

IFT 615 – Intelligence artificielle

Traitement automatique de la langue naturelle

Hugo Larochelle

Département d'informatique

Université de Sherbrooke

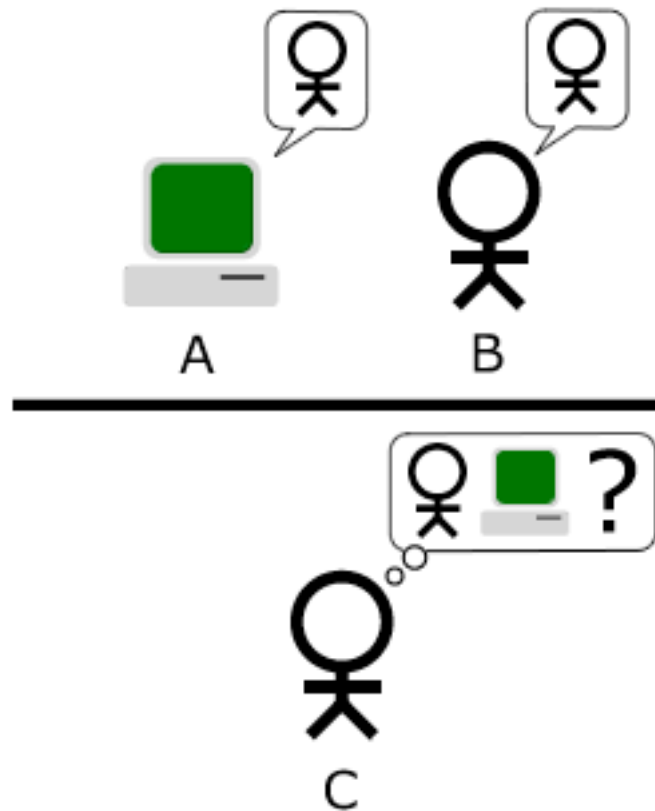
<http://www.dmi.usherb.ca/~larocheh/cours/ift615.html>

Sujets couverts

- Classification de documents
- Modèles de langage
- Étiquetage syntaxique
- Extraction d'information

Mise en situation

- Traitement automatique de la langue naturelle (TALN)
 - ◆ une composante fondamentale du test de Turing...



Mise en situation

- Traitement automatique de la langue (TALN)
 - ◆ ... et des tonnes d'applications!

Recherche d'information



Environ 1 910 000 résultats (0,19 secondes)

Essayez avec cette orthographe : [ift615](#)

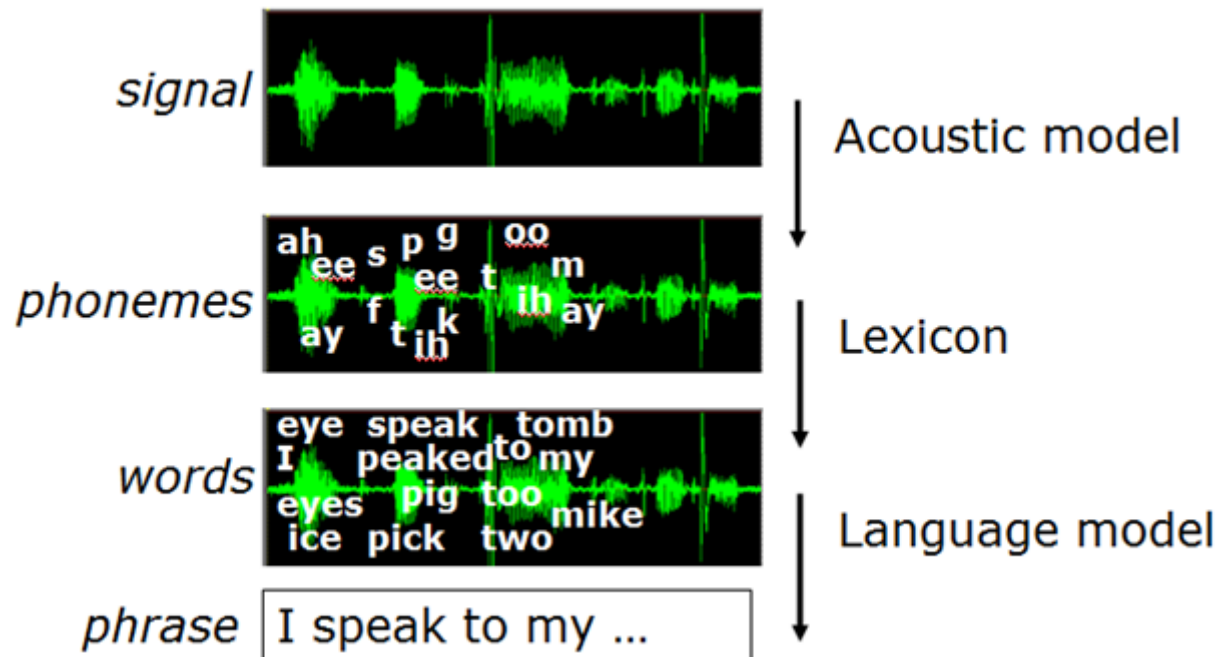
[IFT615 - Intelligence artificielle](#)
planart.usherbrooke.ca/kabanza/cours/ift615/ift615.html
IFT615 Hiver 2011. Liens utiles. Fiche signalétique · Code source des exemples donnés en classe. Vidéos et animations utilisées en classe. Anciens examens ...

[\[PPT\] IFT615 - Satisfaction de contraintes](#)
planart.usherbrooke.ca/.../ift615/ift615-satisfaction-contraintes.ppt
Format de fichier: Microsoft Powerpoint - [Afficher](#)
IFT 615 – Intelligence Artificielle ... **IFT615**. Sujets couverts. Modèle général des problèmes de satisfaction de contraintes; Algorithmes Backtracking et search.

Mise en situation

- Traitement automatique de la langue (TALN)
 - ◆ ... et des tonnes d'applications!

