

IFT 615 – Intelligence Artificielle

Application - Traitement du langage naturel : modèles du langage

Professeur: Froduald Kabanza

Assistants: D'Jeff Nkashama

Sujets couverts

- Le traitement du langage naturel est un domaine important en IA
- Nous avons déjà vu quelques concepts et applications
 - ◆ Représentation des mots par un réseau de neurones (*word embedding*)
 - ◆ Classification de documents par un classifieur bayésien
 - ◆ Étiquetage grammaticale par un réseau de neurones
 - ◆ Reconnaissance de la parole par un modèle de Markov caché
- Cette leçon couvre les modèles de langages
 - ◆ **Modèle *bag-of-words***
 - ◆ **Modèle *n-gram***

Applications du TLN

- Dialogue naturel
 - Traduction automatique
 - Complétion de texte
 - Analyse de sentiments
 - Extraction d'information
 - Automatisation robotisée de processus
 - Service à la clientèle
-
- Beaucoup de plateformes d'IA commerciales et gratuites ont des services ou bibliothèques de TLN: *AWS Sagemaker, MS Azure ML, Google AI, IBM Watson ML*








We had a wonderful meal at [REDACTED]
[REDACTED]. Great menu choices
and friendly personnel. Will go
back for sure!

Commentaire sur TripAdvisor



Modèles de langage

- Un langage est un ensemble de « mots ». Pour un langage naturel c'est l'ensemble de phrases (chaînes de mots) du langage.
- Le langage naturel étant ambiguë, on ne peut pas utiliser les modèles pour les langages de programmation comme par exemple les automates finis ou les grammaires hors-contextes.
- Un **modèle de langage (naturel)** est une distribution de probabilité décrivant la vraisemblance de n'importe quelle chaîne de mots.
 - ◆ Modèles basés sur les réseaux de neurones (ex: GPT, BERT)
 - ◆ Modèles basés sur les distributions sur les mots (ex: *bag-of-words*, *n-grams*)
 - ◆ Modèles basés sur les grammaires probabilistes (distributions sur les règles)
 - ◆ Modèles basés sur les chaînes de Markov (distributions sur les états)
- Cette leçon introduit les modèles *bag-of-words* et *n-grams*.

Le modèle *bag-of-words*

- Étant donné une chaîne de mots $w_{1:N}$, le modèle *bag-of-words* prédit la vraisemblance de la chaîne en ignorant l'ordre des mots.
- Le modèle *bag-of-words* est en fait une généralisation simple du classifieur bayésien naïf des documents.
- Le classifieur bayésien naïf des documents prédit qu'un document (plus précisément les mots clés apparaissant dans le document) est dans une classe donnée (sport, économie, etc.)

$$P(\text{Class} | \text{keyWord}_1, \dots, \text{keyWord}_n) = \alpha P(\text{Class}) \prod_{j=1..n} P(\text{keyWord}_j | \text{Class})$$

- Le **modèle *bag of words*** prédit que l'ensemble de mots de la séquence (pas seulement les mots clés de la séquence) est dans la classée donnée (sport, économie, etc.).

$$P(\text{Class} | w_{1:N}) = \alpha P(\text{Class}) \prod_{j=1..n} P(w_j | \text{Class})$$

Modèle *n-gram*

- Le modèle *bag of words* ignore l'ordre des mots

$$P(\text{Class} | w_{1:N}) = \alpha P(\text{Class}) \prod_{j=1..n} P(w_j | \text{Class})$$

- ◆ Les mots {quart, atteindre, but} sont fréquents dans les chroniques sport et économie. Mais la phrase “le quart arrière a atteint le but” est clairement plus probable dans la chronique sport. Le modèle *bag of words* ne classerait pas la phrase correctement.
- Le **modèle *n-gram*** est un modèle qui décrit la vraisemblance d'une chaîne de mots en faisant dépendre la probabilité de chaque mot des *n* dernier mots (hypothèse markovienne d'ordre *n*).

$$P(w_j | w_{1:j-1}) = P(w_j | w_{j-n+1:j-1})$$

$$P(w_{1:N}) = \prod_{j=1..n} P(w_j | w_{j-n+1:j-1})$$

- Donne de bons résultats dans plusieurs applications: classification de documents, classification de pourriels, analyse de sentiments et autres

Modèle *n-gram*

- Autres nom pour le modèle n-grammes
 - ◆ $n=1$: unigram (équivalent au modèle bayésien naïf)
 - ◆ $n=2$: bigram
 - ◆ $n=3$: trigram

