Introduction aux modèles de langue

felipe@iro.umontreal.ca

RALI
Dept. Informatique et Recherche Opérationnelle
Université de **Montréal**

V1.0 (V0.8 pour les modèles neuronaux)

Last compiled: 23 septembre 2019

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]***
- introduction aux modèles de langue neuronaux

rali

Plan

Quelques Applications

Estimateurs classiques d'un modèle n-gramme

Éléments de la théorie de l'information

Méthodes classiques de lissage

Patchwork d'approches dérivées

Modèles de langue neuronaux

Bilan

Boites à outils pour l'entraînement

rali

Plan

rali

Modèles de langue

- 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

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

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

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

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

Exemples d'applications

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

- 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 $\mathcal{L}$ l'ensemble des langues, l_i une de ces langues

But : Découvrir L_T, la langue de T

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

Identification de la langue

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/

rali

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

rali

Exemples d'applications

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

rali

Exemples d'applications

- 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) :

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

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

Plan

rali

Modèle n-gramme

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

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

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

rali

Modèle n-gramme

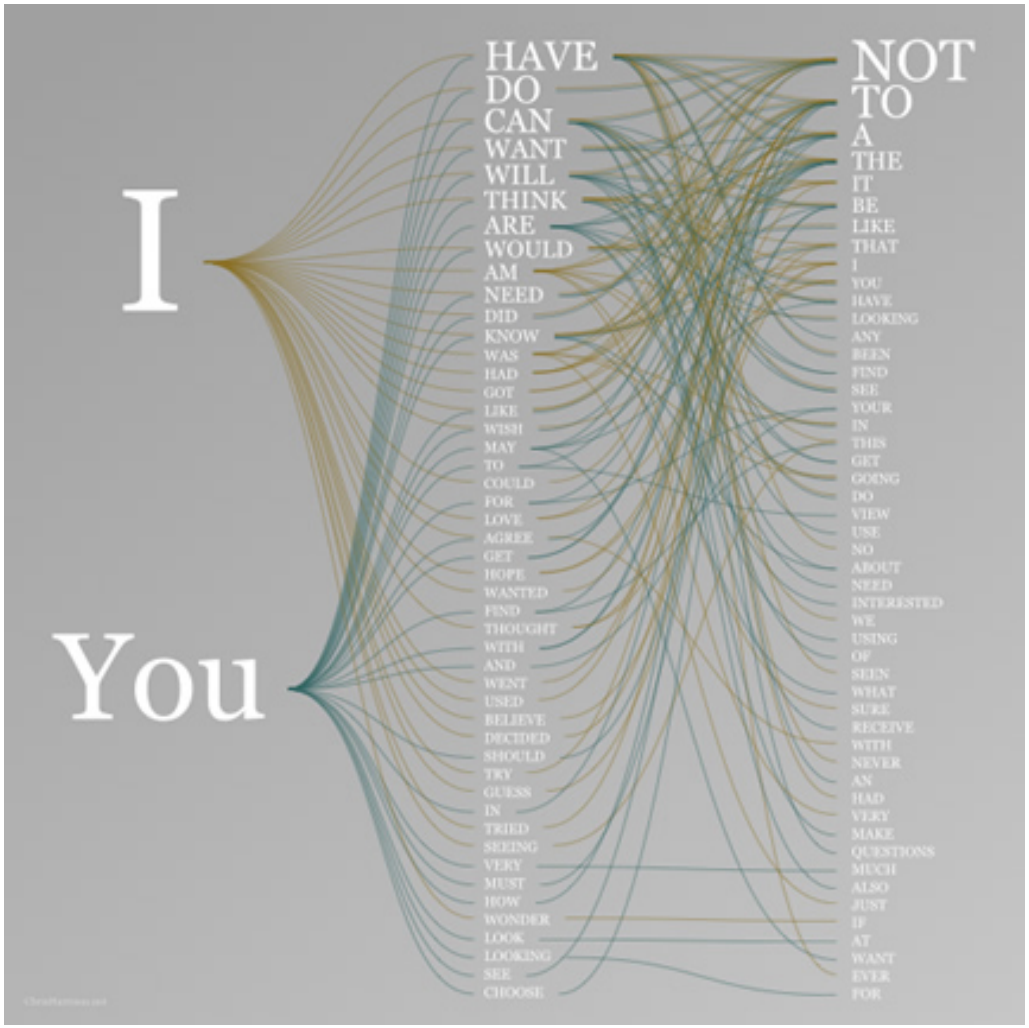
$$p(s) = \prod_{i=1}^{N} p(w_i | w_{i-2} w_{i-1})$$

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

Un modèle trigramme, visuellement

http://www.chrisharrison.net/index.php/Visualizations

Un modèle trigramme, visuellement



http://www.chrisharrison.net/index.php/Visualizations

Estimateur à maximum de vraisemblance

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

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

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

- 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 :

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

rali

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 $|\text{BOS Virginie}| = 0$

rali

Estimateur à maximum de vraisemblance

- 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

- 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

Beaucoup 30M. de mots ?

- Google : http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html

  "We processed 1,011,582,453,213 words of running text and are publishing the counts for all 1,146,580,664 five-word sequences that appear at least 40 times. There are 13,653,070 unique words, after discarding words that appear less than 200 times."

Google n-gram Viewer


English

Version 20120701
total_counts

1-grams 0 1 2 3 4 5 6 7 8 9 a b c d e f g h i j k l m n o other p pos punctuation q r s t u v w x y z

2-grams 0 1 2 3 4 5 6 7 8 9 _ADJ_ _ADP_ _ADV_ _CONJ_ _DET_ _NOUN_ _NUM_ _PRON_ _PRT_ _VERB_ a_ aa ab ac ad ae af ag ah ai aj ak al am an ao ap aq ar as at au av aw ax ay
br bs bt bu bv bw bx by bz c_ ca cb cc cd ce cf cg ch ci cj ck cl cm cn co cp cq cr cs ct cu cv cw cx cy cz d_ da db dc dd de df dg dh di dj dk dl dm dn do dp dq dr ds dt du dv dw dx dy dz e_ e
eu ev ew ex ey ez f_ fa fb fc fd fe ff fg fh fi fj fk fl fm fn fo fp fq fr fs ft fu fv fw fx fy fz g_ ga gb gc gd ge gf gg gh gi gj gk gl gm gn go gp gq gr gs gt gu gv gw gx gy gz h_ ha hb hc hd he hf hg hh
ia ib ic id ie if ig ih ii ij ik il im in io ip iq ir is it iu iv iw ix iy iz j_ ja jb jc jd je jf jg jh ji jj jk jl jm jn jo jp jq jr js jt ju jv jw jx jy jz k_ ka kb kc kd ke kf kg kh ki kj kk kl km kn ko kp kq kr ks kt ku kv kw kx
lw lx ly lz m_ ma mb mc md me mf mg mh mi mj mk ml mm mn mo mp mq mr ms mt mu mv mw mx my mz n_ na nb nc nd ne nf ng nh ni nj nk nl nm nn no np nq nr ns nt nu nv nw nx ny nz o
os ot other ou ov ow ox oy oz p_ pa pb pc pd pe pf pg ph pi pj pk pl pm pn po pp pq pr ps pt pu punctuation pv pw px py pz q_ qa qb qc qd qe qf qg qh qi qj qk ql qm qn qo qp qq qr qs qt qu q
rq rr rs rt ru rv rw rx ry rz s_ sa sb sc sd se sf sg sh si sj sk sl sm sn so sp sq sr ss st su sv sw sx sy sz t_ ta tb tc td te tf tg th ti tj tk tl tm tn to tp tq tr ts tt tu tv tw tx ty tz u_ ua ub uc ud ue uf ug
v_ va vb vc vd ve vf vg vh vi vj vk vl vm vn vo vp vq vr vs vt vu vv vw vx vy vz w_ wa wb wc wd we wf wg wh wi wj wk wl wm wn wo wp wq wr ws wt wu wv ww wx wy wz x_ xa xb xc xd xe xf x
y_ ya yb yc yd ye yf yg yh yi yj yk yl ym yn yo yp yq yr ys yt yu yv yw yx yy yz z_ za zb zc zd ze zf zg zh zi zj zk zl zm zn zo zp zq zr zs zt zu zv zw zx zy zz

3-grams 0 1 2 3 4 5 6 7 8 9 _ADJ_ _ADP_ _ADV_ _CONJ_ _DET_ _NOUN_ _NUM_ _PRON_ _PRT_ _VERB_ a_ aa ab ac ad ae af ag ah ai aj ak al am an ao ap aq ar as at au av aw ax ay
br bs bt bu bv bw bx by bz c_ ca cb cc cd ce cf cg ch ci cj ck cl cm cn co cp cq cr cs ct cu cv cw cx cy cz d_ da db dc dd de df dg dh di dj dk dl dm dn do dp dq dr ds dt du dv dw dx dy dz e_ e
eu ev ew ex ey ez f_ fa fb fc fd fe ff fg fh fi fj fk fl fm fn fo fp fq fr fs ft fu fv fw fx fy fz g_ ga gb gc gd ge gf gg gh gi gj gk gl gm gn go gp gq gr gs gt gu gv gw gx gy gz h_ ha hb hc hd he hf hg hh
ia ib ic id ie if ig ih ii ij ik il im in io ip iq ir is it iu iv iw ix iy iz j_ ja jb jc jd je jf jg jh ji jj jk jl jm jn jo jp jq jr js jt ju jv jw jx jy jz k_ ka kb kc kd ke kf kg kh ki kj kk kl km kn ko kp kq kr ks kt ku kv kw kx
lw lx ly lz m_ ma mb mc md me mf mg mh mi mj mk ml mm mn mo mp mq mr ms mt mu mv mw mx my mz n_ na nb nc nd ne nf ng nh ni nj nk nl nm nn no np nq nr ns nt nu nv nw nx ny nz o
os ot other ou ov ow ox oy oz p_ pa pb pc pd pe pf pg ph pi pj pk pl pm pn po pp pq pr ps pt pu punctuation pv pw px py pz q_ qa qb qc qd qe qf qg qh qi qj qk ql qm qn qo qp qq qr qs qt qu q
rq rr rs rt ru rv rw rx ry rz s_ sa sb sc sd se sf sg sh si sj sk sl sm sn so sp sq sr ss st su sv sw sx sy sz t_ ta tb tc td te tf tg th ti tj tk tl tm tn to tp tq tr ts tt tu tv tw tx ty tz u_ ua ub uc ud ue uf ug
v_ va vb vc vd ve vf vg vh vi vj vk vl vm vn vo vp vq vr vs vt vu vv vw vx vy vz w_ wa wb wc wd we wf wg wh wi wj wk wl wm wn wo wp wq wr ws wt wu wv ww wx wy wz x_ xa xb xc xd xe xf x
y_ ya yb yc yd ye yf yg yh yi yj yk yl ym yn yo yp yq yr ys yt yu yv yw yx yy yz z_ za zb zc zd ze zf zg zh zi zj zk zl zm zn zo zp zq zr zs zt zu zv zw zx zy zz

4-grams 0 1 2 3 4 5 6 7 8 9 _ADJ_ _ADP_ _ADV_ _CONJ_ _DET_ _NOUN_ _NUM_ _PRON_ _PRT_ _VERB_ a_ aa ab ac ad ae af ag ah ai aj ak al am an ao ap aq ar as at au av aw ax ay
br bs bt bu bv bw bx by bz c_ ca cb cc cd ce cf cg ch ci cj ck cl cm cn co cp cq cr cs ct cu cv cw cx cy cz d_ da db dc dd de df dg dh di dj dk dl dm dn do dp dq dr ds dt du dv dw dx dy dz e_ e
eu ev ew ex ey ez f_ fa fb fc fd fe ff fg fh fi fj fk fl fm fn fo fp fq fr fs ft fu fv fw fx fy fz g_ ga gb gc gd ge gf gg gh gi gj gk gl gm gn go gp gq gr gs gt gu gv gw gx gy gz h_ ha hb hc hd he hf hg hh
ia ib ic id ie if ig ih ii ij ik il im in io ip iq ir is it iu iv iw ix iy iz j_ ja jb jc jd je jf jg jh ji jj jk jl jm jn jo jp jq jr js jt ju jv jw jx jy jz k_ ka kb kc kd ke kf kg kh ki kj kk kl km kn ko kp kq kr ks kt ku kv kw kx
lw lx ly lz m_ ma mb mc md me mf mg mh mi mj mk ml mm mn mo mp mq mr ms mt mu mv mw mx my mz n_ na nb nc nd ne nf ng nh ni nj nk nl nm nn no np nq nr ns nt nu nv nw nx ny nz o
os ot other ou ov ow ox oy oz p_ pa pb pc pd pe pf pg ph pi pj pk pl pm pn po pp pq pr ps pt pu punctuation pv pw px py pz q_ qa qb qc qd qe qf qg qh qi qj qk ql qm qn qo qp qq qr qs qt qu q
rq rr rs rt ru rv rw rx ry rz s_ sa sb sc sd se sf sg sh si sj sk sl sm sn so sp sq sr ss st su sv sw sx sy sz t_ ta tb tc td te tf tg th ti tj tk tl tm tn to tp tq tr ts tt tu tv tw tx ty tz u_ ua ub uc ud ue uf ug
v_ va vb vc vd ve vf vg vh vi vj vk vl vm vn vo vp vq vr vs vt vu vv vw vx vy vz w_ wa wb wc wd we wf wg wh wi wj wk wl wm wn wo wp wq wr ws wt wu wv ww wx wy wz x_ xa xb xc xd xe xf x
y_ ya yb yc yd ye yf yg yh yi yj yk yl ym yn yo yp yq yr ys yt yu yv yw yx yy yz z_ za zb zc zd ze zf zg zh zi zj zk zl zm zn zo zp zq zr zs zt zu zv zw zx zy zz