Reconnaissance de la parole



Mise en situation

- Traitement automatique de la langue (TALN)
 - ◆ ... et des tonnes d'applications!

Traduction automatique

Traduction

Source : français ▼ ↔ Cible : anglais ▼ Traduire

français anglais arabe

Se familiariser avec les fondements de l'intelligence artificielle. Connaître les possibilités et les limites des techniques utilisées en intelligence artificielle. Savoir choisir et appliquer les différentes approches en fonction du problème à résoudre.

anglais français arabe

Become familiar with the foundations of artificial intelligence. Know the capabilities and limitations of the techniques used in artificial intelligence. Learn to select and apply different approaches to the problem to solve.

Mise en situation

- Traitement automatique de la langue (TALN)
 - ◆ ... et des tonnes d'applications!

Système de réponse automatique



<http://www.youtube.com/watch?v=yJptrlCVDHI&feature=related>

Dans ce cours...

- On va se concentrer sur des tâches « simples »
 - ◆ classification de documents
 - ◆ modélisation de langage
 - ◆ étiquetage syntaxique
 - ◆ extraction d'information
- Ces tâches sont souvent des outils utilisés dans des systèmes plus grands et plus complexes de TALN

Définitions

- Document: une liste de mots
 - ◆ pourrait être tout un texte
 - ◆ pourrait être une seule phrase
 - ◆ pourrait être quelques mots
- Mots: un mot ou une ponctuation
 - ◆ on suppose que nos documents ont déjà été segmentés en mots
 - ◆ généralement facile à faire en anglais (on sépare en fonction des espaces et des ponctuations)
 - ◆ difficile en chinois ou en japonais (pas d'espaces entre les mots)

Classification de documents

- Soit les deux documents (question d'examen) suivants:

« Dessinez la partie de l'espace d'états qui serait explorée par l'algorithme alpha-beta pruning, en supposant qu'il explore l'espace d'états de la gauche vers la droite. »

« En utilisant l'algorithme d'apprentissage du perceptron et un pas d'apprentissage de 0.3, donnez la sortie et les poids des connexions à la fin de la deuxième itération. »

- Laquelle est une question d'examen final, en IFT 615?

Classification de documents

- Soit les deux documents (question d'examen) suivants:

« d'états d'états de qui explore
qu'il explorée gauche
l'algorithme pruning, l'espace
par en Dessinez alpha-beta
droite. la la supposant l'espace
partie serait la de vers »

« un pas de l'algorithme fin
sortie de perceptron donnez la
deuxième En à poids du et et
des d'apprentissage connexions
les itération. la la
d'apprentissage utilisant 0.3, »

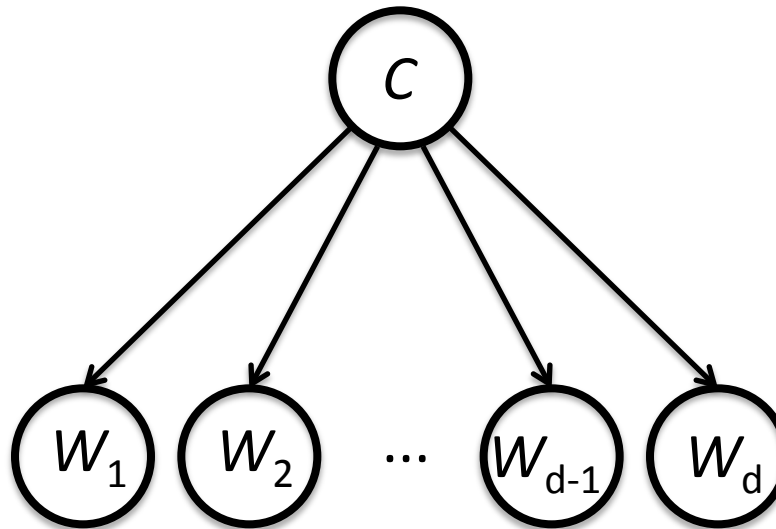
- Laquelle est une question d'examen final, en IFT 615?

Classification de documents

- Les mots individuels sont très informatifs du sujet (catégorie) d'un document
- L'ordre des mots n'est souvent pas utile
 - ◆ l'ordre reflète surtout la syntaxe d'une langue
 - ◆ on suppose que la catégorie n'influence que la probabilité d'observer un mot dans un document
- Ignorer l'ordre des mots va permettre de simplifier le système, sans trop compromettre sa précision
- On va formaliser ces hypothèses à l'aide d'un **réseau bayésien**

Modèle bayésien naïf multinomial

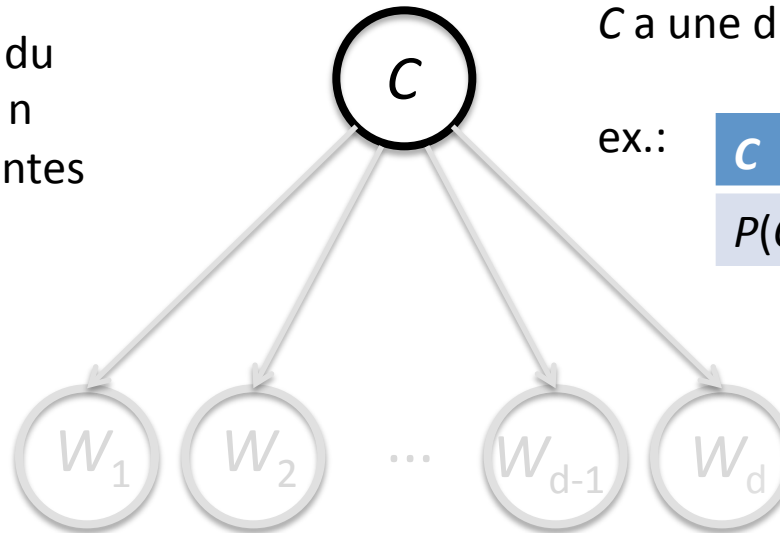
- Réseau bayésien: **modèle bayésien naïf multinomial**



