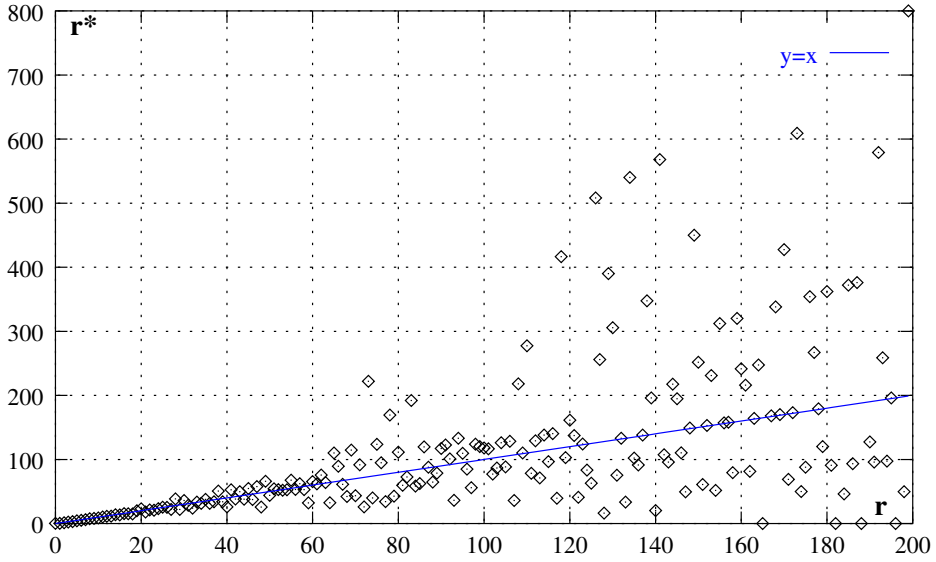
$$p_{GT}(\alpha) = \frac{r^*}{N'} \quad \text{avec } N' = \sum_{r=0}^{\infty} n_r r^*$$

Note: N' est bien le compte original (N) observé dans $\mathcal{T}$, 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-grams (différents) de fréquence r, lorsque r est grand, est faible (n_r est faible). Les estimées de fréquence (r^*) ont tendance dans ce cas à être bruitées. Lorsqu'un n-gram est fréquent, l'estimateur MLE est raisonnable.

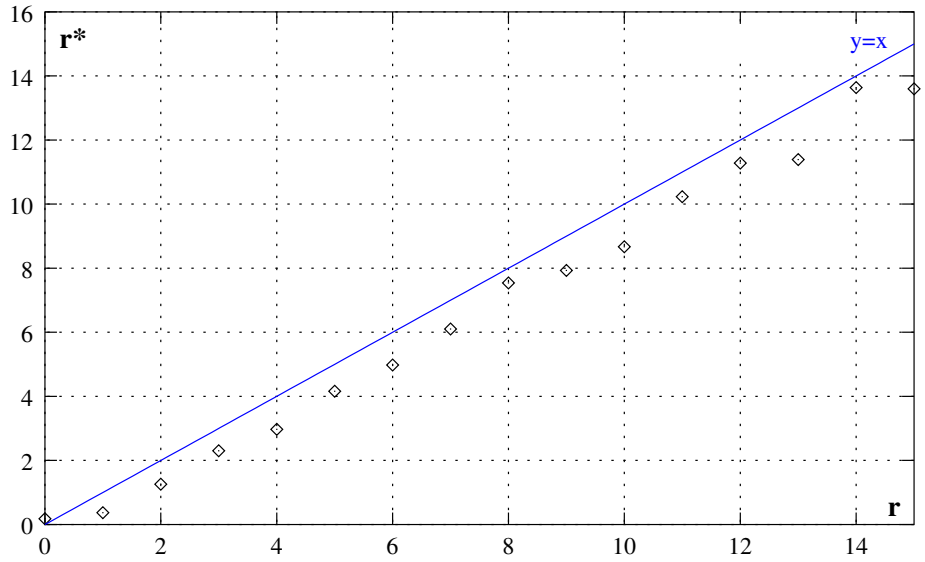


$$Texte{=}Austen,\ |V| = 14274,\ N = 620969,\ N1 = 68\%\ N+$$

Le lissage Good/Turing (1953)

En particulier, le n-gram le plus fréquent (soit r sa fréquence) aurait une estimée nulle par la formule GT, car n_{r+1} est nul.