5-grams 0 1 2 3 4 5 6 7 8 9 _ADJ_ _ADP_ _ADV_ _CONJ_ _DET_ _NOUN_ _NUM_ _PRON_ _PRT_ _VERB_ a_ aa ab ac ad ae af ag ah ai aj ak al am an ao ap aq ar as at au av aw ax ay
br bs bt bu bv bw bx by bz c_ ca cb cc cd ce cf cg ch ci cj ck cl cm cn co cp cq cr cs ct cu cv cw cx cy cz d_ da db dc dd de df dg dh di dj dk dl dm dn do dp dq dr ds dt du dv dw dx dy dz e_ e
eu ev ew ex ey ez f_ fa fb fc fd fe ff fg fh fi fj fk fl fm fn fo fp fq fr fs ft fu fv fw fx fy fz g_ ga gb gc gd ge gf gg gh gi gj gk gl gm gn go gp gq gr gs gt gu gv gw gx gy gz h_ ha hb hc hd he hf hg hh
ia ib ic id ie if ig ih ii ij ik il im in io ip iq ir is it iu iv iw ix iy iz j_ ja jb jc jd je jf jg jh ji jj jk jl jm jn jo jp jq jr js jt ju jv jw jx jy jz k_ ka kb kc kd ke kf kg kh ki kj kk kl km kn ko kp kq kr ks kt ku kv kw kx
lw lx ly lz m_ ma mb mc md me mf mg mh mi mj mk ml mm mn mo mp mq mr ms mt mu mv mw mx my mz n_ na nb nc nd ne nf ng nh ni nj nk nl nm nn no np nq nr ns nt nu nv nw nx ny nz o
os ot other ou ov ow ox oy oz p_ pa pb pc pd pe pf pg ph pi pj pk pl pm pn po pp pq pr ps pt pu punctuation pv pw px py pz q_ qa qb qc qd qe qf qg qh qi qj ql qm qn qo qp qq qr qs qt qu qv q
rr rs rt ru rv rw rx ry rz s_ sa sb sc sd se sf sg sh si sj sk sl sm sn so sp sq sr ss st su sv sw sx sy sz t_ ta tb tc td te tf tg th ti tj tk tl tm tn to tp tq tr ts tt tu tv tw tx ty tz u_ ua ub uc ud ue uf ug u
va vb vc vd ve vf vg vh vi vj vk vl vm vn vo vp vq vr vs vt vu vv vw vx vy vz w_ wa wb wc wd we wf wg wh wi wj wk wl wm wn wo wp wq wr ws wt wu wv ww wx wy wz x_ xa xb xc xd xe xf xg x
ya yb yc yd ye yf yg yh yi yj yk yl ym yn yo yp yq yr ys yt yu yv yw yx yy yz z_ za zb zc zd ze zf zg zh zi zj zk zl zm zn zo zp zq zr zs zt zu zv zw zx zy zz

Google n-gram

```
% time wc -l
  googlebooks-eng-all-4gram-20120701-bu 420836500
  googlebooks-eng-all-4gram-20120701-bu
  10.110u 5.592s 2:28.48 10.5% 0+0k 5+0io 2pf+0w
```

- un extrait :

```
but_CONJ down_ADV with_ADP the_DET      1996    2
but_CONJ down_ADV with_ADP the_DET      1997    2
but_CONJ down_ADV with_ADP the_DET      1998    2
but_CONJ down_ADV with_ADP the_DET      1999    9
but_CONJ down_ADV with_ADP the_DET      2000    1
```

Estimation par maximum de vraisemblance

Soit $\mathcal{D} \equiv (x_1, \ldots, x_n) \sim p(x|\theta)$ 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 :

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

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

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

MLE = Maximum Likelihood Estimation

Estimation à maximum de vraisemblance

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

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

- 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 ($x_i \in \{0, 1\}$) de paramètre θ. On observe s événements positifs. Quelle est la valeur de θ ?

- s est la statistique suffisante. $p(\sum_i x_i = s)$ est donnée par une loi binomiale
- 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 $\mathcal{D}$ sont générés aléatoirement de manière indépendante par des jetés d'un dé à $|V|$ faces (V le vocabulaire). La prob. d'observer x_i, les comptes de chaque dé est régie par une loi multinomiale

- La vraisemblance est alors :

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

où x_i est le compte (observé) de chacun des mots 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

- Idée (simple qui marche mal en pratique) : prétendre que tout événement a été vu une fois de plus dans $\mathcal{D}$
- 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 $\mathcal{D}$, nous prétendons cependant que nous l'avons trouvé une fois.

Le add-one smoothing, Vincent et Virginie

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

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

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

rali

Le add-one smoothing, Vincent et Virginie

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

rali

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 $\mathcal{D}$:

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

- 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) :

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

On parle de maximum à postériori (estimateur MAP)

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

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

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 :

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

Lissage *add-one* et MAP

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

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

(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 :

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

rali

Lissage additif

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

- Si on pose :

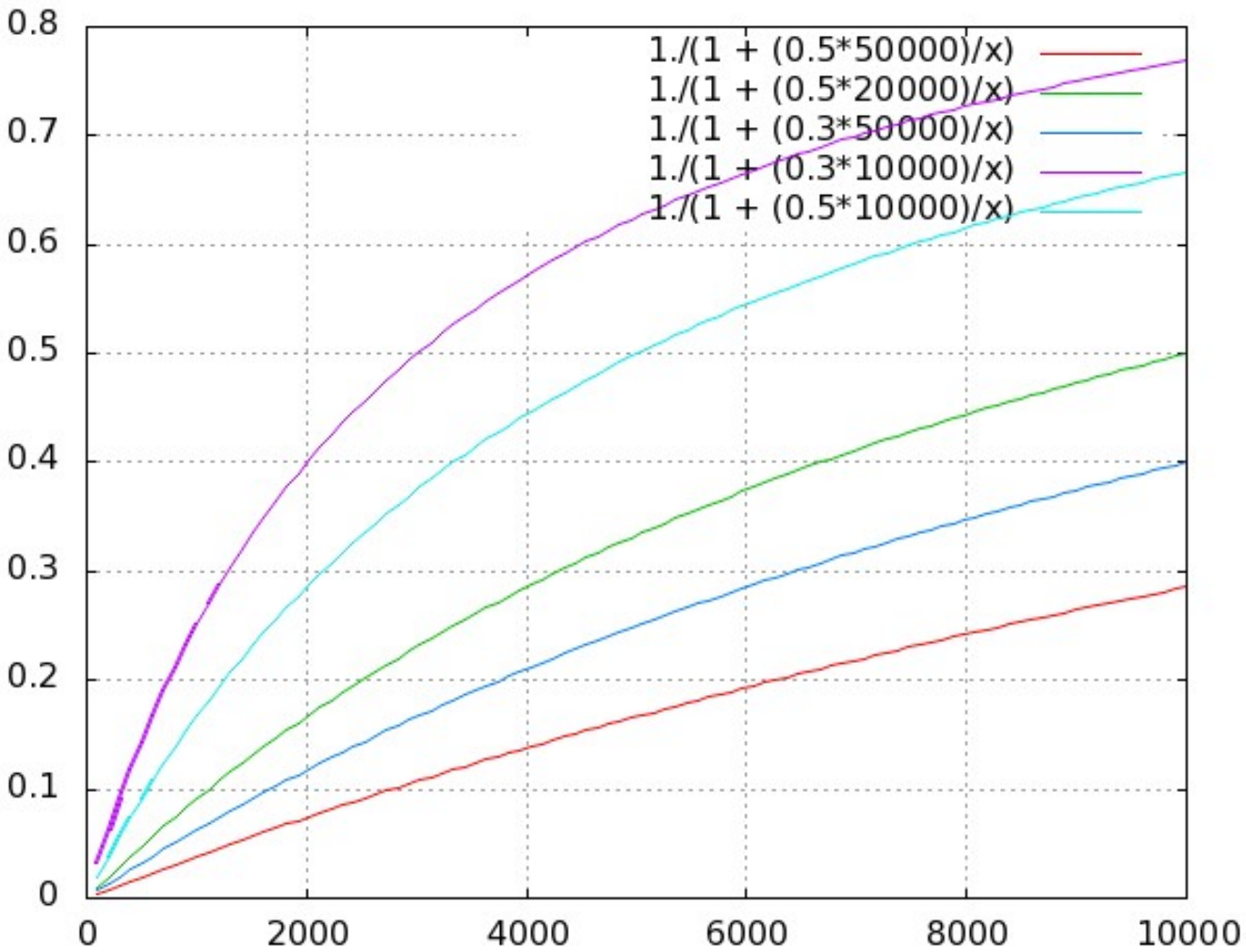
$$\mu_{|w_{i-1}|} = \frac{1}{1 + \delta|V|/|w_{i-1}|}$$

- 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]

Lissage additif



Plan

rali

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

rali

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

rali

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

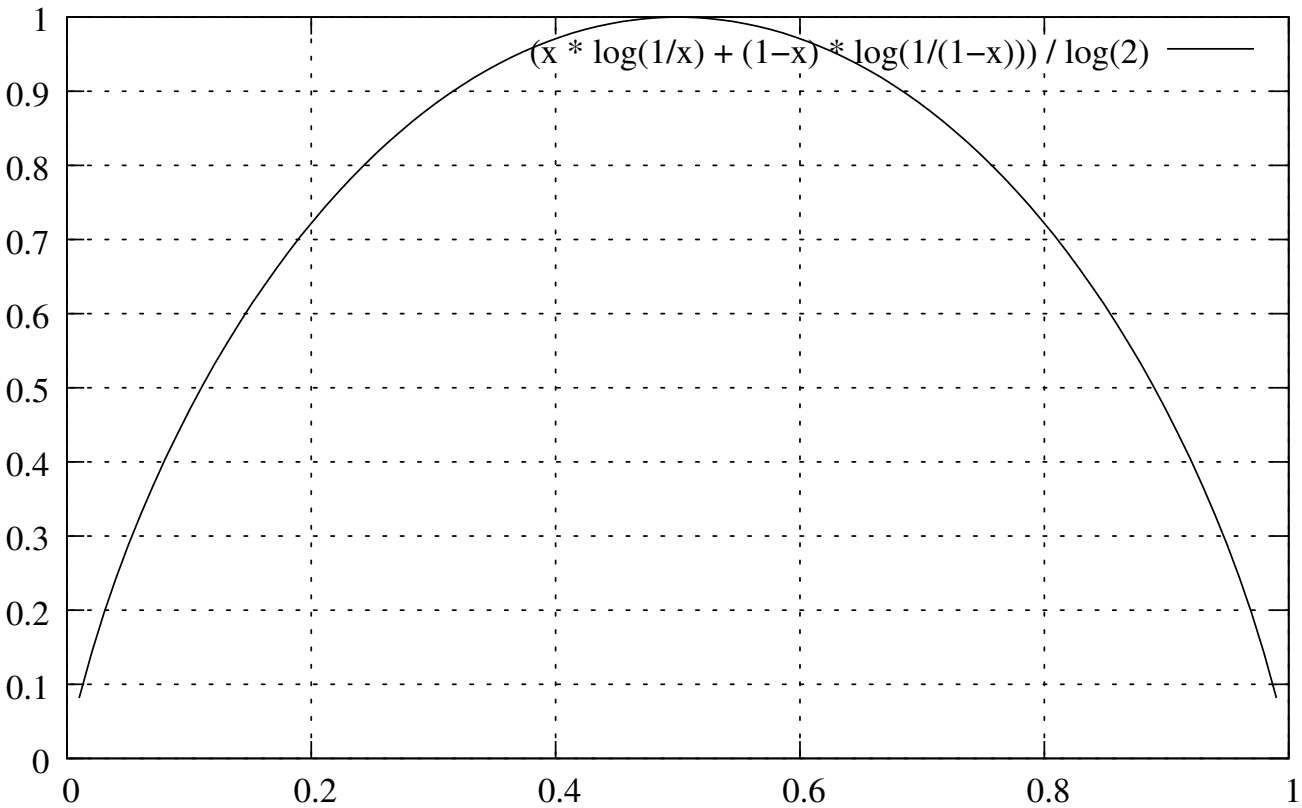
- $H(P) \geq 0$
- Pour toute distribution $q = q_1, q_2, \ldots, q_k$, sous le régime P :

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

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 :

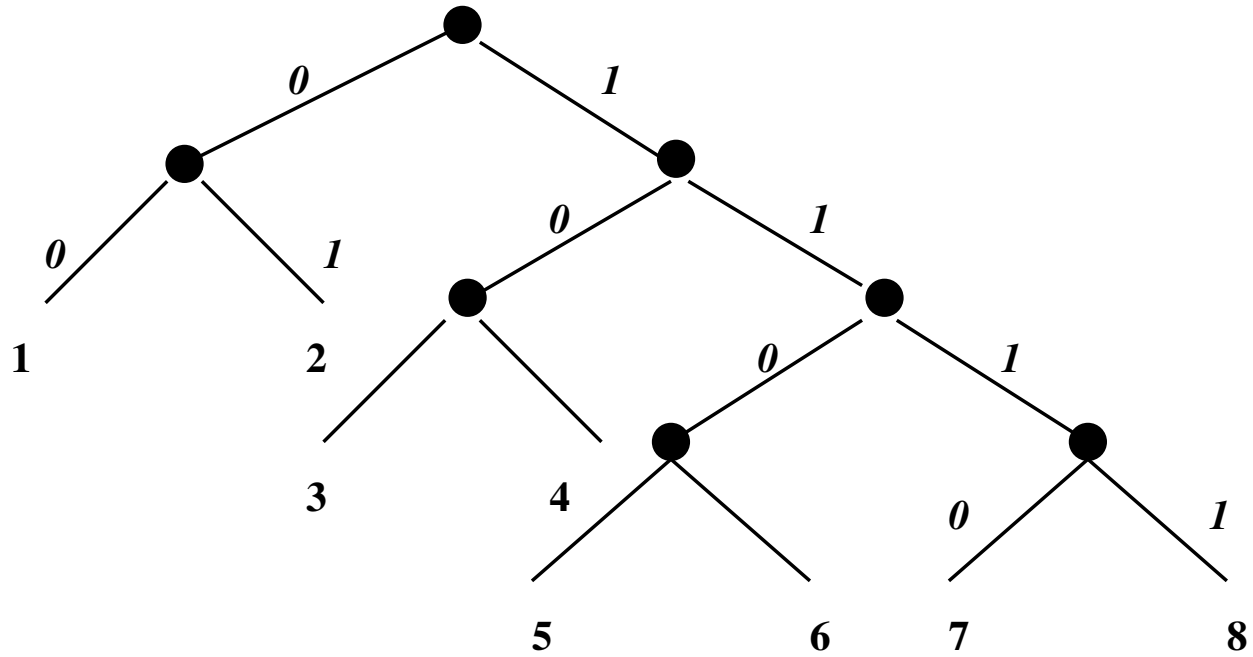
$$\begin{aligned} H(x) &= -\sum_{i=1}^{8} p(i) \log_2 p(i) \\ &= -\frac{1}{2}\log_2 \frac{1}{2} + \frac{1}{4}\log_2 \frac{1}{4} - \ldots - 4 \times \frac{1}{64}\log_2 \frac{1}{64} \\ &= 2 \text{ bits} \end{aligned}$$