Modèle bayésien naïf multinomial

- Réseau bayésien: **modèle bayésien naïf multinomial**

C est la catégorie du document, parmi n catégories différentes



C a une distribution a priori

ex.:

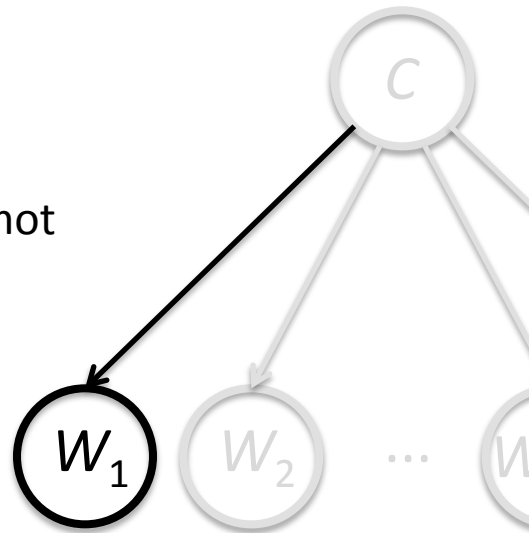
C	<i>intra</i>	<i>final</i>
$P(C)$	0.5	0.5

somme à 1

Modèle bayésien naïf multinomial

- Réseau bayésien: **modèle bayésien naïf multinomial**

W_1 est le premier mot d'un document, contenant d mots



W_1 a une distribution conditionnelle multinomiale

C	<i>intra</i>	<i>final</i>
$P(W_1=\text{« de »} C)$	0.01	0.01
$P(W_1=\text{« qui »} C)$	0.02	0.02
...
$P(W_1=\text{« perceptron »} C)$	10^{-6}	0.002

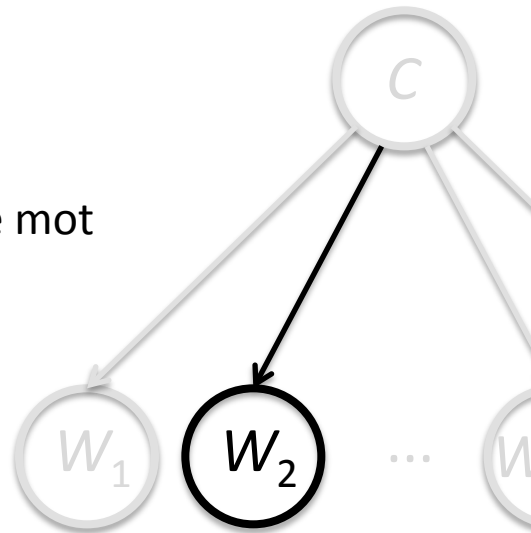
somme à 1

somme à 1

Modèle bayésien naïf multinomial

- Réseau bayésien: **modèle bayésien naïf multinomial**

W_2 est le deuxième mot
d'un document,
contenant d mots



W_2 a **la même** une distribution
conditionnelle multinomiale

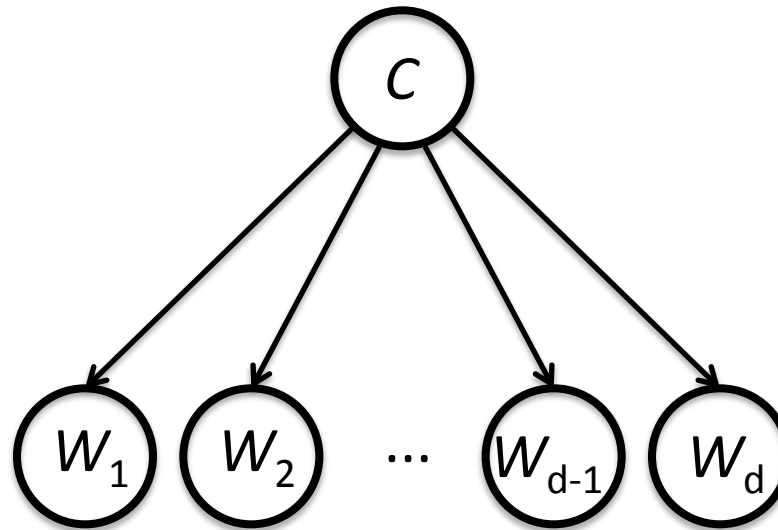
C	<i>intra</i>	<i>final</i>
$P(W_2=\text{« de »} C)$	0.01	0.01
$P(W_2=\text{« qui »} C)$	0.02	0.02
...
$P(W_2=\text{« perceptron »} C)$	10^{-6}	0.002

somme à 1

somme à 1

Modèle bayésien naïf multinomial

- Réseau bayésien: **modèle bayésien naïf multinomial**



- En général la **probabilité conjointe** d'un document $[W_1, \dots, W_d]$ ayant d mots et de sa catégorie C :

$$\mathbf{P}([W_1, \dots, W_d], C) = \mathbf{P}(C) \prod_i \mathbf{P}(W_i \mid C)$$

Modèle bayésien naïf multinomial

- Exemple:

<i>C</i>	<i>intra</i>	<i>final</i>
$P(C)$	0.5	0.5

<i>C</i>	<i>intra</i>	<i>final</i>
$P(W_i = \text{« , »} C)$	0.01	0.01
$P(W_i = \text{« un »} C)$	0.02	0.02
$P(W_i = \text{« d' »} C)$	0.01	0.02
$P(W_i = \text{« Perceptron »} C)$	10^{-6}	0.002
$P(W_i = \text{« algorithme »} C)$	0.005	0.005
$P(W_i = \text{« apprentissage »} C)$	10^{-5}	0.001
$P(W_i = \text{« . »} C)$	0.03	0.03
...