$\hookrightarrow$ On ne peut pas appliquer GT si l'un des comptes n_r est nul. Ou alors il faut lisser ...



$\hookrightarrow$ on applique Good/Turing sur les n-grams dont la fréquence est petite.

Le lissage Good/Turing (1953)

r	n_r	r^*	r	n_r	r^*
0	203747076	.18	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-gram non vu est $p_0 = \frac{n_1}{N \times n_0}$ où n_0 est le nombre de n-gram non vus et N le nombre total de n-grams dans le corpus.

Ici $p_0 = 1.05 \times 10^{-9}$, et la masse totale de probabilité associée à des bigrammes non vus est: 0.21

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 $\mathcal{T}$ tel que: $|\text{journal du}| = |\text{journal de}| = |\text{journal humidifiant}| = 0$

  (par exemple parce que $journal$ n'est pas dans $\mathcal{T}$).

- Selon GT (et également add-one), ces bigrammes vont recevoir la même probabilité. Intuitivement, le troisième devrait être moins fréquent.

$\hookrightarrow$ 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)

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

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

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)

Reste à spécifier comment on calcule $\hat{p}$. On applique un discount $d_r \approx \frac{r^*}{r}$ pour les comptes faibles: $d_r = 1$ pour $\forall r > k$, Katz suggère de prendre 5 pour valeur de k.

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.

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

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

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

Jelinek-Mercer. Chercher les λ: première approche

Problème: estimation des λ, les modèles étant connus.

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

$\hookrightarrow$ 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. Chercher les λ: première approche

- 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. Chercher les λ: seconde approche

Partitionner l'espace des historiques

- Jelinek and Lafferty [1991] suggèrent 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 mots différents qui peuvent suivre un historique donné.

  $\hookrightarrow$ 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. Chercher les λ: estimation

Il existe un algorithme dérivé de Baum-Welch (1972) qui permet de le faire automatiquement. C'est un cas particulier de l'algorithme EM (Expectation, Maximization).

Important: Il faut prendre un autre corpus que $\mathcal{T}$ pour estimer les λ car sinon l'algorithme va trouver la réponse optimale qui est de donner le poids maximal au modèle le meilleur. $\Longrightarrow$ un corpus **held-out**.

Côté pratique: $|\text{held-out}| \sim 10\%$ à 25% de la taille de $\mathcal{T}$.

Si on manque de corpus ? deleted interpolation ou **cross-validation**:

On divise $\mathcal{T}$ en deux parties (ex: 50%/50%). 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 $\mathcal{T}$. Les estimées finales pondèrent les estimées partielles.

Quelques mots sur les held-out corpus

À l'extrême, Ney et al. (1997) ont proposé l'idée du **leaving-one-out**:

$\mathcal{T}$ est divisé en deux corpus: l'un de $N-1$ tokens, l'autre de 1 token (le held-out). Le gros corpus sert à l'estimée des modèles, le held-out sert à l'estimée des λ. On tourne N fois. On est moins soumis aux problèmes de sous représentation des données, mais au sacrifice des temps d'entraînement.

Même problème lorsque l'on teste la qualité d'un modèle. On utilise normalement un corpus held-out pour les tests. Un test final sera produit sur un corpus normalement non connu au moment de la réalisation du modèle.

Voir page 206-211 de Manning and Schütze [1999] pour plus de détails.

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, où 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 $\mathcal{T}$. C'est le MLE obtenu sur un corpus étendu avec un événement qui dit "je n'ai encore jamais été vu après h"

Pour le comprendre: Imaginez que pour chaque historique on passe de gauche à droite sur le corpus $\mathcal{T}$ 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: C'est une méthode de lissage qui avait été proposée initialement pour la compression de textes.

Lissage Absolute-Discounting Ney et al. [1994]

Idée: 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 garder une vrai distribution de probabilité, alors on doit avoir: $\sum_{w_i} p_{abs}(w_i|w_{i-n+1}^{i-1}) = 1$, 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]

Une extension du lissage par absolute discounting, où le modèle d'ordre inférieur est estimé de manière 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:

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

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 ?