Apprentissage de modèle n -gramme

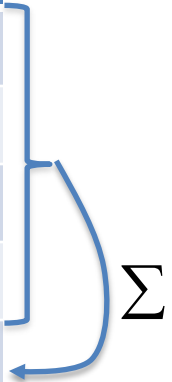
- On peut apprendre un modèle n -gramme à partir des fréquences de n -grammes dans un corpus de documents D

$$P(W_i = w \mid w_{i-n+1}, \dots, w_{i-1}) = \frac{\text{nb. de fois que } w \text{ suit les mots } w_{i-n+1}, \dots, w_{i-1}}{\text{nb. de fois que } w_{i-n+1}, \dots, w_{i-1} \text{ est suivi d'un mot}}$$

Apprentissage de modèle n-gramme

- Exemple: soit les fréquences totales suivantes

Tri-gramme	freq(<i>n</i> -gramme, <i>D</i>)
(« modèle », « de », « Bayes »)	5
(« modèle », « de », « Markov »)	25
(« modèle », « de », « langage »)	10
...	...
(« modèle », « de », *)	200



- Alors le modèle trigramme assignerait les probabilités:

$$P(W_i = \text{« Bayes »} \mid W_{i-2} = \text{« modèle »}, W_{i-1} = \text{« de »}) = 5 / 200$$

$$P(W_i = \text{« Markov »} \mid W_{i-2} = \text{« modèle »}, W_{i-1} = \text{« de »}) = 25 / 200$$

$$P(W_i = \text{« langage »} \mid W_{i-2} = \text{« modèle »}, W_{i-1} = \text{« de »}) = 10 / 200$$

...


...

Application des modèles n-gramme

- Identification de la langue
 - ◆ étant donné un document, identifier dans quelle langue (anglais, français, etc.) il est écrit
- On détermine d'abord un vocabulaire **commun** V pour toutes les langues
- Pour chaque langue l que l'on souhaite détecter
 - ◆ on collecte un corpus de documents dans cette langue
 - ◆ on assigne une probabilité a priori $P(L=l)$ de la langue
 - ◆ on apprend un modèle n -gramme $P(W_i=w \mid w_{i-n+1}, \dots, w_{i-1}, L=l)$ sur ce corpus
- Étant donné un nouveau document, on lui assigne la langue la plus probable

$$\begin{aligned} \operatorname{argmax} P(L=l \mid [w_1, \dots, w_d]) &= \operatorname{argmax} \log P(L=l, [w_1, \dots, w_d]) \\ &= \operatorname{argmax} \log P(L=l) + \sum_i \log P(W_i = w_i \mid w_{i-n+1}, \dots, w_{i-1}, L=l) \end{aligned}$$