$$P(\text{« Perceptron, un algorithme d'apprentissage. »}, C = \textit{intra}) = 0.5 * 10^{-6} * 0.01 * 0.02 * 0.005 * 0.01 * 10^{-5} * 0.03 = 1.5 * 10^{-21}$$

$$P(\text{« Perceptron, un algorithme d'apprentissage. »}, C = \textit{final}) = 0.5 * 0.002 * 0.01 * 0.02 * 0.005 * 0.02 * 0.001 * 0.03 = 6 * 10^{-16}$$

Décision de la catégorie d'un document

- Pour classifier un document contenant les mots $[w_1, \dots, w_d]$, on choisit la classe c ayant la plus grande **probabilité a posteriori** $P(C=c | [w_1, \dots, w_d])$

$$\begin{aligned} & \operatorname{argmax}_c P(C=c | [w_1, \dots, w_d]) \\ &= \operatorname{argmax}_c \alpha P(C=c, [w_1, \dots, w_d]) \\ &= \operatorname{argmax}_c P(C=c, [w_1, \dots, w_d]) \quad \leftarrow \text{pour simplifier les calculs} \\ &= \operatorname{argmax}_c \log P(C=c, [w_1, \dots, w_d]) \\ &= \operatorname{argmax}_c \log P(C=c) \prod_i P(W_i = w_i | C=c) \\ &= \operatorname{argmax}_c \log P(C=c) + \sum_i \log P(W_i = w_i | C=c) \quad \leftarrow \text{pour éviter le « underflow »} \end{aligned}$$

Décision de la catégorie d'un document

- Pour classifier un document fait des mots $[w_1, \dots, w_d]$, on choisit la classe c ayant la plus grande **probabilité a posteriori** $P(C=c | [w_1, \dots, w_d])$
- Exemple:

$$\operatorname{argmax} \log P(C=c) + \sum_i \log P(W_i = w_i \mid C=c)$$

$$= \operatorname{argmax} \left\{ \begin{array}{l} \log(0.5) + \log(10^{-6}) + \log(0.01) + \log(0.02) + \log(0.005) + \log(0.01) + \log(10^{-5}) + \log(0.03), \\ \log(0.5) + \log(0.002) + \log(0.01) + \log(0.02) + \log(0.005) + \log(0.02) + \log(0.001) + \log(0.03) \end{array} \right\}$$

$C = final$

$= final$

$C = intra$

Apprentissage du modèle

- Comment obtient-on les distributions $\mathbf{P}(C)$ et $\mathbf{P}(W_i | C)$?
 - ◆ on les obtient à partir de vraies données
 - ◆ on choisit $\mathbf{P}(C)$ et $\mathbf{P}(W_i | C)$ pour quelles reflètent les statistiques de ces données
- Soit un **corpus**, c.-à-d. un ensemble de T documents $\{D_t, C_t\}$
 - ◆ chaque document D_t est une liste de mots $[w_1^t, \dots, w_d^t]$ de taille variable
 - ◆ C_t est la catégorie de D_t

$$P(C=c) = (\text{nb. de documents de la catégorie } c) / (\text{nb. de documents total}) \\ = |\{t \mid C_t = c\}| / T$$

$$P(W_i = w \mid C=c) = \frac{\text{nb. de fois que } w \text{ apparaît dans les documents de la catégorie } c}{\text{nb. de mots total dans les documents de la catégorie } c} \\ = \frac{\sum_{t \mid C_t=c} \text{freq}(w, D_t)}{\sum_{t \mid C_t=c} |D_t|}$$

Lissage du modèle

- Selon la formule pour $P(W_i = w \mid C=c)$, un mot w aura une probabilité de 0 s'il n'apparaît jamais dans notre corpus
- Si un seul des $P(W_i = w \mid C=c) = 0$, alors tout $P(C=c, [w_1, \dots, w_d]) = 0!$
 - ◆ les mots rares vont beaucoup faire varier $P(C=c, [w_1, \dots, w_d])$ en général
- Pour éviter cette instabilité, deux trucs afin de **lisser la distribution $P(w \mid c)$**
 - ◆ on détermine un **vocabulaire** V de taille fixe, et on associe les mots qui ne sont pas dans ce vocabulaire au **symbole OOV** (*out of vocabulary*)
 - ◆ **lissage δ** : on ajoute une constante δ au numérateur, pour chaque mot

$$P(W_i = w \mid C=c) = \frac{\delta + \sum_{t \mid C_t=c} \text{freq}(w, D_t)}{\delta (|V|+1) + \sum_{t \mid C_t=c} |D_t|}$$

Lissage du modèle

- Exemple: soit le vocabulaire
 $V = \{ \text{« Perceptron »}, \text{« , »}, \text{« un »}, \text{« apprentissage »} \}$

- La phrase

« Perceptron, un algorithme d'apprentissage. »

sera représentée par la liste de mots

[« Perceptron », « , », « un », « OOV », « OOV », « apprentissage », « OOV »]

w_1 w_2 w_3 w_4 w_5 w_6 w_7

- Les statistiques sont calculées à partir de cette représentation
 - ◆ on pourrait aussi enlever les mots « OOV » et les ignorer

Prétraitement des données

- Si, parmi tous les intra des années dernières (corpus de 426 mots)
 - ◆ « Perceptron » apparaît 0 fois
 - ◆ « , » apparaît 15 fois
 - ◆ « un » apparaît 10 fois
 - ◆ « apprentissage » apparaît 1 fois
 - ◆ « OOV » (tous les autres mots) apparaissent 400 fois
- Si on utilisait $\delta = 1$, alors
 - ◆ $P(\text{« Perceptron »} \mid C=\textit{intra}) = (1 + 0) / (1 (4+1) + 426) = 1 / 431$
 - ◆ $P(\text{« , »} \mid C=\textit{intra}) = (1 + 15) / (1 (4+1) + 426) = 16 / 431$
 - ◆ $P(\text{« un »} \mid C=\textit{intra}) = (1 + 10) / (1 (4+1) + 426) = 11 / 431$
 - ◆ $P(\text{« apprentissage »} \mid C=\textit{intra}) = (1 + 1) / (1 (4+1) + 426) = 2 / 431$
 - ◆ $P(\text{« OOV »} \mid C=\textit{intra}) = (1 + 400) / (1 (4+1) + 426) = 401 / 431$