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

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

Entropie chevaline

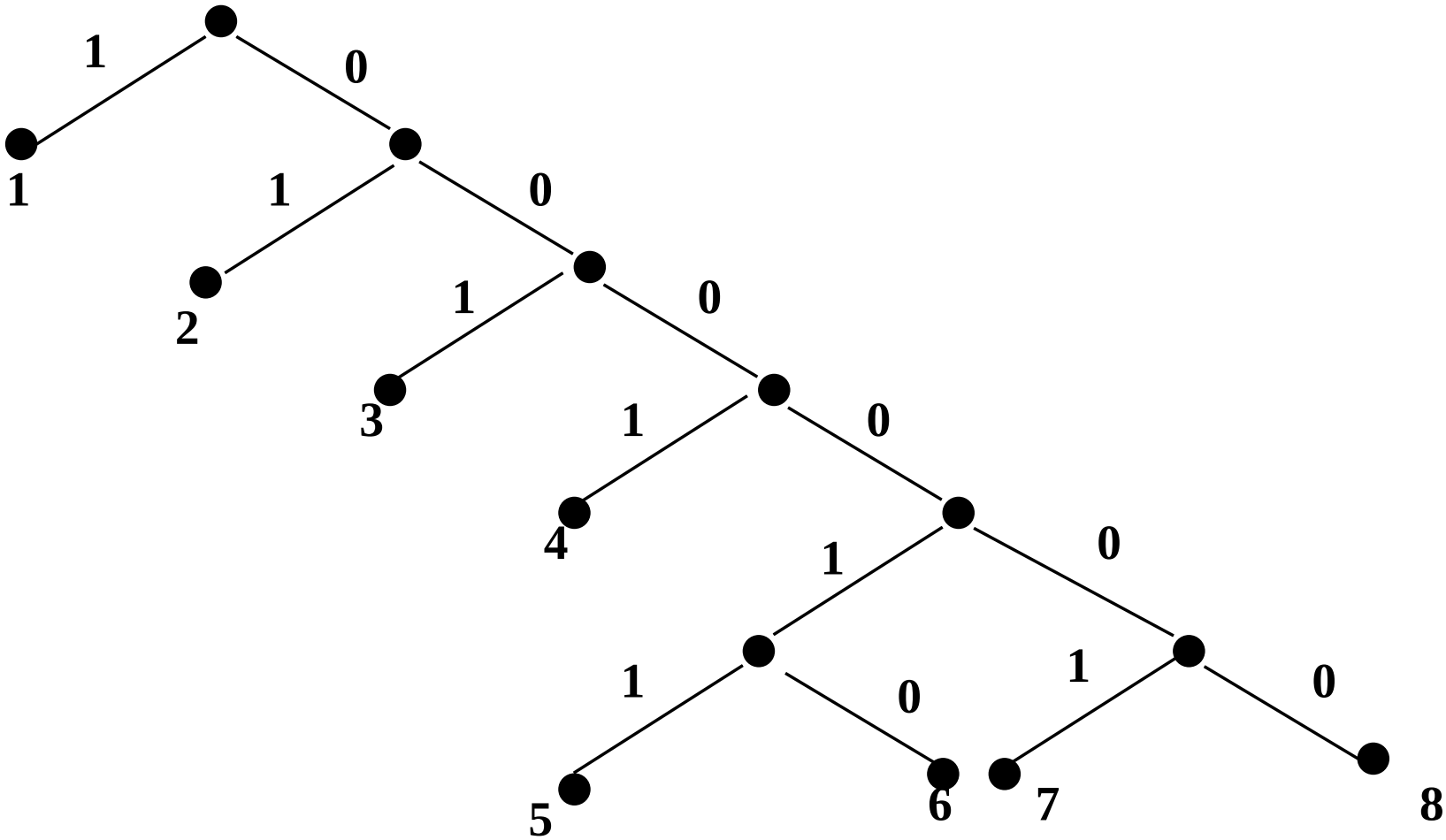
- Exemple (non optimal) de code à longueur variable pour transmettre les gagnants des courses :



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

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

Entropie chevaline



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

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

rali

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)

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

rali

Cas du trigramme

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

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

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 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]***...

rali

Mesure de la qualité d'un modèle

Soit le corpus de test : `choo choo train` (on verra bcp. plus loin d'où vient cet exemple) et les modèles unigramme suivants sur le vocabulaire $\{\text{choo}, \text{train}\}$:

$$p_1 : p_1(\text{choo}) = 0.1, p_1(\text{train}) = 0.9$$
$$p_2 : p_2(\text{choo}) = 0.49, p_2(\text{train}) = 0.51$$
$$p_3 : p_3(\text{choo}) = 0.6, p_3(\text{train}) = 0.4$$
$$p_4 : p_4(\text{choo}) = 0.9, p_3(\text{train}) = 0.1$$

Perplexité en test :

$$p_1 : 2^{-((log_2(0.1)+log_2(0.1)+log_2(0.9))/3)} \approx 4.8$$
$$p_2 : 2^{-((log_2(0.49)+log_2(0.49)+log_2(0.51))/3)} \approx 2.01$$
$$p_3 : 2^{-((log_2(0.6)+log_2(0.6)+log_2(0.5))/3)} \approx 3.5$$
$$p_4 : 2^{-((log_2(0.9)+log_2(0.9)+log_2(0.1))/3)} \approx 2.6$$

rali

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 de la bonne réponse pour la i-ème lettre

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).
  - modèle aveugle : $\log(27)$ = 4.75 bits /caractère
  - modèle unigramme : 4.03 bits /caractère
- ***[Brown et al., 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)

rali

Plan

rali

Le lissage Good/Turing (1953) [4]

- 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 $\mathcal{T}$.

- Ainsi pour tout n-gramme α on a :

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

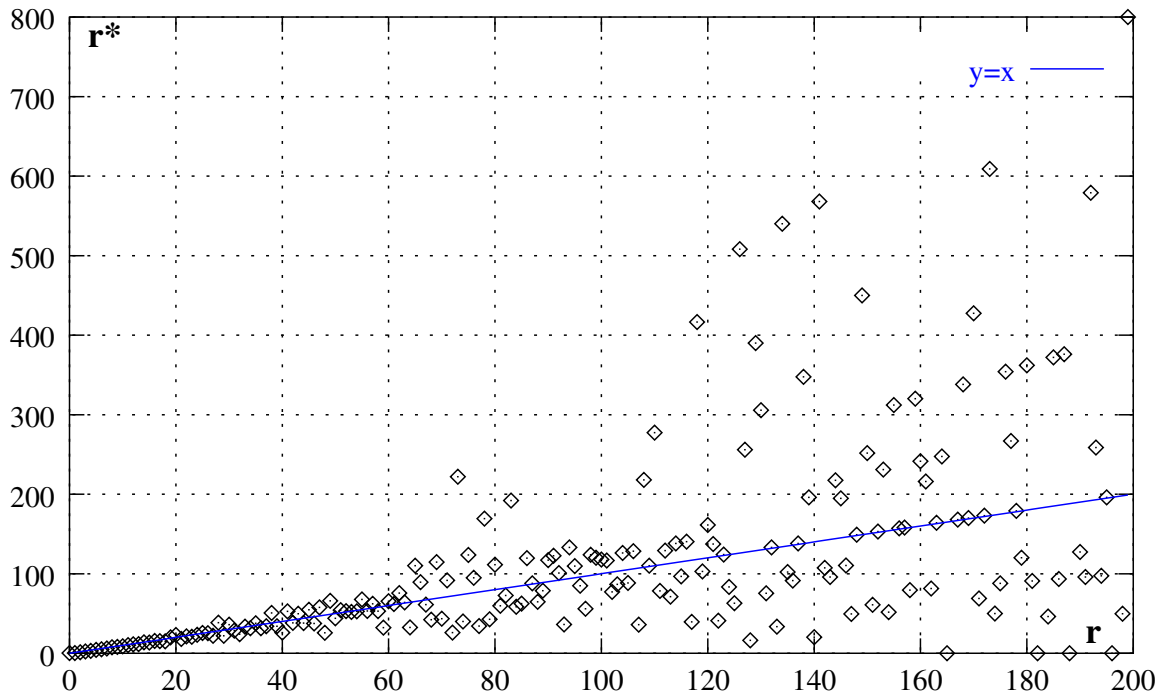
!! N' est bien le compte original (N) observé dans $\mathcal{T}$, car :

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

4. Inventé par Turing pendant la second guerre mondiale (Enigma)

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.

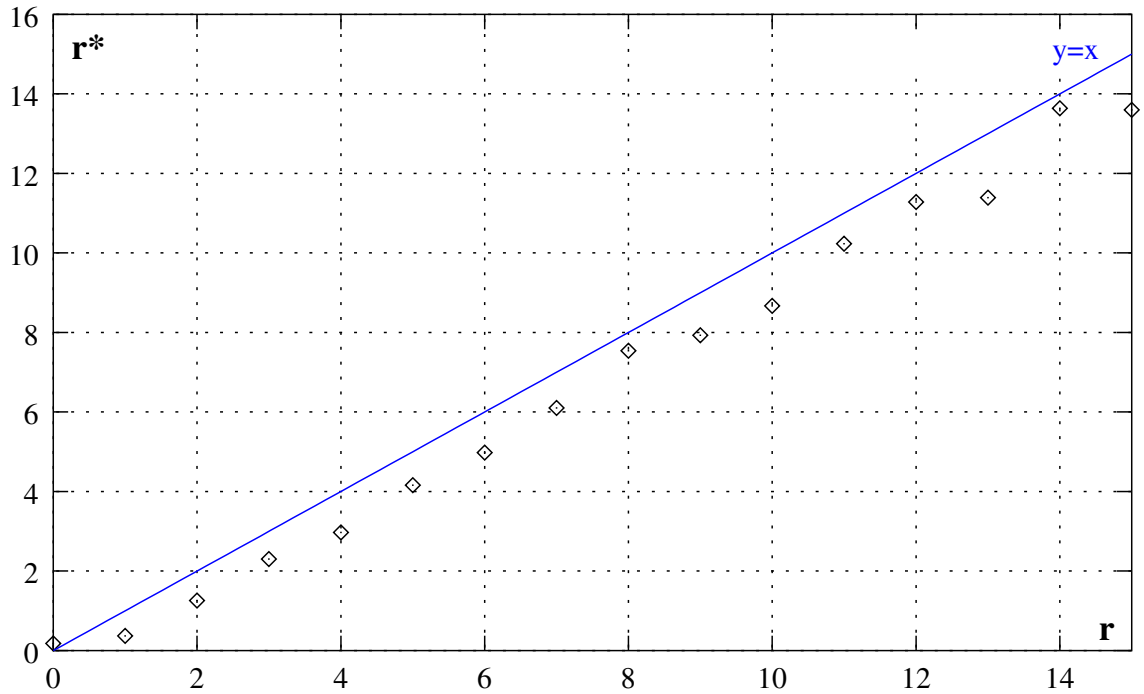


Texte=Austen, $|V| = 14274$, $N = 620969$ $N1 = 68\%$ $N+$

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 de Katz (1987) : modèle backoff (repli)

Idée : étend l'intuition de GT, mais en consultant un modèle d'ordre inférieur si l'unité n'a pas été vue en corpus.

- 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 probable.
- En consultant un modèle unigramme on peut éventuellement parvenir à rendre compte de cette intuition qui devrait donner `humidifiant` comme moins probable (si 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)

- 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 (on parle de modèle interpolé) :

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

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

Jelinek-Mercer

- **Intuitivement :** si un historique h est fréquent dans $\mathcal{T}$, 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.

rali

Jelinek-Mercer

- 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

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.

rali

Jelinek-Mercer

- 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 $\mathcal{T}$ pour estimer les λ. Sinon l'algorithme va converger vers une solution donnant tout le poids au modèle de plus grande capacité.
  - corpus de développement (*held-out*) — ex : 20% de $\mathcal{T}$.
  - validation croisée si $\mathcal{T}$ est trop petit.

    Ex : On divise $\mathcal{T}$ 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 $\mathcal{T}$. Les estimées sont des moyennes par pli (ici 2).

Lissage Absolute-Discounting *[Ney et al., 1994]*

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

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

- $\gamma_{w_{i-n+1}^{i-1}}$ est une fonction de D
- Ney et al. suggèrent d'utiliser : $D = \frac{n_1}{n_1+2n_2}$

Note : d'autres D peuvent être appliqués pour les modèles d'ordre inférieur.

Point d'orgue

[Kneser and Ney, 1995]

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

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

- avec :

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

- 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

- Dans un sous-corpus du Hansard de $N = 153,213$ tokens ($|V| = 7986$ types) 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
    $p(\text{président}|\text{le}) = \frac{738}{|\text{le}|=4425} (\approx 0.166)$

Lissage Kneser-Ney

- 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 : $\frac{|\{\text{présidente}\}|}{16057} (\approx 6.22 \times 10^{-5})$,

(Extreme) Executive summary

Deux familles classiques de lissage

- modèle de repli (*backoff*) Katz

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

où :

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

(Extreme) Executive summary

Deux familles classiques de lissage

- modèle de repli (*backoff*) Katz

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

où :

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

- modèle interpolé Jelinek-Mercer

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

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 sûre adaptée aux comptes plus faibles.
- La réponse dépend de l'application visée (langue, registre) de l'évolution du vocabulaire à travers le temps, des ressources mémoire disponibles, etc.

rali

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.

rali

Plan

rali

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 peu adaptée (meilleure pour les gros comptes).
  - Kneser & Ney stable (les performances ne chutent pas lorsque n augmente).
- si $|\mathcal{T}|$ n'est pas très grand, alors un modèle bigramme et un modèle trigramme ont à peu près la même performance.

rali

Augmenter la taille des données

Stupid Backoff

[Brants et al., 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)

Stupid Backoff

[Brants et al., 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 (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.

rali

Stupid Backoff

[Brants et al., 2007]

- temps (d'entraînement) vs taille :