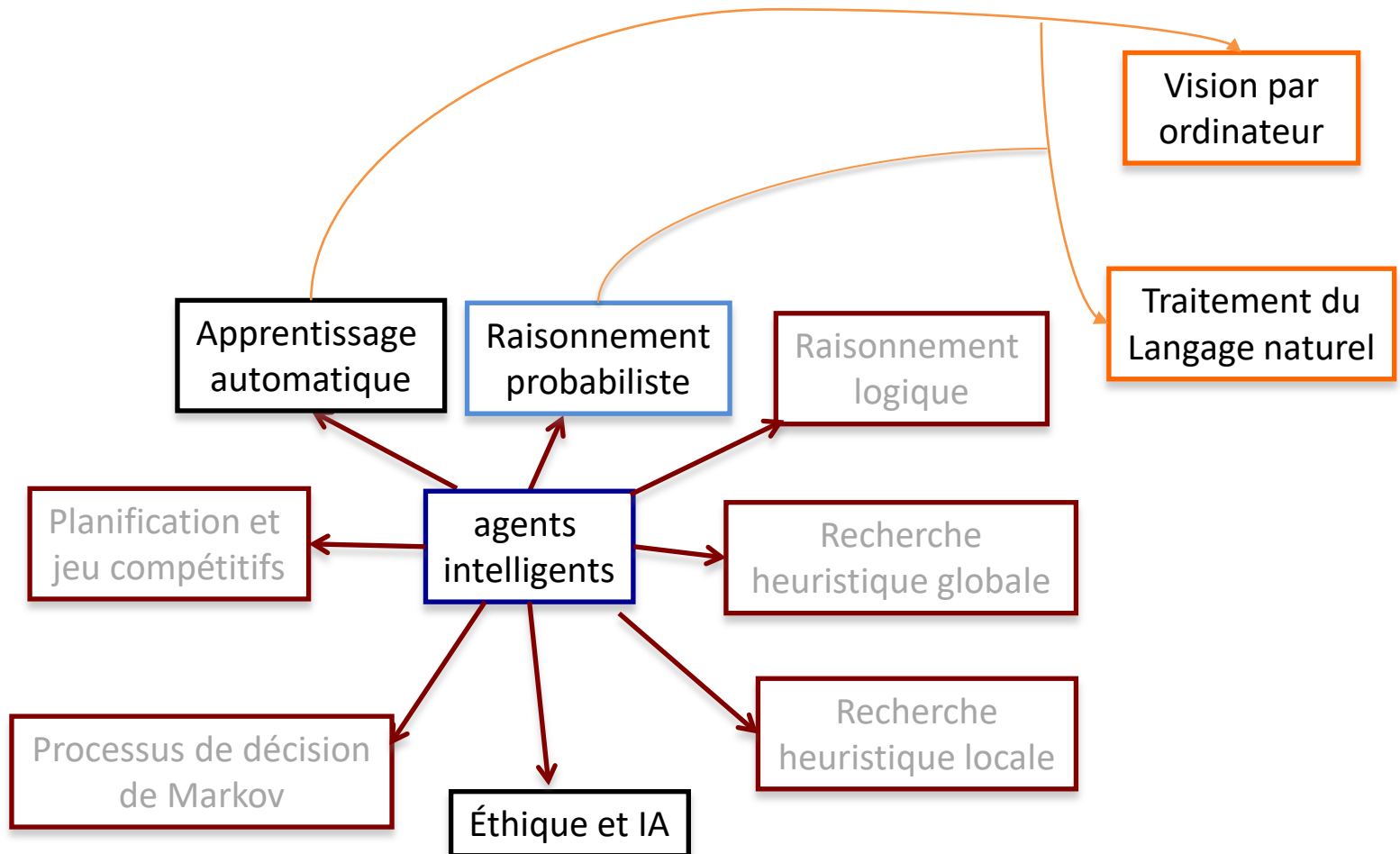
Génération de texte

- Nous avons vu qu'un réseau pour l'étiquetage syntaxique peut aussi générer du texte. C'est une façon d'évaluer la qualité du réseau.
- On peut faire la même chose avec le modèle n-gram.A small diagram titled 'RNN pour l'étiquetage syntaxique' showing a sequence of hidden states and input/output tokens for syntactic tagging.
- Exemples générés par des n-gram à partir du texte du livre du cours:
 - ◆ *n=1: logical are as are confusion a may right tries agent goal the was*
 - ◆ *n=2: systems are very similar computational approach would be represented*
 - ◆ *n=4: taking advantage of the structure of Bayesian networks and developed various languages for writting « templates » with logical variables ...*
- Exemple lorsqu'on ajoute la Bible (version King James) au livre :
 - ◆ *Prove that any 3-SAT problem can be reduced to simpler ones using the laws of thy God*

Sujets couverts par le cours

Concepts et algorithmes

Applications



Vous devriez être capable de...

- Modèle de langage
 - ◆ savoir ce qu'est un modèle de langage
 - ◆ savoir ce qu'est un modèle *bag-of-words*
 - ◆ savoir ce qu'est un modèle *n*-gram
 - ◆ savoir à quoi peut servir un modèle de langage