somme à 1



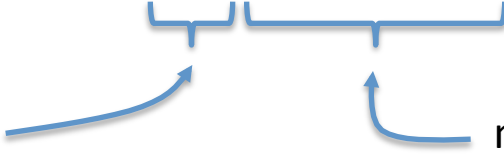
Prétraitement des données

- Comment choisir V
 - ◆ ne garder que **les mots les plus fréquents** (ex.: apparaissent au moins 10 fois)
 - ◆ **ne pas garder les mots trop communs**
 - » ne pas inclure la ponctuation
 - » ne pas inclure les déterminants (« un », « des », etc.)
 - » ne pas inclure les conjonction (« mais », « ou », etc.)
 - » ne pas inclure les pronoms (« je », « tu », etc.)
 - » ne pas inclure les verbes communs (« être », « avoir », « faire », etc.)
 - » etc.
 - ◆ utiliser une **forme normalisée des mots** (fusion de mots différents en un seul)
 - » **enlever les majuscules** (« Perceptron » → « perceptron »)
 - » **lemmatiser** les mots (« marchons » → « marcher »,
« suis » → « être », « est » → « être »)
- Il n'y a pas de recette universelle, le meilleur choix de V varie d'une application à l'autre

Modèle de langage

- Dans le modèle de bayes naïf multinomial, on peut distinguer deux parties

$$\mathbf{P}([W_1, \dots, W_d], C) = \mathbf{P}(C) \prod_i \mathbf{P}(W_i | C)$$

modèle des catégories  modèle de langage

- Un **modèle de langage** est une **distribution sur du texte**, c.-à-d. sur des séquences de mots
 - ◆ étant donné un texte $[w_1, \dots, w_d]$, lui assigne une probabilité $P([w_1, \dots, w_d])$
- Dans modèle de bayes naïf multinomial, le modèle de langage est très (trop?) simple
 - ◆ les mots sont générés indépendamment les uns des autres (étant donnée la catégorie C)

Modèle n-gramme

- Un meilleur modèle générerait le i^{e} mot d'une phrase au moins à partir des quelques mots précédents dans la phrase

$$P([W_1, \dots, W_d]) = \prod_i P(W_i \mid \underbrace{W_{i-n+1}, \dots, W_{i-1}}_{n-1 \text{ mots précédents}})$$

- On appelle de tels modèles de langage des **modèles n -gramme**
 - ◆ un **n -gramme** est une **sous-séquence de n mots**, extraite d'un corpus
 - ◆ on les appelle modèles n -gramme parce qu'ils **sont estimés à partir des fréquences de tous les n -grammes** d'un corpus
- Ces modèles sont en fait des **modèles (chaînes) de Markov d'ordre n**

Modèle n-gramme

- Exemple: dans le document

« Perceptron , un OOV OOV apprentissage OOV »

il y a:

- ◆ 7 **unigrammes** ($n=1$) dont 5 différents

« Perceptron »

« , »

« un »

« OOV »

« OOV »

« apprentissage »

« OOV »

Modèle n-gramme

- Exemple: dans le document

« Perceptron , un OOV OOV apprentissage OOV »

il y a:

- ◆ 6 **bigrammes** ($n=2$), tous différents

(« Perceptron », « , »)

(« , », « un »)

(« un », « OOV »)

(« OOV », « OOV »)

(« OOV », « apprentissage »)

(« apprentissage », « OOV »)

Modèle n-gramme

- Exemple: dans le document

« Perceptron , un OOV OOV apprentissage OOV »

il y a:

- ◆ 5 **trigrammes** ($n=3$), tous différents

(« Perceptron », « , », « un »)

(« , », « un », « OOV »)

(« un », « OOV », « OOV »)

(« OOV », « OOV », « apprentissage »)

(« OOV », « apprentissage », « OOV »)

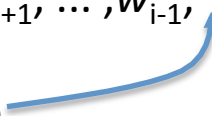
- ◆ etc.

- Tendence historique des n -grammes: <http://books.google.com/ngrams>

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_t

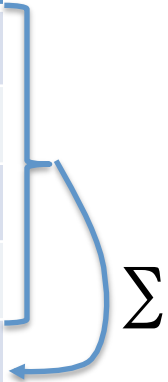
$$P(W_i = w \mid w_{i-n+1}, \dots, w_{i-1}) = \frac{\text{nb. de fois que } w \text{ suit les mots } w_{i-n+1}, \dots, w_{i-1}}{\text{nb. de fois que } w_{i-n+1}, \dots, w_{i-1} \text{ est suivi d'un mot}}$$
$$= \frac{\sum_t \text{freq}(w_{i-n+1}, \dots, w_{i-1}, w), D_t)}{\sum_t \text{freq}(w_{i-n+1},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

mot quelconque 

Apprentissage de modèle n-gramme

- Exemple: soit les fréquences totales suivantes

<i>n</i> -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:

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

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

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

...

...