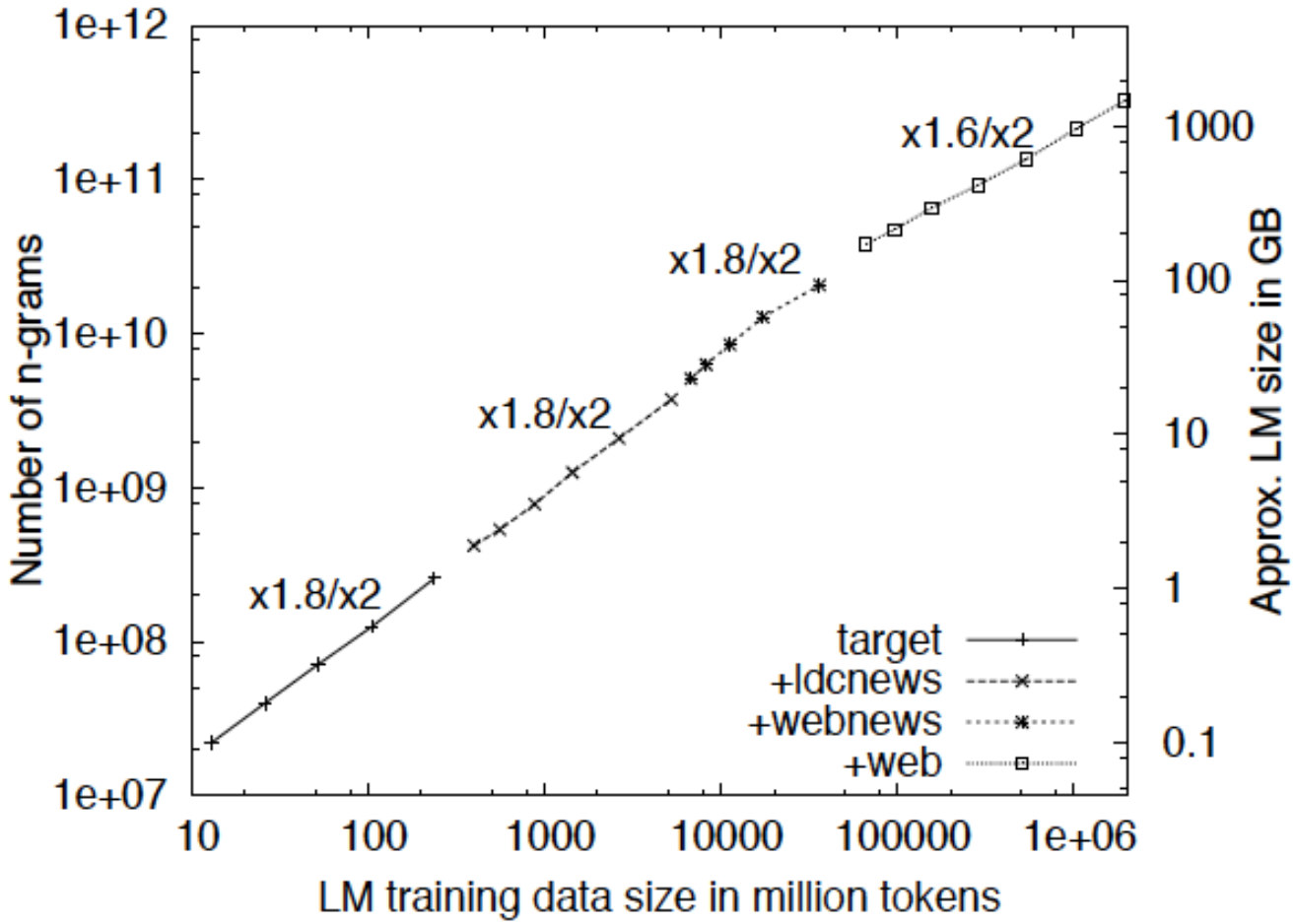
	target	webnews	web
$\|tokens\|$	237M	31G	1.8T
$\|V\|$	200k	5M	16M
$\|n-gram\|$	257M	21G	300G
$\|LM\|$ (SB)	2G	89G	1.8T
time (SB)	20 min	8h	1 day
time (KN)	2.5h	2 days	1 week [5]
# machines	100	400	1500

- 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).

5. temps estimé

Stupid Backoff

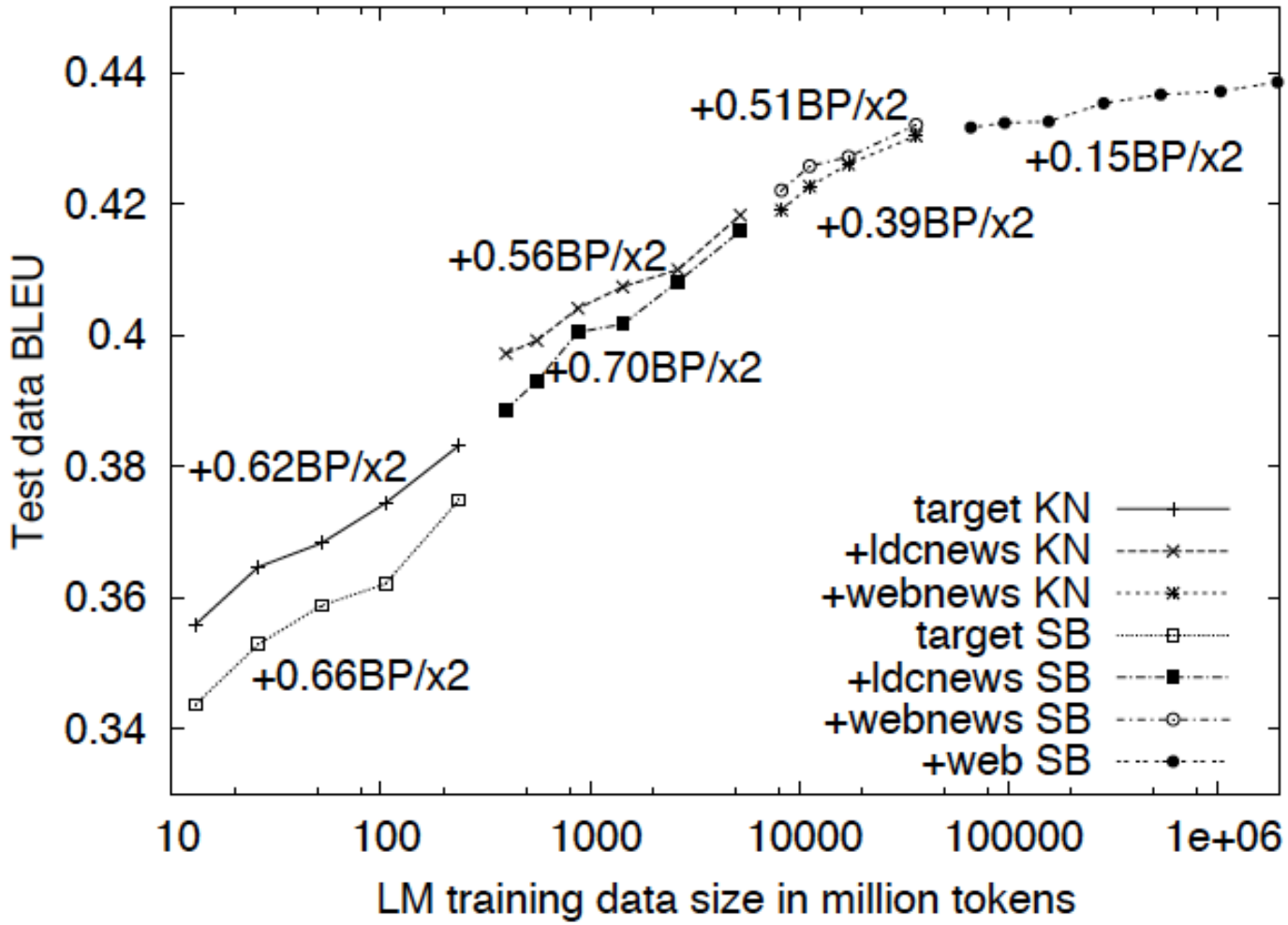
[Brants et al., 2007]



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

Stupid Backoff

[Brants et al., 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 (2009) :
  - 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 $|\texttt{il parle avec PIERRE de lui}| = 0$,
  - $\Rightarrow p(\texttt{lui} \mid \texttt{il parle avec --- de})$
- En pratique on combine différents modèles skippés avec un modèle non skippé (on augmente la complexité du modèle).
- peut servir à créer 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

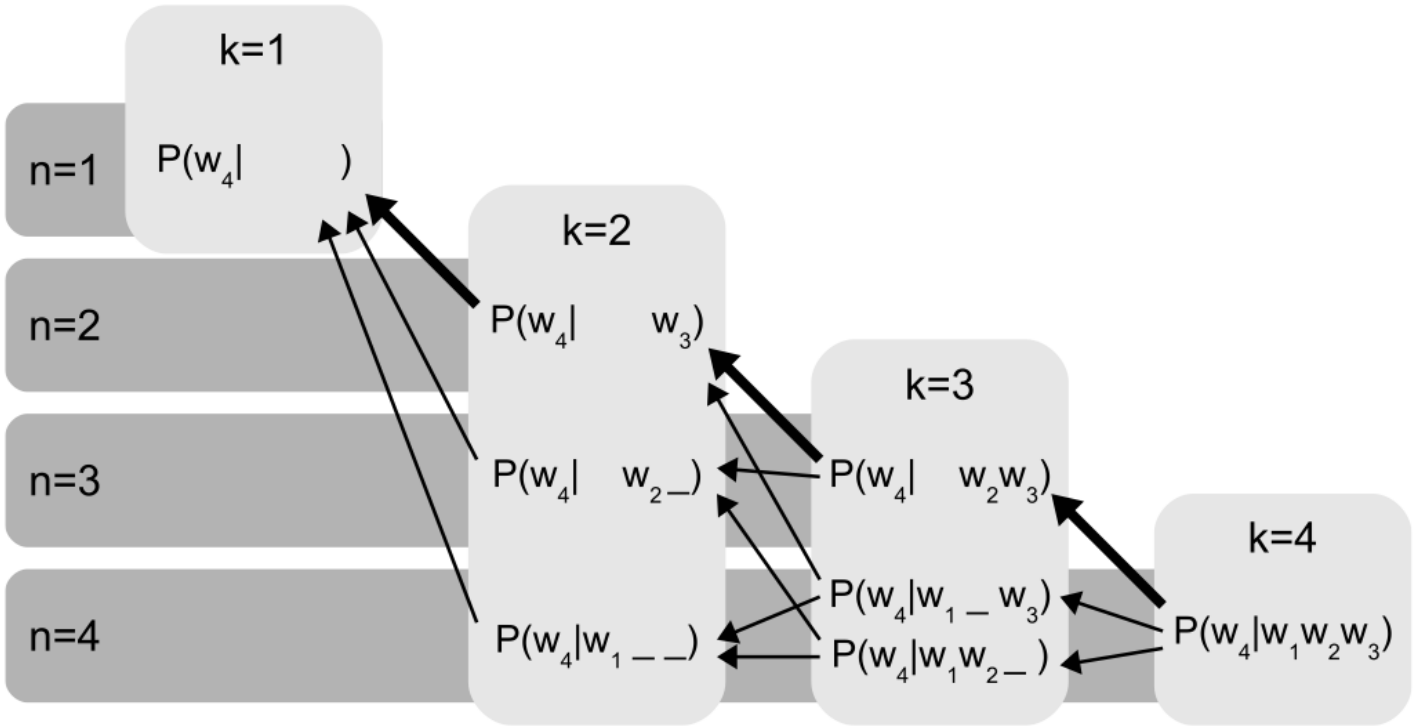
$$\text{manger une} \left\{ \begin{array}{l} \text{pomme} \\ \text{poire} \\ \text{pêche} \end{array} \right\} \text{avec} \left\{ \begin{array}{l} \text{délice} \\ \text{délectation} \\ \text{gourmandise} \end{array} \right\}$$

Mais :

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

Skipping et KN sont dans un bateau

[Pickhardt et al., 2014]



- baisse de perplexité (3% à 12% par rapport à MKN) plus marquée pour de petits corpus (25%)
- toolkit disponible
- testé sur 100 000 séquences de 5 mots (bizarre)

Clustering (regroupement)

Voir *[Brown et al., 1992]*

- 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 $\mathcal{T}$ 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 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]

rali

Clustering

- Plusieurs possibilités ; entre autres :

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

- 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).

Adaptation/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.

Adaptation/Modèle cache *[Kuhn and Mori, 1990]*

- si l'historique est bruité, le système peut se bloquer sur des erreurs [6] :

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
...	...

6. Exemple factice bien que probable.

Modèle à maximum d'entropie

- 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, ou fonction de partition : $(\mathcal{Z}(h) = \sum_w \exp \sum_i \lambda_i f_i(h, w))$;
  - 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).
- modèle de langue = ingénierie de *features*

Modèle à maximum d'entropie

- Exemple de trait :

$$f_{1267}(w_1^i) = \begin{cases} 1 & \text{si } \exists j \in [1, i[/ \begin{cases} i - j < 5 \\ w_j = \text{microsoft} \\ w_i = \text{bug} \end{cases} \\ 0 & \text{sinon} \end{cases}$$

si *microsoft* apparaît dans les 5 derniers mots de l'historique et que *bug* est le mot conjecturé, alors le feature est activé (*microsoft* est un **trigger**)

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

rali

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 et al., 1996]*** pour une bonne exposition à cette méthode d'apprentissage.

rali

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 :

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

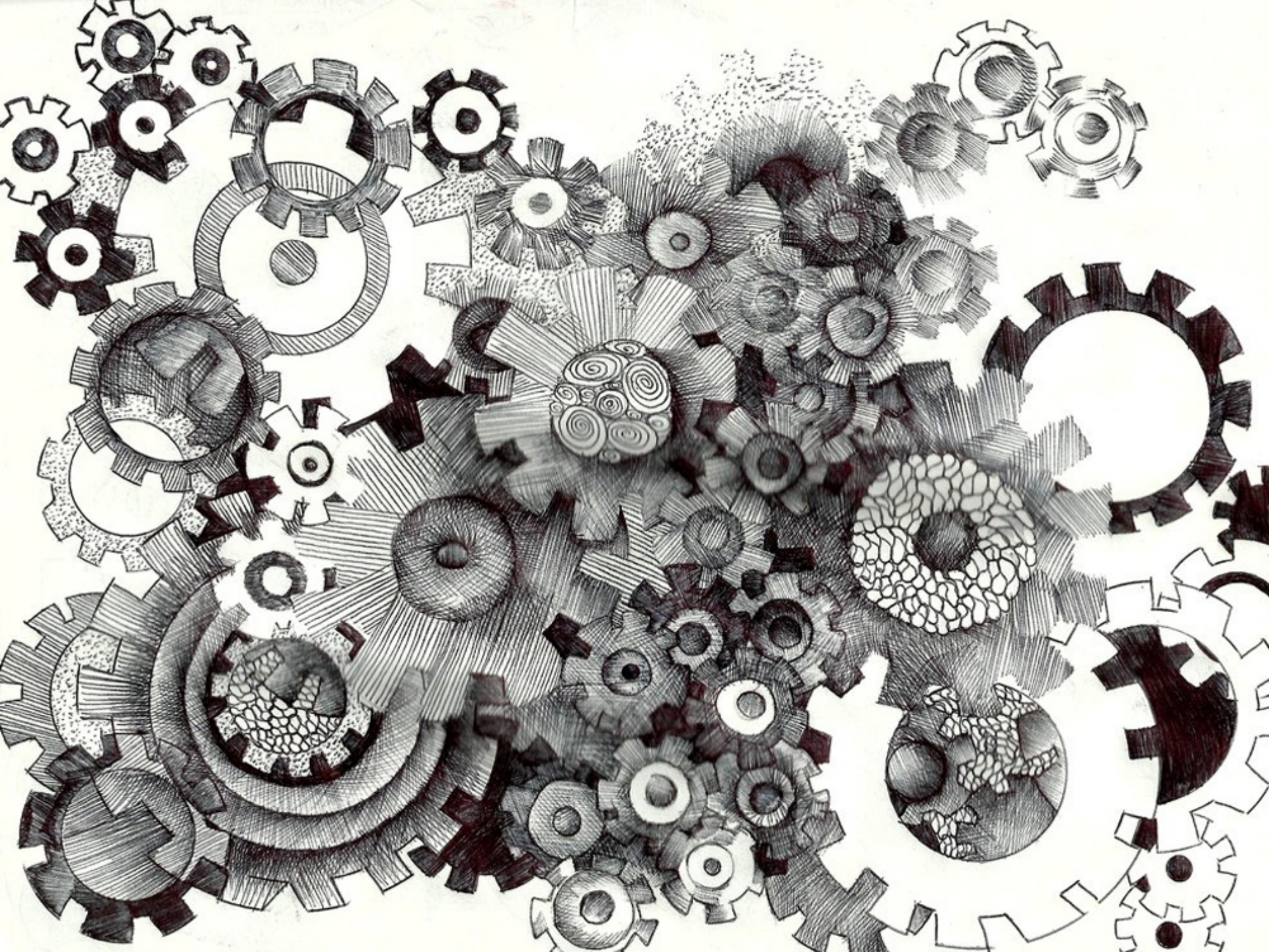
- $\mathcal{Z}$ est cette fois-ci constant et $p_0(s)$ est un modèle de départ (par exemple votre meilleur n-gramme)

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

Cons Problème non trivial de la découverte des traits pertinents.