Avant de commencer à coder: lire Chen and Goodman [1996] et Goodman [2001]. Ces papiers proposent des tests assez systématiques des différentes techniques (et d'autres) exposées ici.

On vous dira de manière quantifiée que le lissage Kneser & Ney semble être parmi les meilleures techniques de lissage; que Katz est également une assez bonne méthode qui marche d'autant mieux que les comptes initiaux sont grands. Jelinek & Mercer semble également être une valeur sure, plus adaptée aux comptes plus faibles.

Ce qu'il faut bien comprendre, c'est que 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. On sait par ex. (d'après des estimations faites à IBM) qu'un modèle bigramme sature au delà de quelques centaines de millions de mots. On estime qu'il en sera de même des trigrammes au delà de quelques billions de mots. Exemple reporté dans 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):

Exemple reporté dans 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.

D'autres approches à l'estimation des ML: A) Augmenter n

Problème: augmenter $n \Longrightarrow$ augmenter le problème de sous-représentation des données.

Mais pour les cas — assez rares — où un n-gram (avec $n = 5$ par exemple) a été vu souvent dans $\mathcal{T}$, pourquoi ne pas l'utiliser ?

Dans ce cas, la technique de lissage prend de l'importance

- Katz semble par exemple peu adaptée (meilleure pour les gros comptes).
- Kneser & Ney semble une méthode adaptée dont les performances ne chutent pas lorsque n augmente.

Note: si $|\mathcal{T}|$ n'est pas très grand, alors un bigramme et un trigramme ont à peu près la même performance (le trigramme demande cependant beaucoup plus de ressources).

B) le skipping

Idée: Éliminer du contexte conditionnant des éléments: plus n est grand, et plus on est susceptible de rencontrer des n-grammes proches mais non semblables.

Exemple: Si $|$il parle avec JEAN de lui$| > 0$ mais $|$il parle avec PIERRE de lui$| = 0$, alors on peut tenter d'estimer: $p(\text{lui}|\text{il parle avec} \text{ — } \text{de})$

En pratique on combine ensemble différents modèles skippés avec un modèle non skippé $\Longrightarrow$ on augmente la complexité du modèle.

Note: C'est une technique qui peut servir à créer également des modèles n-gram (n grand) du pauvre:

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

B) le skipping

En pratique, 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:

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

C) Clustering (regroupement)

Séduisant par son double but (en fait ils sont liés):

- Réduire le problème de sparse-data
- Introduire de l'information linguistique

Exemple: on observe dans $\mathcal{T}$ 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 n'est associé qu'à une seule classe, alors on parle de clustering dur (hard clustering); sinon on parle de clustering mou (soft clustering).

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

Comment utiliser le clustering ?

Plusieurs possibilités; entre autres:

$$\begin{aligned} p(w_3|w_1w_2) &\approx p(w_3|C_3)p(C_3|w_1w_2) \\ &\approx p(w_3|C_3)p(C_3|w_1C_2) \\ &\approx p(w_3|C_3)p(C_3|C_1C_2) \\ &\approx p(w_3|C_1C_2) \end{aligned}$$

Note: 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).

D) Modèles adaptatifs

La langue n'est pas figée dans le temps $\Longrightarrow$ un modèle statique n'est pas approprié.

A) Adaptation inter-domaine: 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-gram $p_{cache}(w|h)$ qui est combiné au modèle statique.

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

Améliore les performances, surtout si le corpus d'entraînement est petit. Goodman [2001] observe pour des corpus assez grands, une réduction d'entropie de 13% avec des modèles caches bigrammes et trigrammes.

D) Modèles adaptatifs: Problème avec le caching

Les tests sont menés sur un historique (grandissant) **connu**. Dans une véritable application (ex: reconnaissance de la parole) , l'historique est bruité (contient des erreurs de reconnaissance).

$\longrightarrow$ le système peut bloquer sur des erreurs[3]:

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
...	...

[3] Exemple factice bien que probable.

D) Modèles adaptatifs

B) Adaptation intra-domaine

Textes d'une même source, mais assez hétérogènes (ex: dépêches AFP).