Lissage de modèle n-gramme

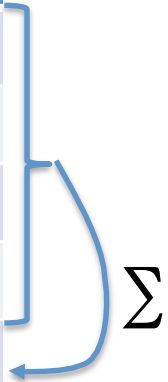
- On peut également lisser les modèles n -gramme en général
 - ◆ encore plus important, puisque plus un n -gramme est long, moins il sera fréquent
 - ◆ la plupart des n -grammes imaginable auront une fréquence de zéro, pour n grand
- Première approche: **lissage δ**

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

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 $\lambda_1 > 0$

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

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

Application des modèles n-gramme

- Classification de documents plus puissante
 - ◆ l'identification de la langue peut être vue comme de la classification de documents
 - ◆ équivaut à remplacer le modèle unigramme du modèle de bayes naïf par un modèle de langage possiblement plus puissant
 - ◆ nécessaire si l'ordre des mots est important (« maison blanche » vs. « blanche maison »)
- Et plusieurs autres
 - ◆ réaccentuation de texte (« modele bayesien » → « modèle bayésien »)
 - ◆ traduction automatique
 - ◆ reconnaissance de la parole

Évaluation d'un modèle de langage

- Afin de choisir n , δ ou les λ_i (des hyper-paramètres) on a besoin de définir une notion de performance
 - ◆ on choisirait les valeurs qui optimisent cette performance sur un **corpus de validation**, autre que le corpus d'entraînement et de test
- Si on sait dans quel système sera utilisé le modèle de langage, on utilise la performance de ce système
 - ◆ ex.: taux de succès d'un système de classification de documents
- Sinon, on peut calculer la **perplexité** (perplexité basse= bonne performance)

$$\begin{aligned}\text{Perp}([w_1, \dots, w_d]) &= (P([w_1, \dots, w_d]))^{-1/d} = \prod_i (P(W_i = w_i \mid w_{i-n+1}, \dots, w_{i-1}))^{-1/d} \\ &= \exp((-1/d) \sum_i \log P(W_i = w_i \mid w_{i-n+1}, \dots, w_{i-1}))\end{aligned}$$

Échantillonner d'un modèle n-gramme

- Pour avoir une idée de la qualité d'un modèle de langage appris, on peut aussi échantillonner de nouveaux documents
 - ◆ on laisse la machine parler d'elle-même
- Voici des échantillons de modèles unigramme, bigramme et trigramme, appris à partir du livre de référence

unigramme: « logical are as are confusion a may right tries agent goal the was... »

bigramme: « systems are very similar computational approach would be represented... »

trigramme: « planning and scheduling are integrated the success of naive bayes model is... »

Étiquetage syntaxique

- En plus de l'identité des mots, il peut être utile de connaître l'**étiquette syntaxique** de chacun de ces mots
 - ◆ « une visite à la ferme » → « ferme » est un nom
 - ◆ « Jean ferme la porte » → « ferme » est un verbe
- Connaître la catégorie grammaticale d'un mot peut faciliter une autre tâche
 - ◆ ex.: traduction automatique
 - » si « ferme » est un nom → « *farm* »
 - » si « ferme » est un verbe → « *close* »

Étiquetage syntaxique

- On suppose qu'on a accès à T **corpus étiquetés** D_t
(pour simplifier: un corpus = une phrase)

w_t	e_t
Jean	Nom
ferme	Verbe
la	Article
porte	Nom
.	.

$$D_t = [(w_1^t, e_1^t), \dots, (w_d^t, e_d^t)]$$

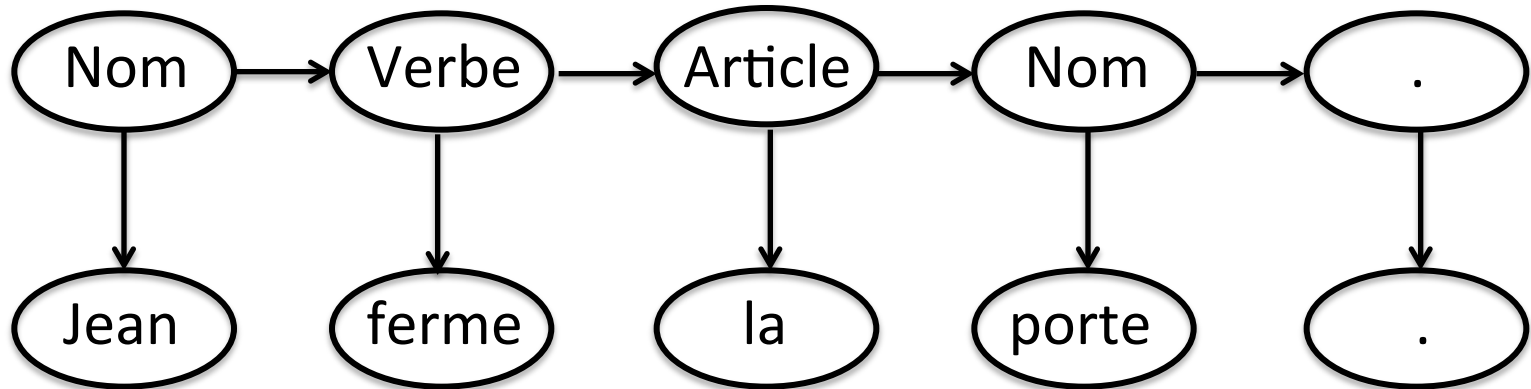
$$\text{mots}(D_t) = [w_1^t, \dots, w_d^t]$$

$$\text{étiquettes}(D_t) = [e_1^t, \dots, e_d^t]$$

- On pourrait prendre une approche similaire à la classification de documents
 - ◆ définir un réseau bayésien sur les mots et les étiquettes
 - ◆ apprendre le réseau sur notre corpus étiqueté
 - ◆ pour faire des prédictions, faire de l'inférence dans le réseau bayésien