Plan

rali



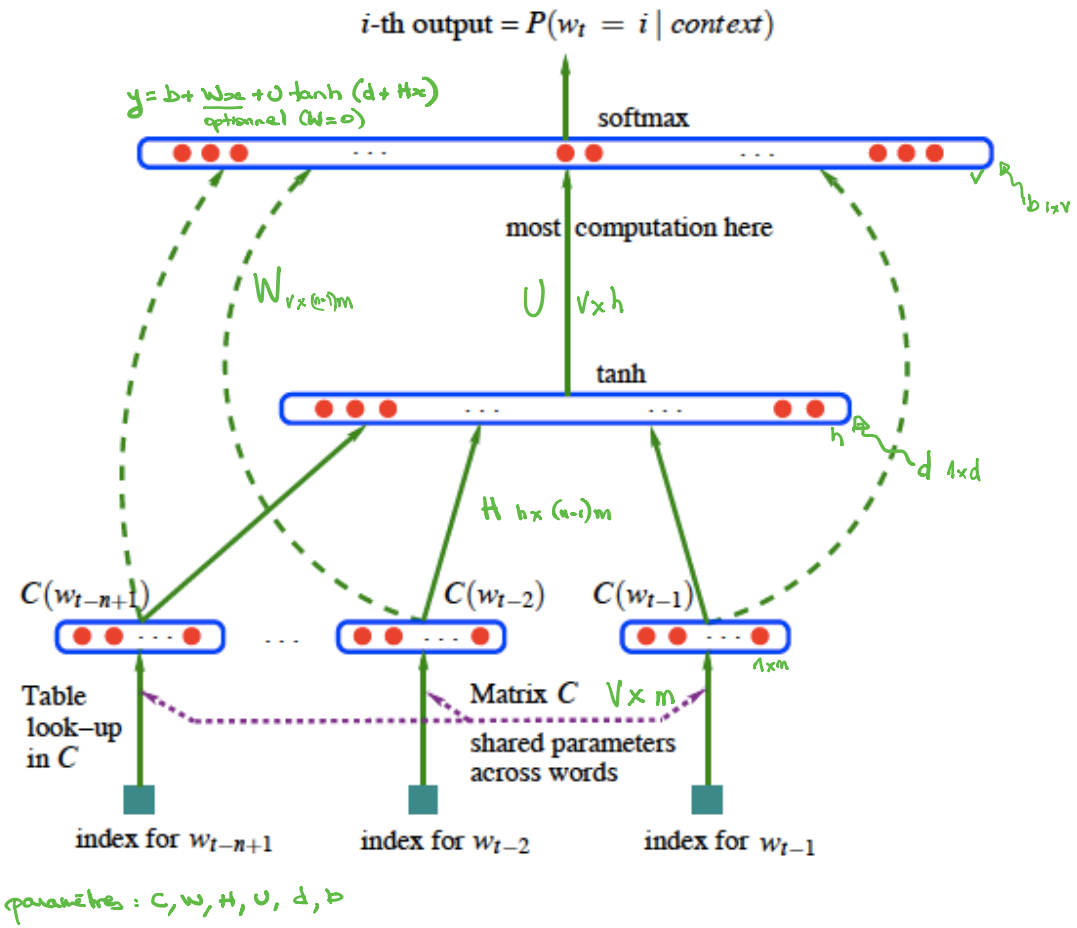
Le point de départ *[Bengio et al., 2001]*

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 (mot = vecteur) ; la projection est **apprise** sur le corpus d'entraînement.
  - **intuition :** deux mots partageant des contextes similaires (ex : cat & dog) sont sémantiquement reliés et ont des vecteurs proches $\rightarrow$ lissage sémantique plutôt que fréquentiel.
- 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.

rali

[Bengio et al., 2001]

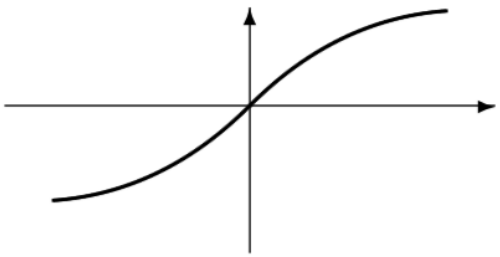


Fonctions d'activation classiques

Hyperbolic tangent

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

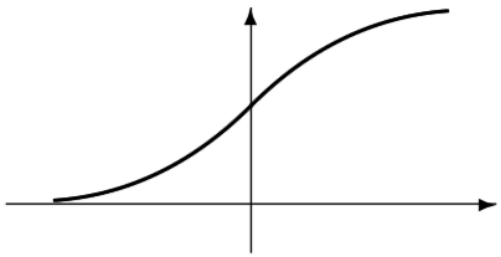
output ranges
from –1 to +1



Logistic function

$$\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$$

output ranges
from 0 to +1



Rectified linear unit

$$\text{relu}(x) = \max(0,x)$$

output ranges
from 0 to ∞

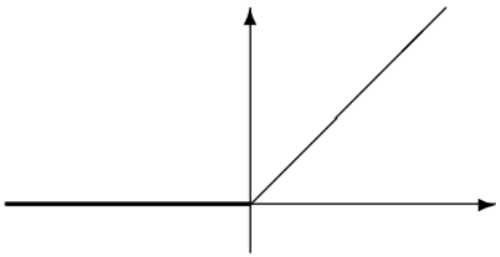


Figure 13.3: Typical activation functions in neural networks.

https ://arxiv.org/abs/1709.07809

Softmax

- pour $z = (z_1, \ldots, z_K) \in \mathbb{R}^K$,

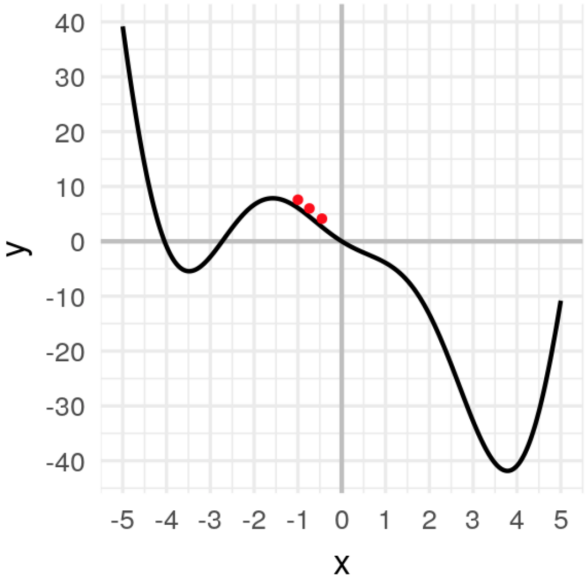
$$\sigma(z)_i = \frac{e^{z_i}}{\sum_i e^{z_i}}, \forall i \in [1, K]$$

  - $\sigma(z)_i$ est positif, et $\sum_i \sigma(z)_i = 1 \Rightarrow \sigma(z)$ est une distribution

- la partie coûteuse du modèle précédant (car K = |vocabulaire|)

Descente de gradient

- Les paramètres du modèle sont appris par descente de gradient (petits pas dans la direction inverse de la pente si on minimise) de façon à optimiser le logprob donné par le modèle au corpus d'entraînement.
  - Pour minimiser f, 1) choisir x_0 (alétoirement), 2) suivre le gradient : $x_t = x_{t-1} - \alpha f'(x_{t-1})$, où α est le learning rate (paramètre sensible)



Voir ce blog très bien fait pour une explication intuitive de la descente de gradient et ses problèmes.

[Bengio et al., 2003]

- Brown : 800K (train), 200k mots (dev), 181k mots (test)
  $|V|$ = 16k
- Associated Press : 14M mots (train), 1M (dev), 1M (test)

	Brown			AP News		
modèle	Dev	Test	ordre	Dev	Test	ordre
MLP (best)	265	252	$n = 5$	104	109	$n = 6$
JM	352	336	$n = 3$	126	132	$n = 3$
KN	332	321	$n = 3$	112	117	$n = 5$
class-based back-off	326	312	$n = 3$			

FFN *[Bengio et al., 2003]*

- pro : dépend faiblement de la taille de l'historique, généralise mieux
- cons :
  - la couche de sortie nécessite le calcul de toutes les probabilités (sommation sur tout le vocabulaire)
  - GPU realm . . .

- Énormément de travaux sur ces modèles.
- Lire *[Le et al., 2012]* pour une bonne introduction.

rali

[Mikolov et al., 2010]

- a **largement** contribué à la popularité des réseaux récurrents (RNN) en traitement des langues

	PPL
KN5	93.7
feedforward NN	85.1
recurrent NN	80.0
4$\times$RNN + KN5	73.5

Switchboard (4M de mots) — conversations téléphoniques

[Mikolov et al., 2010]

	PPL		WER	
Model	RNN	RNN+KN	RNN	RNN+KN
KN5 - baseline	-	221	-	13.5
RNN 60/20	229	186	13.2	12.6
RNN 90/10	202	173	12.8	12.2
RNN 250/5	173	155	12.3	11.7
RNN 250/2	176	156	12.0	11.9
RNN 400/10	171	152	12.5	12.1
3xRNN static	151	143	11.6	11.3
3xRNN dynamic	128	121	11.3	11.1

- RNN configuration is written as *hidden/threshold* - 90/10 means that network has 90 neurons in hidden layer and threshold for keeping words in vocabulary is 10.

- 37M de mots du NYT (6.4M seulement pour les RNNs)
  - plusieurs semaines d'entraînement
- RNN + KN = combinaison linéaire (0.75 sur le RNN)

Réseau récurrent

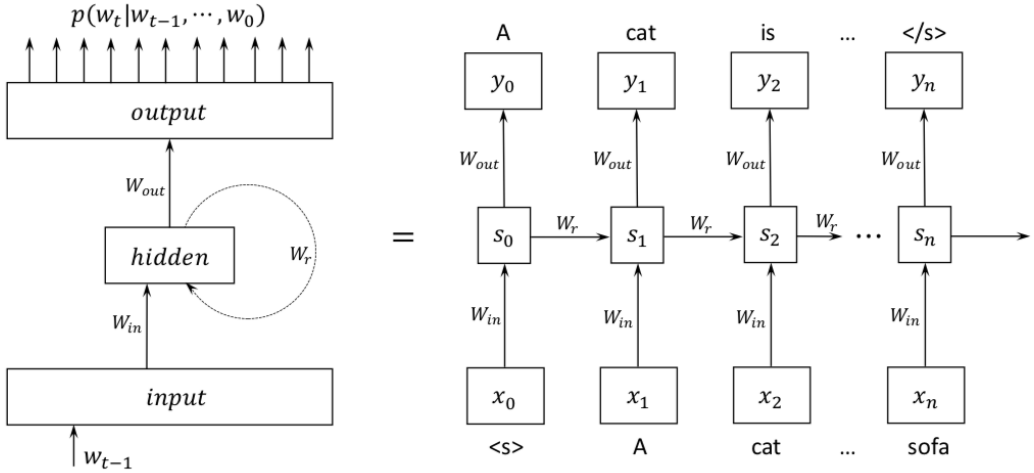


Figure 1: The network architecture of a standard RNNLM and its unrolled version for an example input sentence: <s> A cat is sitting on a sofa </s>.

fig. prise de : [Ji et al., 2015]

- $s_t = \sigma(W_i^T \times x_t + W_r \times s_{t-1} + b_i) \in \mathbb{R}^h$
  $W_i \in \mathbb{R}^{|V| \times h}$, $W_r \in \mathbb{R}^{h \times h}$, $b_i \in \mathbb{R}^h$, $\sigma(x) = 1/(1 + \exp(-x))$
- $y_t = f(W_o \times s_t + b_o) \in \mathbb{R}^{|V|}$
  $W_o \in \mathbb{R}^{|V| \times h}$, $b_o \in \mathbb{R}^{|V|}$, $f(v_i) = \exp(v_i)/\sum_j \exp(v_j)$

Réseau récurrent

- Pas de panique !
  - à un haut niveau d'abstraction : $h_t = \text{RNN}(w_t, h_{t-1})$ où w_t est l'*input* au temps t (le vecteur associé à x_t) et h_{t-1} est l'état caché au temps $t-1$.
- x_t est représenté sous forme d'un one-hot-vector ($\in \mathbb{R}^{|V|}$ avec des 0 partout sauf pour la dimension représentant le mot qui est à 1)

$$\begin{array}{ll} \text{chat} & (0,0,0,1,0,0,0,0,0,0,\ldots)^T \\ \text{chien} & (0,0,0,0,0,0,1,0,0,0,\ldots)^T \end{array}$$