Idée: découper $\mathcal{T}$ 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.

Cas typique: on a un gros corpus $\mathcal{T}_h$ (hors-data) pour entraîner un modèle "général", puis un plus petit corpus $\mathcal{T}_a$ plus représentatif de l'application. En fait, la combinaison des deux modèles donne habituellement des performances décevantes en comparaison du modèle entraîné directement sur $\mathcal{T}_a$!

D) Modèles adaptatifs

Exemple pris de Rosenfeld [2000]:

$\mathcal{T}_a =$ Switchboard (25 millions de mots);
$\mathcal{T}_g = \{$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-data (généraux).

$\Longrightarrow$ Le problème n'est pas simple. Importance des techniques de combinaison.

Note: l'adaptabilité intra-domaine est assez peu étudiée.

D) Modèles adaptatifs

Une alternative à l'adaptation intra-domaine 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 $a\ priori$ 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] reporte des améliorations si on prend S grand (64 ou 128).

Note: Entraîne une division du corpus d'entraînement.

E) Maximum Entropy Modeling

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

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

où $\mathcal{Z}(h)$ est un terme de normalisation ($\mathcal{Z}(h) = \sum_w \exp \sum_i \lambda_i f_i(h, w)$); 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) .

E) Maximum Entropy Modelling

Exemple de trait:

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

Traduction: 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.

E) Maximum Entropy Modeling

Rosenfeld reporte 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 système gagnant, et pas dans le baseline ayant servi à la comparaison $\longrightarrow$ résultats trop optimistes.

Goodman [2001] reporte qu'avec le même type d'information, ME ne fonctionne pas mieux que la combinaison "classique" de modèles.

La modélisation par maximisation d'entropie est très à la mode et donne des résultats dans de nombreuses applications (grammaires probabilistes, modèles de langue, traduction probabiliste). Lire Berger et al. [1996] pour une bonne exposition à cette méthode d'apprentissage.

F) Sentence Maximum Entropy models

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:

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

$\mathcal{Z}$ est cette fois-ci constant.

$p_0(s)$ est un modèle de départ (par exemple votre meilleur n-gram)

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

Problème: Les algorithmes d'entraînement sont lourds et complexes. On déplace finalement le problème sur le problème non trivial de la découverte des traits $\longrightarrow$ pas certain que cela marche mieux qu'une grammaire probabiliste ("classique") combinée à un n-gram.

G) Modèles connexionistes Bengio et al. [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 $\longrightarrow$ 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.

Note: une approches très prometteuse, mais des temps d'entraînement (moins grave) et d'utilisation (plus gênant) trop importants.

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

Références

Yoshua Bengio, Rejean Ducharme, and Pascal Vincent. A neural probabilistic language model. In *Advances in Neural Information Processing Systems*, 2001.

Adam Berger, Stpehn Della Pietra, and Vincent Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

Stanley F. Chen. Building probabilistic models for natural language, 1996. URL `citeseer.nj.nec.com/chen96building.html`.

Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. In Arivind Joshi and Martha Palmer, editors, *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, pages 310–318, San Francisco, 1996. Morgan Kaufmann Publishers. URL `citeseer.nj.nec.com/chen96empirical.html`.

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

W. Gale and K. W. Church. Poor estimates are worse than none. In *proceedings of DARPA Speech and Natural Language Workshop*, pages 283–287, Hidden Valley, PA, june 1990.

Joshua Goodman. A bit of progress in language modeling. *Computer Speech and Language*, pages 403–434, oct 2001.

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

Fred Jelinek and John D. Lafferty. Computation of the probability of initial substring generation by stochastic context-free grammars. *Computational Linguistics*, 17:315–324, 1991.

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

Reinhard Kneser and Hermann Ney. Improved backing-off for m-gram language modeling. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 1, pages 181–184, 1995.

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

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

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

R. Rosenfeld. Two decades of statistical language modeling: Where do we go from here, 2000. URL `citeseer.nj.nec.com/rosenfeld00two.html`.

Andreas Stolcke. Srilm - an extensible language modeling toolkit. In *Intl. Conf. Spoken Language Processing*, Denver, Colorado, 2002.