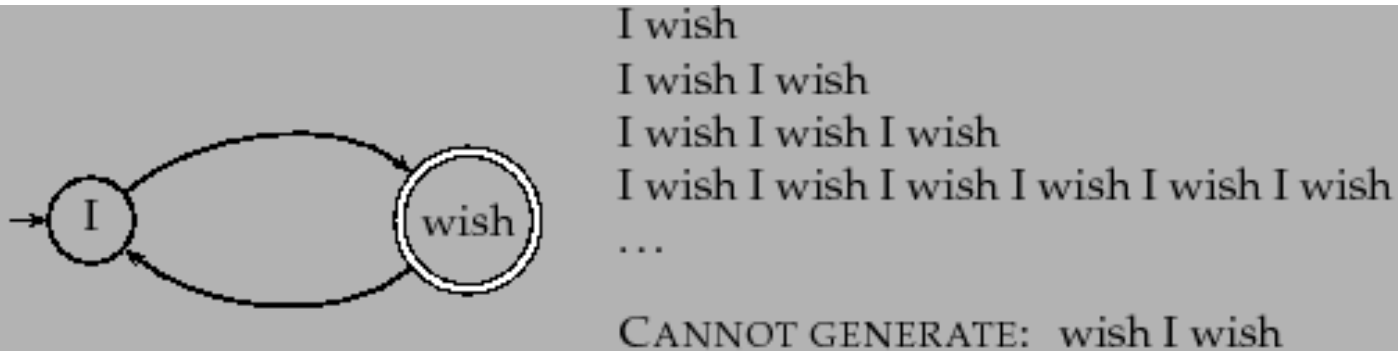
SLIDES RÉFÉRENCÉES

Certaines des plateformes d'IA commerciales les plus populaires

	Amazon ML and SageMaker	Microsoft Azure AI Platform	Google AI Platform (Unified)	IBM Watson Machine Learning
Classification	✓	✓	✓	✓
Regression	✓	✓	✓	✓
Clustering	✓	✓	✓	✗
Anomaly detection	✓	✓	✗	✗
Recommendation	✓	✓	✓	✗
Ranking	✓	✓	✗	✗
Data Labeling	✓	✓	✓	✓
MLOps pipeline support	✓	✓	✓	✓
Built-in algorithms	✓	✓	✓	✗
Supported frameworks	TensorFlow, MXNet, Keras, Gluon, Pytorch, Caffe2, Chainer, Torch	TensorFlow, scikit-learn, PyTorch, Microsoft Cognitive Toolkit, Spark ML	TensorFlow, scikit-learn, XGBoost, Keras	TensorFlow, Keras, Spark MLlib, scikit-learn, XGBoost, PyTorch, IBM SPSS, PMML

altexsoft
software r&d engineering

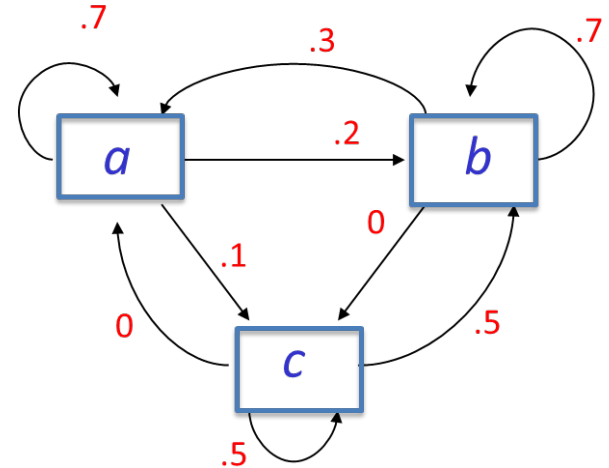
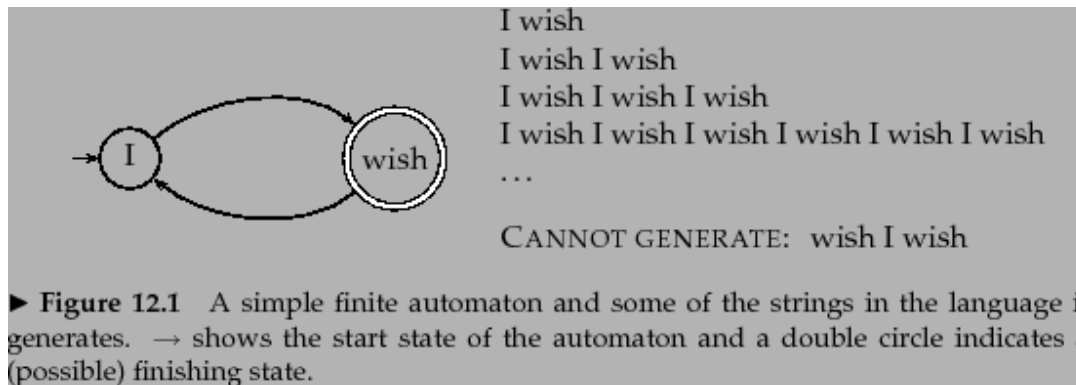
A finite automaton



► **Figure 12.1** A simple finite automaton and some of the strings in the language it generates. → shows the start state of the automaton and a double circle indicates a (possible) finishing state.