- $\theta = (W_i, W_o, W_r, b_i, b_o)$ sont les paramètres du modèle

BiLSTM

- Les réseaux récurrents sont difficiles à entraîner *[Bengio et al., 1994]*
- solutions http://colah.github.io/posts/2015-08-Understanding-LSTMs/
  - LSTM *[Hochreiter and Schmidhuber, 1997]*
  - GRU *[Chung et al., 2014]*
- Si le contexte droit est pertinent pour la prédiction (et disponible), on peut également avoir une couche qui va de la droite vers la gauche
  - baptisé BiLSTM *[Graves et al., 2013]*
  - terriblement populaire (2015 et encore maintenant)

rali

BiLSTM

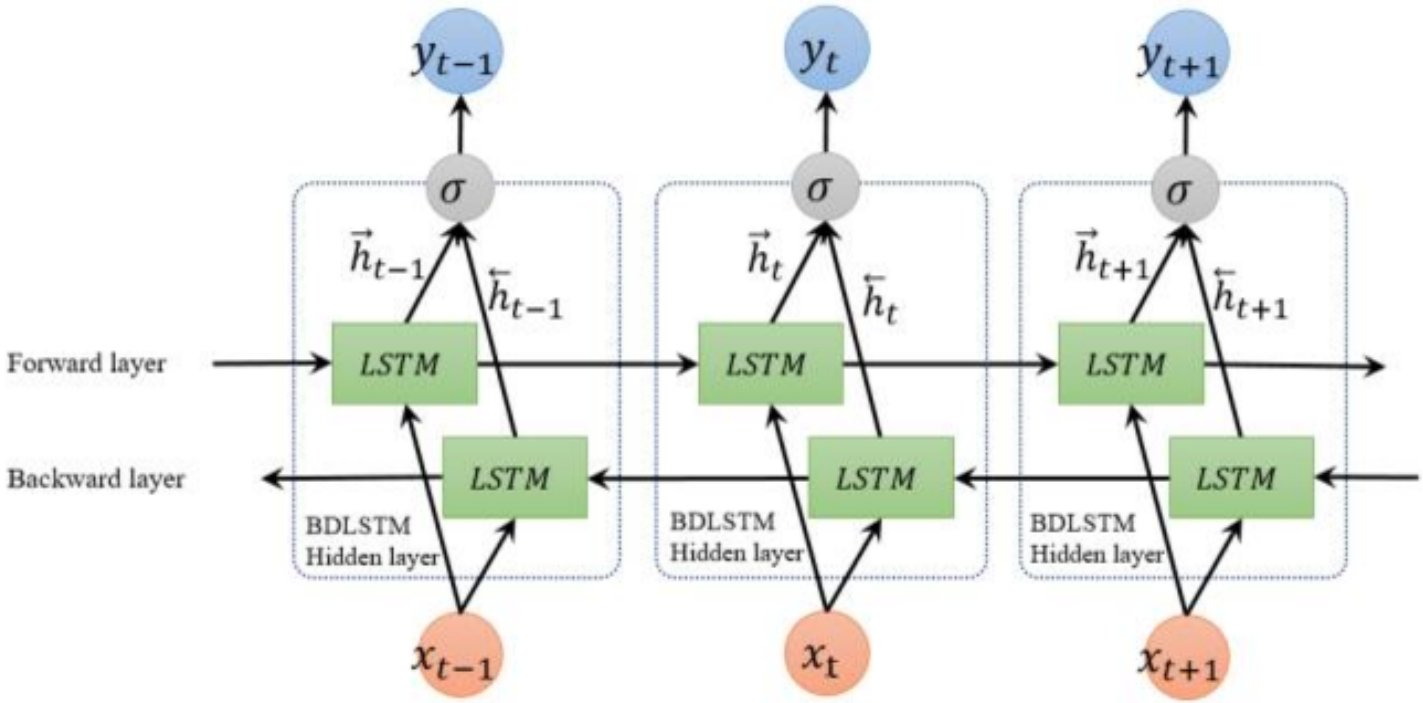


fig. prise de gabormelli.com

LSTM + CNN [Kim et al., 2015]

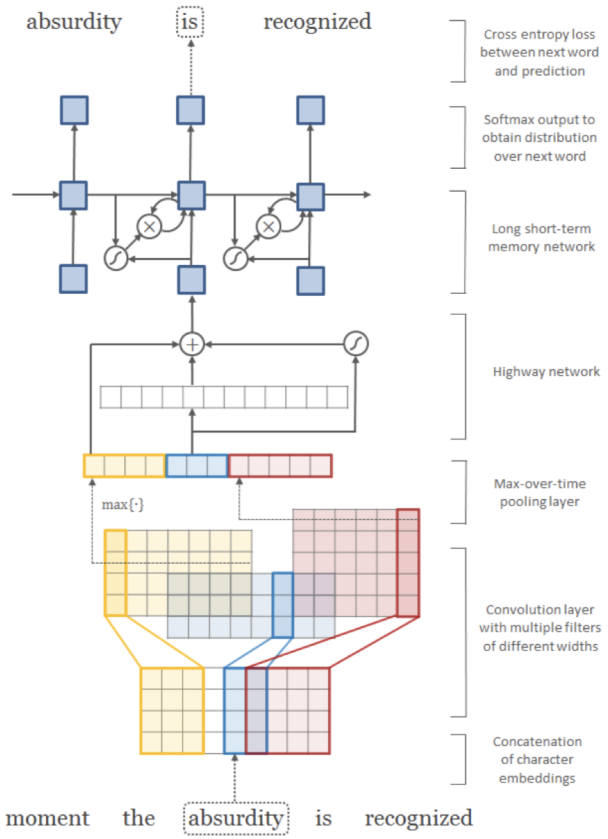


Fig 1. de l'article

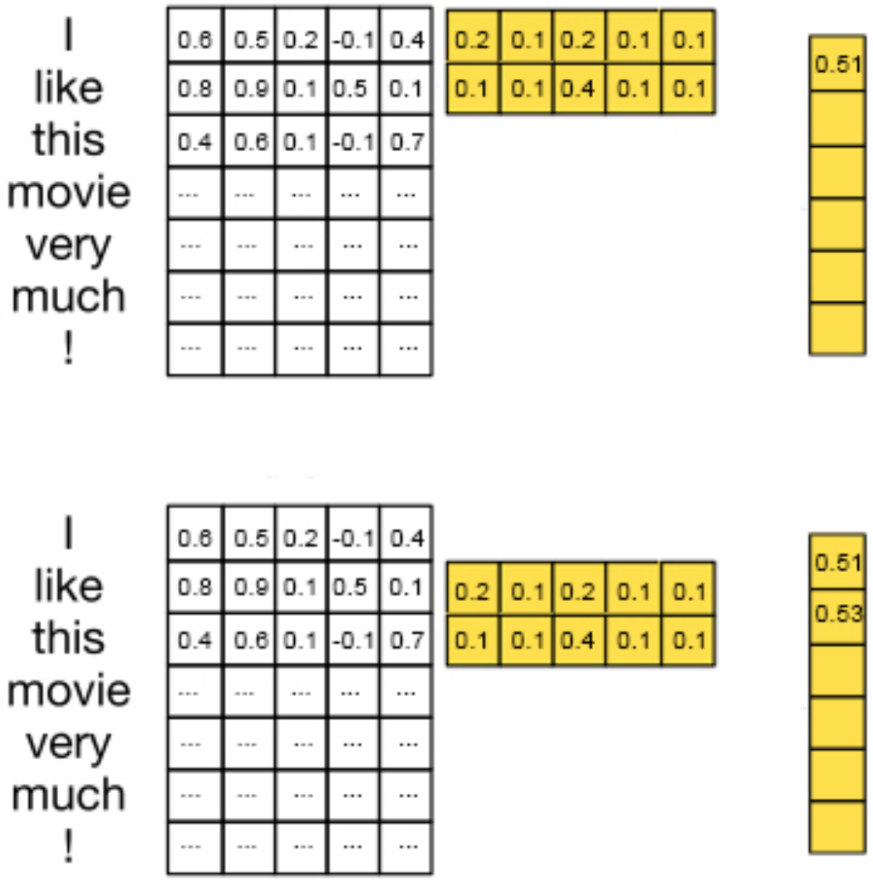
test sur Penn Treebank

w word

c char

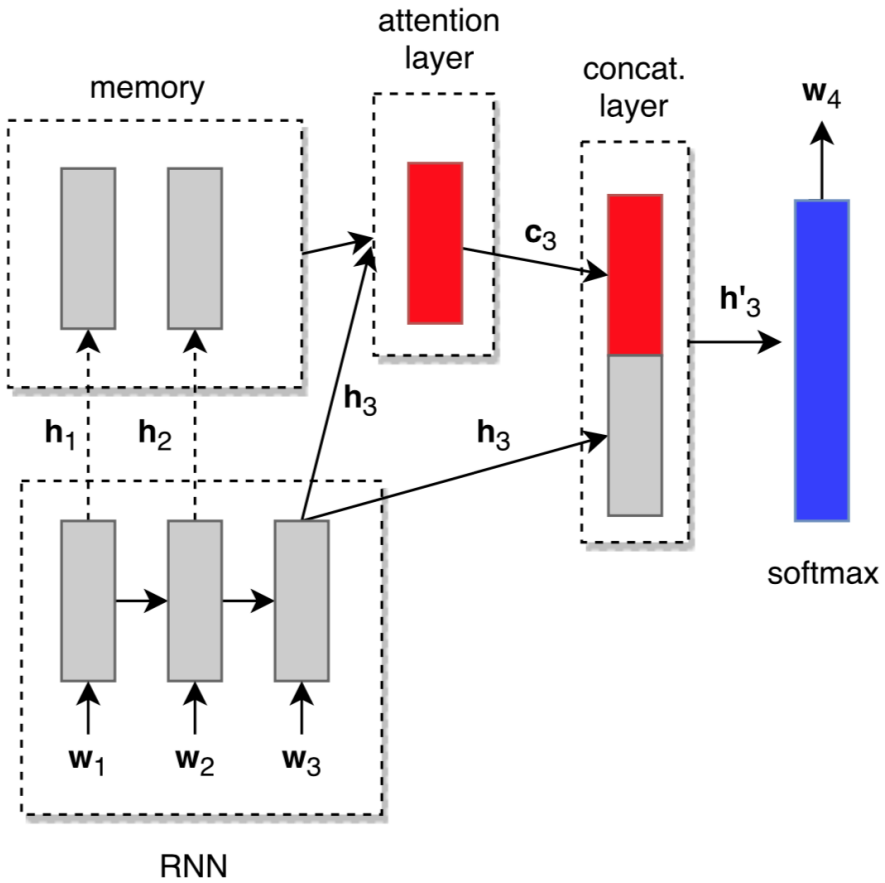
modèle		ppx
LSTM	w 200	97.6
LSTM	c 200	92.3
LSTM	w 650	85.4
LSTM	c 650	78.9
KN-5		141.2
RNN		124.7

À propos de convolution



Pris de cet excellent blog

LSTM avec attention *[Salton et al., 2017]*



$$c_t = \sum_{i=1}^{t-1} a_i h_i$$

$$a_i = \frac{\exp(score(h_i, h_t))}{\sum_{j=1}^{t-1} \exp(score(h_j, h_t))}$$

$$h'_t = \tanh(W_c[h_t; c_t] + b_t)$$

- h_t (sortie courante) sert de *sonde* aux sorties passées qui sont pondérées (a) pour créer un vecteur de contexte (c_t).

LSTM avec attention *[Salton et al., 2017]*

Model	Params	Valid. Set	Test Set
Single Models			
Medium Regularized LSTM (Zaremba et al., 2015)	20M	86.2	82.7
Large Regularized LSTM (Zaremba et al., 2015)	66M	82.2	78.4
Large + BD + WT (Press and Wolf, 2016)	51M	75.8	73.2
Neural cache model (size = 500) (Grave et al., 2017)	-	-	72.1
Medium Pointer Sentinel-LSTM (Merity et al., 2017)	21M	72.4	70.9
Attentive LM w/ *combined* score function	14.5M	72.6	70.7
Attentive LM w/ *single* score function	14.5M	71.7	**70.1**

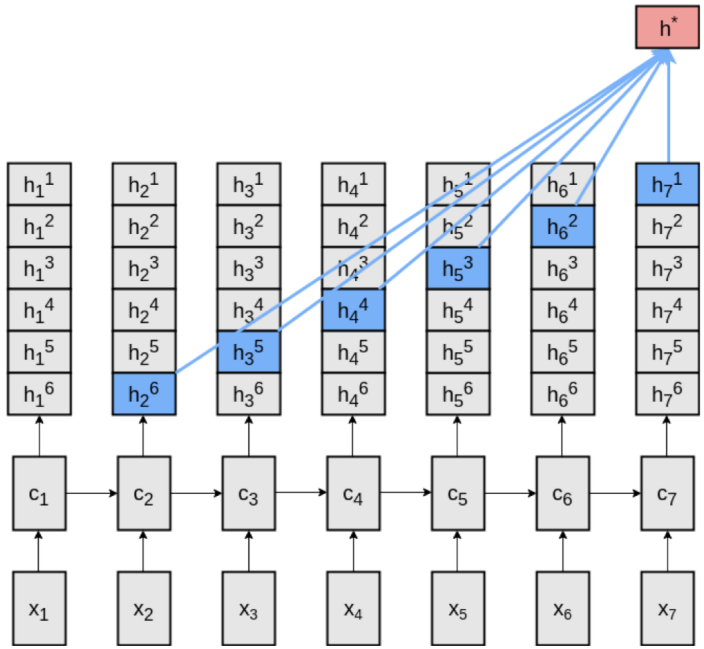
Model	Params	Valid. Set	Test Set
Zoneout + Variational LSTM (Merity et al., 2017)	20M	108.7	100.9
LSTM-LM (Grave et al., 2017)	-	-	99.3
Variational LSTM (Merity et al., 2017)	20M	101.7	96.3
Neural cache model (size = 100) (Grave et al., 2017)	-	-	81.6
Pointer LSTM (window = 100) (Merity et al., 2017)	21M	84.8	80.8
Attentive LM w/ *combined* score function	50M	74.3	70.8
Attentive LM w/ *single* score function	50M	73.7	69.7
Neural cache model (size = 2000) (Grave et al., 2017)	-	-	**68.9**

- Penn Tree Bank (news/financial) : 887K/70K/78K mots
- Wikitext2 : 2M/217K/245K mots

- l'analyse du modèle d'attention n'a rien révélé (poids d'attention distribués uniformément)

[Daniluk et al., 2017]

- proposent différents modèles d'attention, dont un modèle sans attention, N-gram RNN :
  - $h_t^\star = \tanh\left(W_N[h_t^1, h_{t-1}^2, \ldots h_{t-N+1}^{N-1}]^T\right), W_N \in \mathbb{R}^{k\times(n-1)k}$



(d) Concatenation of previous output representations.