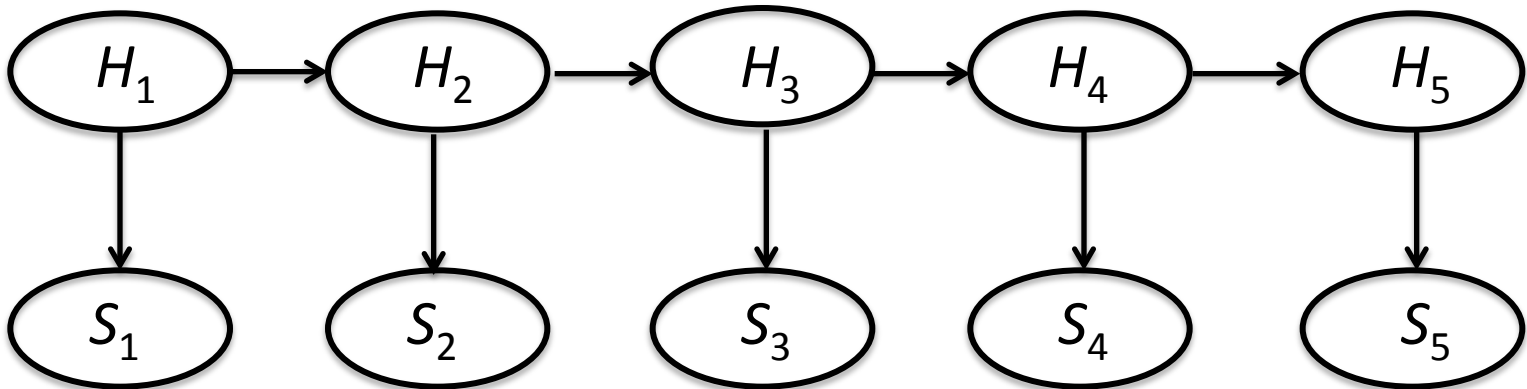
Étiquetage syntaxique par HMM

- On va utiliser un modèle de Markov caché (HMM)



Étiquetage syntaxique par HMM

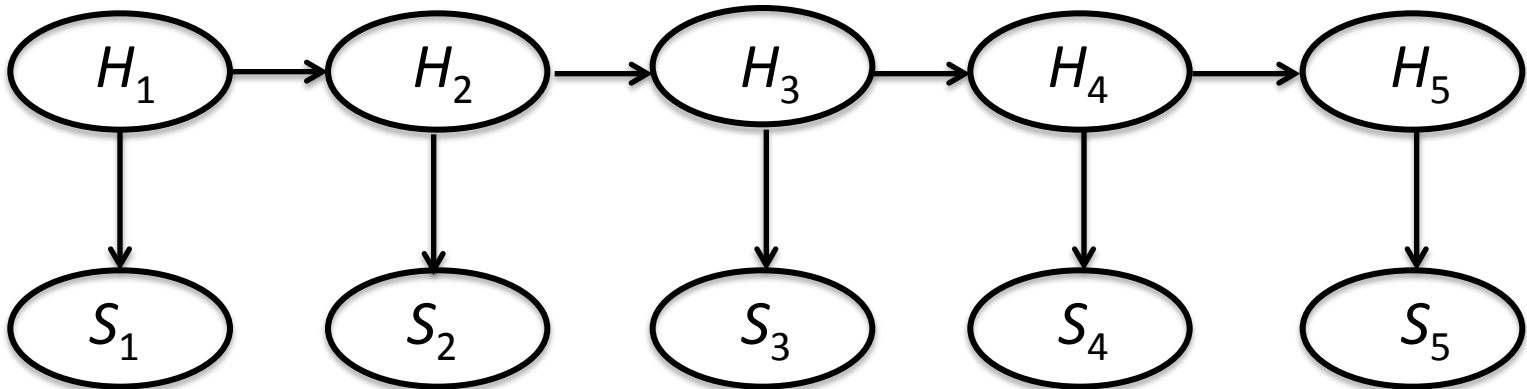
- On va utiliser un modèle de Markov caché (HMM)



- De notre corpus d'entraînement, on peut extraire des statistiques
 - ◆ sur la première étiquette d'une phrase ($P(H_1)$)
 - ◆ sur la relation entre un mot et sa classe syntaxique ($P(S_k | H_k)$)
 - » ex.: « ferme » peut être un nom, un verbe, mais pas un article
 - ◆ sur la relation entre les étiquettes syntaxiques adjacentes ($P(H_{k+1} | H_k)$)
 - » ex.: on ne peut avoir deux articles qui se suivent

Étiquetage syntaxique par HMM

- On apprend le HMM à partir de ces statistiques (fréquences)

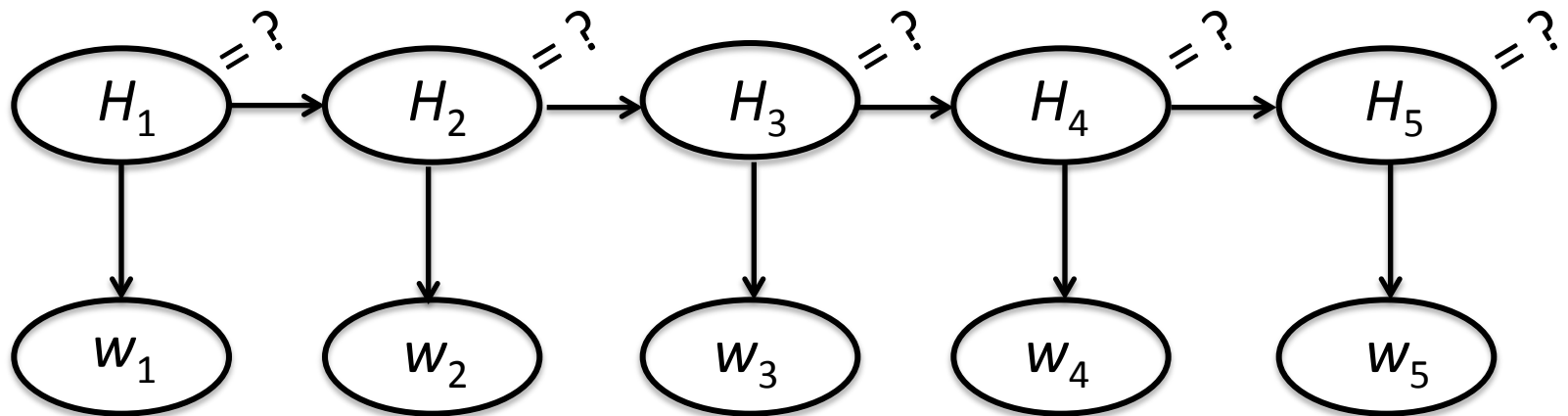


$$P(H_{k+1} = a \mid H_k = b) = \frac{\sum_t \text{freq}((b, a), \text{étiquettes}(D_t))}{\sum_t \text{freq}((b, *), \text{étiquettes}(D_t))} \quad P(S_k = w \mid H_k = b) = \frac{\sum_t \text{freq}((w, b), D_t)}{\sum_t \text{freq}(*, b, D_t)}$$

$$P(H_1 = a) = \frac{\sum_t \text{freq}(e_1^t = a, D_t)}{T}$$

Étiquetage syntaxique par HMM

- Pour étiqueter une nouvelle phrase $[w_1, \dots, w_d]$



- On calcule l'explication la plus plausible $h^*_{1:d}$
 - ◆ c.-à-d. $h^*_{1:d}$ qui maximise $P(S_{1:d} = [w_1, \dots, w_d], H_{1:d} = h^*_{1:d})$
 - ◆ on utilise le programme dynamique de α^* (cours sur réseaux bayésiens dynamiques)