Source: [Manning et al. \(2008\) Introduction to IR](#)

Automate à états finis vs Chaîne de Markov



Source: [Manning et al. \(2008\) Introd to IR](#)

Grammaire Hors-Contexte

$S \rightarrow NP VP$	I + feel a breeze
$ S Conj S$	I feel a breeze + and + It stinks
$NP \rightarrow Pronoun$	I
$ Name$	Ali
$ Noun$	pits
$ Article Noun$	the + wumpus
$ Article Adjs Noun$	the + smelly dead + wumpus
$ Digit Digit$	3 4
$ NP PP$	the wumpus + in 1 3
$ NP RelClause$	the wumpus + that is smelly
$ NP Conj NP$	the wumpus + and + I
$VP \rightarrow Verb$	stinks
$ VP NP$	feel + a breeze
$ VP Adjective$	smells + dead
$ VP PP$	is + in 1 3
$ VP Adverb$	go + ahead
$Adjs \rightarrow Adjective$	smelly
$ Adjective Adjs$	smelly + dead
$PP \rightarrow Prep NP$	to + the east
$RelClause \rightarrow RelPro VP$	that + is smelly

Grammaire Hors-Contexte Probabiliste

S	$\rightarrow NP VP$	[0.90]	I + feel a breeze
	$S Conj S$	[0.10]	I feel a breeze + and + It stinks
NP	$\rightarrow Pronoun$	[0.25]	I
	$Name$	[0.10]	Ali
	$Noun$	[0.10]	pits
	$Article Noun$	[0.25]	the + wumpus
	$Article Adjs Noun$	[0.05]	the + smelly dead + wumpus
	$Digit Digit$	[0.05]	3 4
	$NP PP$	[0.10]	the wumpus + in 1 3
	$NP RelClause$	[0.05]	the wumpus + that is smelly
	$NP Conj NP$	[0.05]	the wumpus + and + I
VP	$\rightarrow Verb$	[0.40]	stinks
	$VP NP$	[0.35]	feel + a breeze
	$VP Adjective$	[0.05]	smells + dead
	$VP PP$	[0.10]	is + in 1 3
	$VP Adverb$	[0.10]	go + ahead
$Adjs$	$\rightarrow Adjective$	[0.80]	smelly
	$Adjective Adjs$	[0.20]	smelly + dead
PP	$\rightarrow Prep NP$	[1.00]	to + the east
$RelClause$	$\rightarrow RelPro VP$	[1.00]	that + is smelly