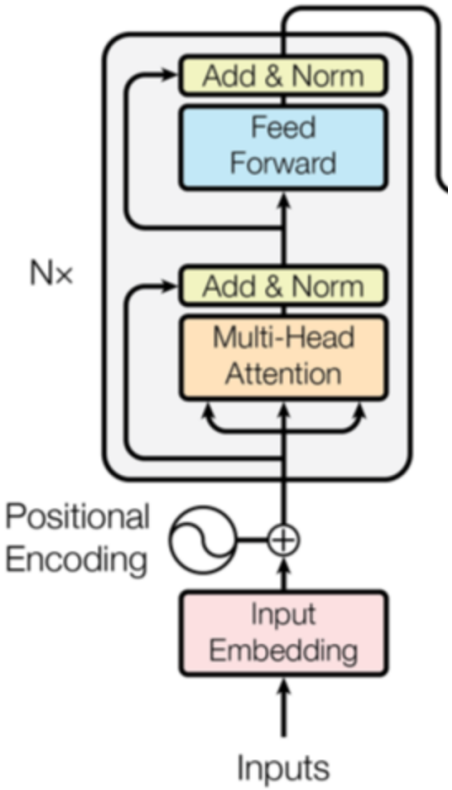
fig. 1d) prise de [Daniluk et al., 2017]

7500 art. eng. de Wikipedia associés à l'une de 5 catégories : personnes, pays, villes, universités et nouvelles

modèle	ppx
RNN	125.7
LSTM	85.2
Attention	82.0
4-gram RNN	75.9

Transformer (attention is all you need)

[Vaswani et al., 2017]



- Intuition : considérer toute l'entrée (modèle global)
- Un modèle complexe mais redoutable d'efficacité

Pris de l'article

Lire ce magnifique blog pour une description claire du modèle

ELMO/GPT/(Ro)BERT(a)/XLNet sont dans un bateau

- (nouveau) point tournant en 2017/2018 : de gros modèles de langues que l'on peut ajuster (fine tuner) rapidement sur une tâche d'intérêt (modèle de langue = modèle universel)

ELMO ***[Peters et al., 2018]*** plongement de caractères, modèle bidirectionnel (biLM)
`https://allennlp.org/elmo`

GPT ***[Radford et al., 2018]*** entraîné sur le sous-ensemble du Common Crawl vers lequel pointent des articles de Reddit et autres réseaux sociaux – 8M de documents, 40GB de texte

[Radford et al., 2018]

Language Models are Unsupervised Multitask Learners

	LAMBADA (PPL)	LAMBADA (ACC)	CBT-CN (ACC)	CBT-NE (ACC)	WikiText2 (PPL)	PTB (PPL)	enwik8 (BPB)	text8 (BPC)	WikiText103 (PPL)	1BW (PPL)
SOTA	99.8	59.23	85.7	82.3	39.14	46.54	0.99	1.08	18.3	**21.8**
117M	**35.13**	45.99	**87.65**	**83.4**	**29.41**	65.85	1.16	1.17	37.50	75.20
345M	**15.60**	55.48	**92.35**	**87.1**	**22.76**	47.33	1.01	**1.06**	26.37	55.72
762M	**10.87**	**60.12**	**93.45**	**88.0**	**19.93**	**40.31**	**0.97**	**1.02**	22.05	44.575
1542M	**8.63**	**63.24**	**93.30**	**89.05**	**18.34**	**35.76**	**0.93**	**0.98**	**17.48**	42.16

Table 3. Zero-shot results on many datasets. No training or fine-tuning was performed for any of these results. PTB and WikiText-2 results are from (Gong et al., 2018). CBT results are from (Bajgar et al., 2016). LAMBADA accuracy result is from (Hoang et al., 2018) and LAMBADA perplexity result is from (Grave et al., 2016). Other results are from (Dai et al., 2019).

rali

ELMO/GPT/(Ro)BERT(a)/XLNet sont dans un bateau

- (nouveau) point tournant en 2017/2018 : de gros modèles de langues que l'on peut ajuster (fine tuner) rapidement sur une tâche d'intérêt (modèle de langue = modèle universel)

ELMO ***[Peters et al., 2018]*** plongement de caractères, modèle bidirectionnel (biLM)
`https://allennlp.org/elmo`

GPT ***[Radford et al., 2018]*** entraîné sur le sous-ensemble du Common Crawl vers lequel pointent des articles de Reddit et autres réseaux sociaux – 8M de documents, 40GB de texte

BERT ***[Devlin et al., 2018]*** Bidirectional Transformer. Entraîné sur 800M mots de BooksCorpus + Wikipedia (2,500M mots)

rali

ELMO/GPT/(Ro)BERT(a)/XLNet sont dans un bateau

- (nouveau) point tournant en 2017/2018 : de gros modèles de langues que l'on peut ajuster (fine tuner) rapidement sur une tâche d'intérêt (modèle de langue = modèle universel)

ELMO ***[Peters et al., 2018]*** plongement de caractères, modèle bidirectionnel (biLM)
`https://allennlp.org/elmo`

GPT ***[Radford et al., 2018]*** entraîné sur le sous-ensemble du Common Crawl vers lequel pointent des articles de Reddit et autres réseaux sociaux – 8M de documents, 40GB de texte

BERT ***[Devlin et al., 2018]*** Bidirectional Transformer. Entraîné sur 800M mots de BooksCorpus + Wikipedia (2,500M mots)

XLNet ***[Yang et al., 2019]*** Entraîné sur encore plus de textes (512 TPU pendant 2.5 jours)

rali

[Chelba et al., 2013]

- un benchmark de taille raisonnable ($\sim$ 1 milliard de mots)
- extrait de WMT 11 (campagne d'évaluation de traduction) : common crawl data, *English*
  - normalisation minimaliste
    - ponctuations, apostrophes
    - phrases dédoublées (2.9 milliards de mots à 0.8)
  - voc. size : 793 471 (mots vus au moins 3 fois, autres tokens mappés vers `UNK`)
  - ramdomisé (eurk)
- découpé en 100 tranches :
  - 99 tranches pour l'entrainement
  - tranche heldout découpée en 50
    - 1 de ces tranches utilisée pour les tests (les 49 autres) pour l'ajustement des paramètres
    - OOV rate : 0.28%
  - `http://www.statmt.org/lm-benchmark/`

[Chelba et al., 2013]

Model	Num. Params [billions]	Training Time [hours]	[CPUs]	Perplexity
Interpolated KN 5-gram, 1.1B n-grams (KN)	1.76	3	100	67.6
Katz 5-gram, 1.1B n-grams	1.74	2	100	79.9
Stupid Backoff 5-gram (SBO)	1.13	0.4	200	87.9
Interpolated KN 5-gram, 15M n-grams	0.03	3	100	243.2
Katz 5-gram, 15M n-grams	0.03	2	100	127.5
Binary MaxEnt 5-gram (n-gram features)	1.13	1	5000	115.4
Binary MaxEnt 5-gram (n-gram + skip-1 features)	1.8	1.25	5000	107.1
Hierarchical Softmax MaxEnt 4-gram (HME)	6	3	1	101.3
Recurrent NN-256 + MaxEnt 9-gram	20	60	24	58.3
Recurrent NN-512 + MaxEnt 9-gram	20	120	24	54.5
Recurrent NN-1024 + MaxEnt 9-gram	20	240	24	51.3

Table 1: Results on the 1B Word Benchmark test set with various types of language models.

rali

[Chelba et al., 2013]

Model	Perplexity
Interpolated KN 5-gram, 1.1B n-grams	67.6
All models	**43.8**

Table 2: Model combination on the 1B Word Benchmark test set. The weights were tuned to minimize perplexity on held-out data. The optimal interpolation weights for the KN, rnn1024, rnn512, rnn256, SBO, HME were, respectively: 0.06, 0.61, 0.13, 0.00, 0.20, 0.00.

rali

NLPProgress

- Prenons le temps d'aller voir les performances actuelles : http://nlpprogress.com/english/language_modeling.html
- **note :** ce site Web est une mine d'or !

Plan

rali


So what?

[Shen et al., 2017]

Est-il encore utile de faire des travaux en modélisation de la langue ?

- basé sur un test de type Turing : tester si un modèle peut générer des phrases sans qu'un juge ne soit capable de les discerner de phrases existantes.
- jugements humains sur une échelle de 4 points :
  - 3-clairement humain,
  - 2-un peu humain,
  - 1-un peu inhumain,
  - 0-clairement non humain
- 400 phrases (au moins 16 mots, les 8 premiers sont conservés, les autres sont générés par le modèle (approche gloutonne))
- "The results imply that LMs need about 10 to 20 more years of research before human performance is reached"
- perplexité estimée d'un modèle de langue humain : 12

rali

[Shen et al., 2017]

Sur une version tronquée sur *One Billion Word corpus*

modèle	ppl	top1
3gram	112.2	24.0
5gram	73.7	31.2
MaxEnt (5g features)	68.8	31.8
FFNN	83.0	26.3
RNN	45.7	31.9
LSTM*	33.6	36.2
humain (estimée)	12.0	40.5

ppl perplexité

top1 percentage de mots prédits correctement en tête

* 2 semaines d'entraînement pour le meilleur modèle

RNN versus KN *[Tang and Lin, 2018]* [7]

- test sur un Rasberry Pi (meme architecture qu'un téléphone) en mesurant la latence (ms par requête) et la consommation d'énergie (mJ par requête) !
- trois modèles : un basé sur les comptes (KN-5), un LSTM (AWD) et un RNN (QRNN)
- deux benchmarks :

PTB	tokens : 887k (train), 70k (valid), 78k (test) – domaine de la finance
WikiText-103	tokens : 103M (train), 217k (valid), 245k (test) – Wikipedia

- deux mesures d'évaluation :

perplexité	(colonne Test dans la table qui suit)
r@3	pourcentage de mots attendus proposés en top-3 (j'aurais appelé cela p@3)

7. https ://arxiv.org/abs/1811.00942

RNN versus KN *[Tang and Lin, 2018]* [8]

#	Method	Model Quality			RPi		CPU | GPU	
		Val.	Test	R@3	ms/q	mJ/q	ms/q	ms/q
	Penn Treebank							
1	KN-5	148.4	141.5	36.7%	7	6	0.8	–
2	AWD	59.2	56.8	44.9%	223	295	7.9	1.7
3	QRNN	59.1	56.8	44.7%	224	296	7.5	1.6
	WikiText-103							
4	KN-5	145.2	152.7	39.8%	264	229	37	–
5	QRNN	31.9	32.8	53.5%	1240	1480	59	3.5

Table 2: Language modeling results on performance and model quality.

8. https ://arxiv.org/abs/1811.00942

RNN versus KN *[Tang and Lin, 2018]* [8]

#	Method	Model Quality			RPi		CPU | GPU	
		Val.	Test	R@3	ms/q	mJ/q	ms/q	ms/q
	Penn Treebank							
1	KN-5	148.4	141.5	36.7%	7	6	0.8	–
2	AWD	59.2	56.8	44.9%	223	295	7.9	1.7
3	QRNN	59.1	56.8	44.7%	224	296	7.5	1.6
	WikiText-103							
4	KN-5	145.2	152.7	39.8%	264	229	37	–
5	QRNN	31.9	32.8	53.5%	1240	1480	59	3.5

Table 2: Language modeling results on performance and model quality.

- réduction de perplexité . . . impressionnante !

8. https ://arxiv.org/abs/1811.00942

RNN versus KN *[Tang and Lin, 2018]* [8]

#	Method	Model Quality			RPi		CPU | GPU	
		Val.	Test	R@3	ms/q	mJ/q	ms/q	ms/q
	Penn Treebank							
1	KN-5	148.4	141.5	36.7%	7	6	0.8	–
2	AWD	59.2	56.8	44.9%	223	295	7.9	1.7
3	QRNN	59.1	56.8	44.7%	224	296	7.5	1.6
	WikiText-103							
4	KN-5	145.2	152.7	39.8%	264	229	37	–
5	QRNN	31.9	32.8	53.5%	1240	1480	59	3.5

Table 2: Language modeling results on performance and model quality.

- augmentation de la latence . . . impressionnante
  - 49x plus lent sur PTB
  - latence de 1.2s sur WikiText dans le cas du RNN !

8. https ://arxiv.org/abs/1811.00942

RNN versus KN *[Tang and Lin, 2018]* [8]

#	Method	Model Quality			RPi		CPU | GPU	
		Val.	Test	R@3	ms/q	mJ/q	ms/q	ms/q
	Penn Treebank							
1	KN-5	148.4	141.5	36.7%	7	6	0.8	–
2	AWD	59.2	56.8	44.9%	223	295	7.9	1.7
3	QRNN	59.1	56.8	44.7%	224	296	7.5	1.6
	WikiText-103							
4	KN-5	145.2	152.7	39.8%	264	229	37	–
5	QRNN	31.9	32.8	53.5%	1240	1480	59	3.5

Table 2: Language modeling results on performance and model quality.

- consommation du RNN plus grande (32x sur PTB). Note : un téléphone gère >10k joules

8. https ://arxiv.org/abs/1811.00942

RNN versus KN *[Tang and Lin, 2018]* [8]

#	Method	Model Quality			RPi		CPU | GPU	
		Val.	Test	R@3	ms/q	mJ/q	ms/q	ms/q
	Penn Treebank							
1	KN-5	148.4	141.5	36.7%	7	6	0.8	–
2	AWD	59.2	56.8	44.9%	223	295	7.9	1.7
3	QRNN	59.1	56.8	44.7%	224	296	7.5	1.6
	WikiText-103							
4	KN-5	145.2	152.7	39.8%	264	229	37	–
5	QRNN	31.9	32.8	53.5%	1240	1480	59	3.5

Table 2: Language modeling results on performance and model quality.

- consommation et de latence plus proches sur un (C/G)PU (tout de même en faveur de KN)

8. https ://arxiv.org/abs/1811.00942

RNN versus KN *[Tang and Lin, 2018]* [8]

#	Method	Model Quality			RPi		CPU | GPU	
		Val.	Test	R@3	ms/q	mJ/q	ms/q	ms/q
	Penn Treebank							
1	KN-5	148.4	141.5	36.7%	7	6	0.8	–
2	AWD	59.2	56.8	44.9%	223	295	7.9	1.7
3	QRNN	59.1	56.8	44.7%	224	296	7.5	1.6
	WikiText-103							
4	KN-5	145.2	152.7	39.8%	264	229	37	–
5	QRNN	31.9	32.8	53.5%	1240	1480	59	3.5

Table 2: Language modeling results on performance and model quality.

- les différences r@3 ne sont pas aussi marquées que les différences de perplexité

8. https ://arxiv.org/abs/1811.00942

RNN versus KN *[Tang and Lin, 2018]* [9]

Soit le corpus de test : `choo choo train` (on a déjà vu cela) et les modèles unigramme suivants sur le vocabulaire $\{\text{choo}, \text{train}\}$:

$$p_1 : p_1(\text{choo}) = 0.1, p_1(\text{train}) = 0.9$$
$$p_2 : p_2(\text{choo}) = 0.49, p_2(\text{train}) = 0.51$$

Perplexité en test :

$$p_1 : 2^{-((log_2(0.1)+log_2(0.1)+log_2(0.9))/3)} \approx 4.8$$
$$p_2 : 2^{-((log_2(0.49)+log_2(0.49)+log_2(0.51))/3)} \approx 2.01$$

Tant que $p(\text{choo}) < 0.5$, r@1 vaudra 1/3, tandis que la perplexité peut varier.