Extraction d'information

- Nous avons vu comment catégoriser des documents
- Nous avons vu comment les étiqueter automatiquement
- Une fois un document trouvé, comment y extraire l'information désirée automatiquement?
- Exemple: extraire l'information d'une annonce de séminaire
 - ◆ le nom du présentateur
 - ◆ la date de la présentation

« *There will be a seminar by Dr. Andrew McCallum on Friday* »



Présentateur: *Dr. Andrew McCallum*
Date: *Friday (vendredi)*

Extraction d'information

- On peut aussi **formuler comme un problème d'étiquetage de mots!**

- ◆ il y a **4 étiquettes**

- » **PRE** : préambule de l'information cherchée
- » **TARGET** : l'information à extraire
- » **POST** : fin de l'information
- » **-** : autres mots

Text:	There	will	be	a	seminar	by	Dr.	Andrew	McCallum	on	Friday
Speaker:	-	-	-	-	PRE	PRE	TARGET	TARGET	TARGET	POST	-
Date:	-	-	-	-	-	-	-	-	-	PRE	TARGET

- On entraînerait un HMM par information recherchée
 - ◆ HMM « présentateur »
 - ◆ HMM « date »

Extraction d'information

- On peut aussi **formuler comme un problème d'étiquetage de mots!**

- ◆ il y a **4 étiquettes**

- » **PRE** : préambule de l'information cherchée
- » **TARGET** : l'information à extraire
- » **POST** : fin de l'information
- » **-** : autres mots

Text:	There	will	be	a	seminar	by	Dr.	Andrew	McCallum	on	Friday
Speaker:	-	-	-	-	PRE	PRE	TARGET	TARGET	TARGET	POST	-
Date:	-	-	-	-	-	-	-	-	-	PRE	TARGET

- Pour le HMM « présentateur », le corpus d'entraînement contiendrait

[(« There », -), (« will », -), (« be », -), (« a », -), (« seminar », PRE), (« by », PRE),
(« Dr. », TARGET), (« Andrew », TARGET), (« McCallum », TARGET), (« on », POST), (« Friday », -)]

Extraction d'information

- On peut aussi **formuler comme un problème d'étiquetage de mots!**

- ◆ il y a **4 étiquettes**

- » **PRE** : préambule de l'information cherchée
- » **TARGET** : l'information à extraire
- » **POST** : fin de l'information
- » **-** : autres mots

Text:	There	will	be	a	seminar	by	Dr.	Andrew	McCallum	on	Friday
Speaker:	-	-	-	-	PRE	PRE	TARGET	TARGET	TARGET	POST	-
Date:	-	-	-	-	-	-	-	-	-	PRE	TARGET

- Pour le HMM « présentateur », le corpus d'entraînement contiendrait

[(« There », -), (« will », -), (« be », -), (« a », -), (« seminar », -), (« by », -),
(« Dr. », -), (« Andrew », -), (« McCallum », -), (« on », PRE), (« Friday », TARGET)]

Extraction d'information

- On peut aussi **formuler comme un problème d'étiquetage de mots!**

- ◆ il y a **4 étiquettes**

- » **PRE** : préambule de l'information cherchée
- » **TARGET** : l'information à extraire
- » **POST** : fin de l'information
- » **-** : autres mots

Text:	There	will	be	a	seminar	by	Dr.	Andrew	McCallum	on	Friday
Speaker:	-	-	-	-	PRE	PRE	TARGET	TARGET	TARGET	POST	-
Date:	-	-	-	-	-	-	-	-	-	PRE	TARGET

- Étant donnée une nouvelle phrase
 - ◆ l'explication la plus plausible calculée à partir du HMM « présentateur » permettrait d'isoler l'information sur le présentateur
 - ◆ l'explication la plus plausible calculée à partir du HMM « date » permettrait d'isoler l'information sur la date de présentation

Conclusion

- Le traitement automatique de la langue est un des domaines piliers en IA
- Les approches probabilistes et d'apprentissage automatique sont actuellement les outils les plus souvent employés
 - ◆ ex.: on aurait aussi pu utiliser les algorithmes de classification vus dans le cours d'apprentissage automatique
- L'état de l'art est très bon pour modéliser les relations syntaxiques entre les mots (particulièrement en anglais)
- Modéliser les relations sémantiques entre les mots reste un défi...
- Voir aussi **IFT 501 – Recherche d'information et forage de données**
 - ◆ PageRank: l'algorithme au coeur de la première version de l'engin de recherche de Google

Vous devriez être capable de...

- Classification de documents
 - ◆ simuler la classification à l'aide du modèle bayésien naïf multinomial
 - ◆ comprendre les hypothèses faites par ce modèle
 - ◆ comprendre l'impact du prétraitement des données
- Modèle de langage
 - ◆ savoir ce qu'est un modèle de langage
 - ◆ savoir ce qu'est un modèle n -gramme
 - ◆ connaître les techniques de lissage et à quoi elles servent
 - ◆ savoir à quoi peut servir un modèle de langage
- Étiquetage syntaxique et extraction d'information
 - ◆ pouvoir décrire les étapes pour résoudre ces tâches