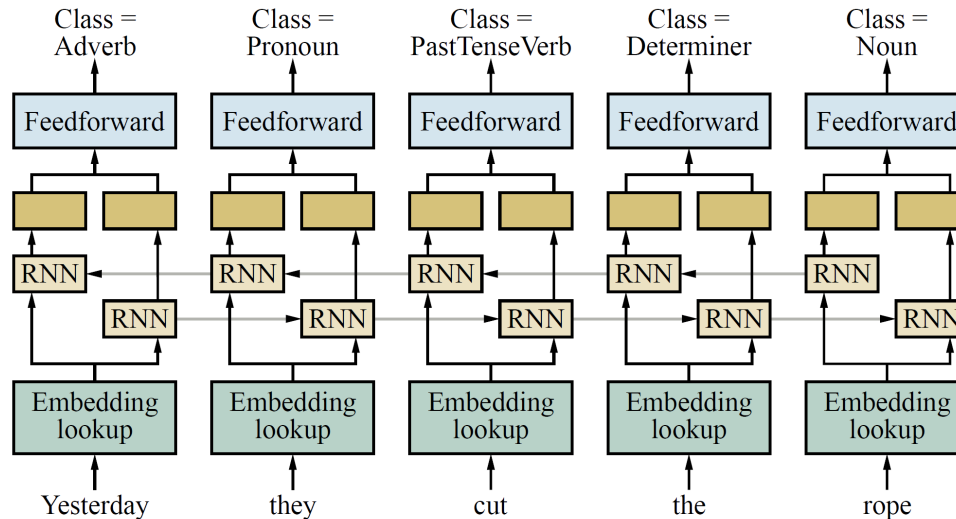
Lexique

<i>Noun</i>	→	stench [0.05] breeze [0.10] wumpus [0.15] pits [0.05] ...
<i>Verb</i>	→	is [0.10] feel [0.10] smells [0.10] stinks [0.05] ...
<i>Adjective</i>	→	right [0.10] dead [0.05] smelly [0.02] breezy [0.02] ...
<i>Adverb</i>	→	here [0.05] ahead [0.05] nearby [0.02] ...
<i>Pronoun</i>	→	me [0.10] you [0.03] I [0.10] it [0.10] ...
<i>RelPro</i>	→	that [0.40] which [0.15] who [0.20] whom [0.02] ...
<i>Name</i>	→	Ali [0.01] Bo [0.01] Boston [0.01] ...
<i>Article</i>	→	the [0.40] a [0.30] an [0.10] every [0.05] ...
<i>Prep</i>	→	to [0.20] in [0.10] on [0.05] near [0.10] ...
<i>Conj</i>	→	and [0.50] or [0.10] but [0.20] yet [0.02] ...
<i>Digit</i>	→	0 [0.20] 1 [0.20] 2 [0.20] 3 [0.20] 4 [0.20] ...

Modèle de langage

- Un tel modèle me permettrait de calculer la probabilité qu'une chaîne de mots soit dans le langage :
 - ◆ je suis dans le cours IFT615 -> probabilité très élevée
 - ◆ Je suis dans IFT615 cours -> probabilité élevée
 - ◆ cours je -> probabilité très basse
 - ◆ Je oursc -> probabilité extrêmement faible

RNN pour l'étiquetage syntaxique



Une fois entraîné, le modèle peut générer du texte.

*Mary, and will, my lord, to weep in such a one were prettiest
Yet now I was adopted heir
Of the world's lamentable day
To watch the next way with his father with his face?*

C'est une façon d'évaluer la qualité du modèle. Plus le modèle est bon, plus il génère des textes vraisemblables (GPT et BERT sont des modèles du langage très connus basés sur l'architecture *Transformer* non couvert dans ce cours)

**LA PARTIE SUIVANTE N'EST PAS
COUVERTE PAR L'EXAMEN**

Lissage de modèle n-gramme


- On peut également lisser les modèles n -gramme en général
 - ◆ encore plus important, puisque plus un n -gramme est long, moins il sera fréquent
 - ◆ la plupart des n -grammes imaginable auront une fréquence de zéro, pour n grand
- Première approche: **lissage δ**

$$P(W_i = w \mid w_{i-n+1}, \dots, w_{i-1}) = \frac{\delta + \sum_t \text{freq}(w_{i-n+1}, \dots, w_{i-1}, w), D_t)}{\delta (|V| + 1) + \sum_t \text{freq}(w_{i-n+1}, \dots, w_{i-1}, *), D_t)}$$

Lissage δ

- Exemple: soit les fréquences totales suivantes

n -gramme	freq(n -gramme, D)
(« modèle », « de », « Bayes »)	5
(« modèle », « de », « langage »)	10
(« modèle », « de », « langue »)	0
...	...
(« modèle », « de », *)	200



- Trigramme avec lissage $\delta = 0.1$ et un vocabulaire de taille $|V|=999$

$$P(W_i = \text{« Bayes »} \mid W_{i-2} = \text{« modèle »}, W_{i-1} = \text{« de »}) = (0.1+5) / (100+200) = 5.1 / 300$$

$$P(W_i = \text{« langage »} \mid W_{i-2} = \text{« modèle »}, W_{i-1} = \text{« de »}) = (0.1+10) / (100+200) = 10.1 / 300$$

$$P(W_i = \text{« langue »} \mid W_{i-2} = \text{« modèle »}, W_{i-1} = \text{« de »}) = (0.1+0) / (100+200) = 0.1 / 300$$

...

...

Lissage par interpolation linéaire

- Deuxième approche: **lissage par interpolation linéaire**
 - ◆ faire la moyenne (pondérée) de modèles unigrammes, bigrammes, trigrammes, ... jusqu'à n -gramme

$$P_{\lambda}(W_i = w \mid w_{i-n+1}, \dots, w_{i-1}) = \lambda_1 P(W_i = w) + \\ \lambda_2 P(W_i = w \mid w_{i-1}) + \\ \lambda_3 P(W_i = w \mid w_{i-2}, w_{i-1}) + \dots + \\ \lambda_n P(W_i = w \mid w_{i-n+1}, \dots, w_{i-1})$$

où $\sum_i \lambda_i = 1$

- Exemple:
 - ◆ le trigramme (« modèle », « de », « langue ») a une fréquence de 0
 - ◆ le bigramme (« de », « langue ») est présent dans le corpus
 - ◆ alors $P_{\lambda}(W_i = w \mid w_{i-n+1}, \dots, w_{i-1}) > 0$, en autant que λ_2 ou $\lambda_1 > 0$