9. https ://arxiv.org/abs/1811.00942

Show some love *[Shareghi et al., 2019]*

- testé sur :
  - un benchmark de 50 langues *[Gerz et al., 2018]* ($\sim$ 40k phrases par langue)
  - 3 langues d'Europarl *[Koehn, 2005]* (10x plus gros)
- 3 modèles basés sur les comptes : KN, MKN, GKN
  (encore de la recherche sur les modèles basés sur les comptes *[Shareghi et al., 2016]*)
- 3 modèles neuronaux : LSTM, CNN, Attack-Preserve
  [Gerz et al., 2018]

rali

Show some love *[Shareghi et al., 2019]*

- Sur le corpus de 50 langues :
  - KN $>$ LSTM : 26/50, GKN$>$LSTM : 42/50, BKN$>$LSTM : 48/50
  - AP (whatever it is :-) est meilleur
  - BKN $>$ KN
- Sur Europarl (plus grand) : modèles neuronaux meilleurs
  - AP : 26h d'entraînement sur un GPU
  - KN : 6 secondes (!) sur un CPU
- Détail : pointent une pratique (cutoff) dans les évaluations qui défavorise les modèles basés sur les comptes (11% de perte en ppx)

rali

Show some love *[Shareghi et al., 2019]*

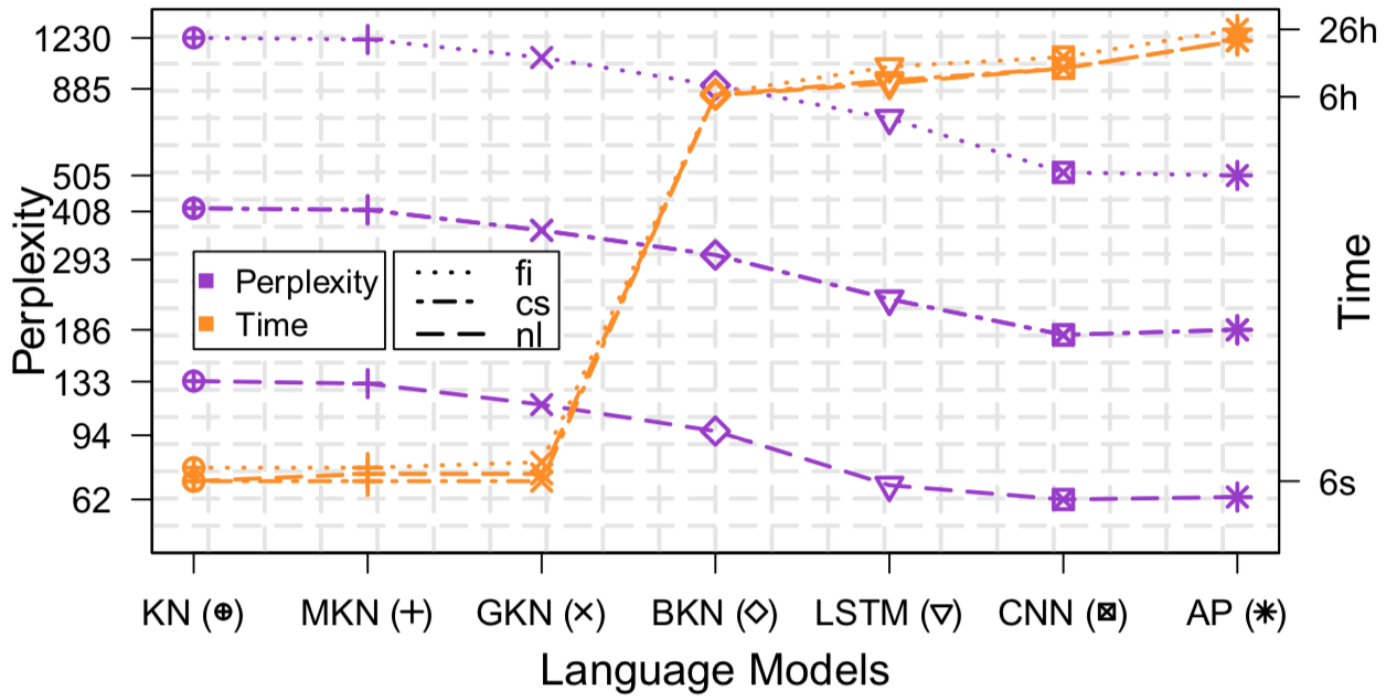


Figure 1: Time-Perplexity for *n*-gram and neural LMs across 3 languages fi, cs, nl. Timings done on a single core of AMD Ryzen 1900X for *n*-grams, and on a Nvidia TITAN X Pascal GPU for neural models.

Modèles neuronaux à grande échelle

NVidia : ***[Puri et al., 2018]*** – Converging on 40GB of Text in Four hours

- LSTM simple modélisant des séquences de caractères (V est petit)
- sur une architecture gérant 128 NVidia Tesla V100 GPUs en faisant du parallélisme sur les données (*batches*)
  - 109x plus rapide que d'utiliser un seul GPU

Baidu ***[Patwary et al., 2018]*** – en exploitant la loi de Zipf

- modèles de mots (V=100k), FP16, sampled softmax
- 3.5 heures par epoch (64 GPU)

Voir aussi les travaux sur la distillation

Plan

rali

Packages (non neuronaux)

- 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`
- BerkeleyLM ***[Pauls and Klein, 2011]***
  `http://code.google.com/p/berkeleylm/`
  ($\sim$ speed, 1/4 de la mémoire requise par SRILM)
- KenLM ***[Heafield, 2011]***
  `https://kheafield.com/code/kenlm/`
  (plus rapide que BerkeleyLM, populaire)

rali

SRILM

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

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

où :

  - $train est un texte (phrase = ligne, mots séparés par des espaces)
  - $out le répertoire où le modèle sera stocké

- consultation des probabilités (pas très pratique)

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

où : $dev est un texte (même format que $train)

KenLM

1. Installation de KenLM : github

   note : dépendance à Cmake (qui requiert boost), mais ça se fait, même sur un mac !

2. Entrainer un modèle : page projet

```
bin/lmplz -o 5 -S 80% -T /tmp < text_file >text.arpa
bin/build_binary text.arpa text.bin
```

3. Installer le module **python** :

```
pip install https://github.com/kpu/kenlm/archive/mast
```

4. Accès au log prob d'une phrase

```
import kenlm
model = kenlm.Model('text.bin') # ou .arpa
print(model.score('this is a sentence .', bos = True,
```

KenLM : requête avec état

```
state_in = kenlm.State();
state_out = kenlm.State();

model.BeginSentenceWrite(state_in)
# alternative (no <bos>): model.NullContextWrite(state_in)

s_le = model.baseScore(state_in, "le", state_out)
s_la = model.baseScore(state_in, "la", state_out)

print(f"p(le|<bos>)={s_le} p(la | <bos>)={s_la}")

print("p(femme | <bos> la))= {}" % model.baseScore(state_out, "femme",
print("p(rousse | la femme))= {}" % model.baseScore(state_in, "rousse"
#...
# !!! state_in = state_out ne fonctionne pas comme attendu
```

Packages (neuronaux)

- https ://www.lsv.uni-saarland.de/index.php ?id=200

About us
Teaching
Research
Projects
Publications
Resources
Announcements
Impressum
Datenschutz

Neural Network Language Modeling Toolkit

TF-NNLM-TK is a toolkit written in Python3 for neural network language modeling using Tensorflow. It includes basic models like RNNs and LSTMs as well as more advanced models. It provides functionality to preprocess the data, train the models and evaluate them. The toolkit is open-source under the Apache 2 license.

Currently, the following models are supported:

- Vanilla-RNN
- LSTM
- LSTM with projection
- GRU
- Sequential RNN (word-dependent, word-independent and with forgetting-factor)
- Long-Short Range Context

The code, introductory examples and more information can be found on GitHub.

rali

Bengio, Y., Ducharme, R., and Vincent, P. (2001).
A neural probabilistic language model.
In *Advances in Neural Information Processing Systems*.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003).
A neural probabilistic language model.
J. Mach. Learn. Res., 3 :1137–1155.

Bengio, Y., Simard, P., and Frasconi, P. (1994).
Learning long-term dependencies with gradient descent is difficult.
Trans. Neur. Netw., 5(2) :157–166.

Berger, A., Pietra, S. D., and Pietra, V. D. (1996).
A maximum entropy approach to natural language processing.
Computational Linguistics, 22(1) :39–71.

Brants, T., Popat, A. C., Xu, P., Och, F. J., and Dean, J. (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)*, pages 858–867, Prague, Czech Republic. Association for Computational Linguistics.

Brown, P., Pietra, S. D., Pietra, V. D., Lai, J., and Mercer, R. (1992).
An estimate of an upper bound for the entropy of english.
Computational Linguistics, 18(1) :31–40.

Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., and Koehn, P. (2013).
One billion word benchmark for measuring progress in statistical language modeling.
CoRR, abs/1312.3005.

Chen, S. F. (1996).
Building probabilistic models for natural language.

Chen, S. F. and Goodman, J. (1996).

rali

An empirical study of smoothing techniques for language modeling.
In Joshi, A. and Palmer, M., editors, *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, pages 310–318, San Francisco. Morgan Kaufmann Publishers.

Chung, J., Gülçehre, Ç., Cho, K., and Bengio, Y. (2014).
Empirical evaluation of gated recurrent neural networks on sequence modeling.
CoRR, abs/1412.3555.

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

Daniluk, M., Rocktäschel, T., Welbl, J., and Riedel, S. (2017).
Frustratingly short attention spans in neural language modeling.
CoRR, abs/1702.04521.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018).

BERT : pre-training of deep bidirectional transformers for language understanding.
CoRR, abs/1810.04805.

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

Gerz, D., Vulić, I., Ponti, E., Naradowsky, J., Reichart, R., and Korhonen, A. (2018).
Language modeling for morphologically rich languages : Character-aware modeling for word-level prediction.
Transactions of the Association for Computational Linguistics, 6 :451–465.

Goodman, J. (2001).
A bit of progress in language modeling.
Computer Speech and Language, pages 403–434.

Graves, A., Mohamed, A., and Hinton, G. E. (2013).
Speech recognition with deep recurrent neural networks.
CoRR, abs/1303.5778.

Heafield, K. (2011).
KenLM : faster and smaller language model queries.
In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, United Kingdom.

Hochreiter, S. and Schmidhuber, J. (1997).
Long short-term memory.
Neural computation, 9 :1735–1780.

Jelinek, F. and Lafferty, J. D. (1991).
Computation of the probability of initial substring generation by stochastic context-free grammars.
Computational Linguistics, 17 :315–324.

Ji, S., Vishwanathan, S. V. N., Satish, N., Anderson, M. J., and Dubey, P. (2015).

Blackout : Speeding up recurrent neural network language models with very large vocabularies.
CoRR, abs/1511.06909.

Jurafsky, D. and Martin, J. H. (2000).
Speech and Language Processing.
Prentice Hall.

Kim, Y., Jernite, Y., Sontag, D., and Rush, A. M. (2015).
Character-aware neural language models.
CoRR, abs/1508.06615.

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

Koehn, P. (2005).
Europarl : A Parallel Corpus for Statistical Machine Translation.
In *tenth Machine Translation Summit*, pages 79–86.

Kuhn, R. and Mori, R. D. (1990).

rali

A cache-based natural language model for speech reproduction.

IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-12(6) :570–583.

Le, H.-S., Allauzen, A., and Yvon, F. (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*, pages 1–10, Montréal, Canada.

Manning, C. D. and Schütze, H. (1999).
Foundations of Statistical Natural Language Processing.
MIT Press.

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., and Khudanpur, S. (2010).
Recurrent neural network based language model.

In *Proceedings of the 11th Annual Conference of the International Speech Communication Association, INTERSPEECH 2010*, volume 2, pages 1045–1048.

Ney, H., Essen, U., and Kneser, R. (1994).
On structuring probabilistic dependences in stochastic language modeling.
Computer, Speech, and Language, 8 :1–38.

Patwary, M. M. A., Chabbi, M., Jun, H., Huang, J., Diamos, G. F., and Church, K. (2018).
Language modeling at scale.
CoRR, abs/1810.10045.

Pauls, A. and Klein, D. (2011).
Faster and smaller n-gram language models.
In *Proceedings of the 49th ACL/HLT*, pages 258–267, Portland, Oregon, USA.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018).
Deep contextualized word representations.

In *Proc. of NAACL*.

Pickhardt, R., Gottron, T., Körner, M., Wagner, P. G., Speicher, T., and Staab, S. (2014).
A generalized language model as the combination of skipped n-grams and modified kneser-ney smoothing.

Puri, R., Kirby, R., Yakovenko, N., and Catanzaro, B. (2018).
Large scale language modeling : Converging on 40gb of text in four hours.
CoRR, abs/1808.01371.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2018).
Language models are unsupervised multitask learners.

Rosenfeld, R. (1997).
A whole sentence maximum entropy language model.
In *Proceedings of the IEEE Workshop on Speech Recognition and Understanding*.

rali

Rosenfeld, R. (2000).
Two decades of statistical language modeling : Where do we go from here.

Salton, G., Ross, R., and Kelleher, J. (2017).
Attentive language models.
In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1 : Long Papers)*, pages 441–450, Taipei, Taiwan.

Shareghi, E., Gerz, D., Vulić, I., and Korhonen, A. (2019).
Show some love to your n-grams : A bit of progress and stronger n-gram language modeling baselines.
In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4113–4118.

Shareghi, E., Petri, M., Haffari, G., and Cohn, T. (2016).

Fast, small and exact : Infinite-order language modelling with compressed suffix trees.
Transactions of the Association for Computational Linguistics, 4 :477–490.

Shen, X., Oualil, Y., Greenberg, C., Singh, M., and Klakow, D. (2017).
Estimation of gap between current language models and human performance.
In *INTERSPEECH 2017, Proceedings of the 18th Annual Conference of the International Speech Communication Association*, pages 553–557.

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

Tang, R. and Lin, J. (2018).
Progress and tradeoffs in neural language models.
CoRR, abs/1811.00942.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017).
Attention is all you need.
CoRR, abs/1706.03762.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J. G., Salakhutdinov, R., and Le, Q. V. (2019).
Xlnet : Generalized autoregressive pretraining for language understanding.
CoRR, abs/1906.08237.