IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

UVF3B403 MS IABDA
Fouille de texte

Yannis Haralambous (IMT Atlantique)

27 janvier 2017

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

La fouille de texte

Yannis Haralambous
(IMT Atlantique)

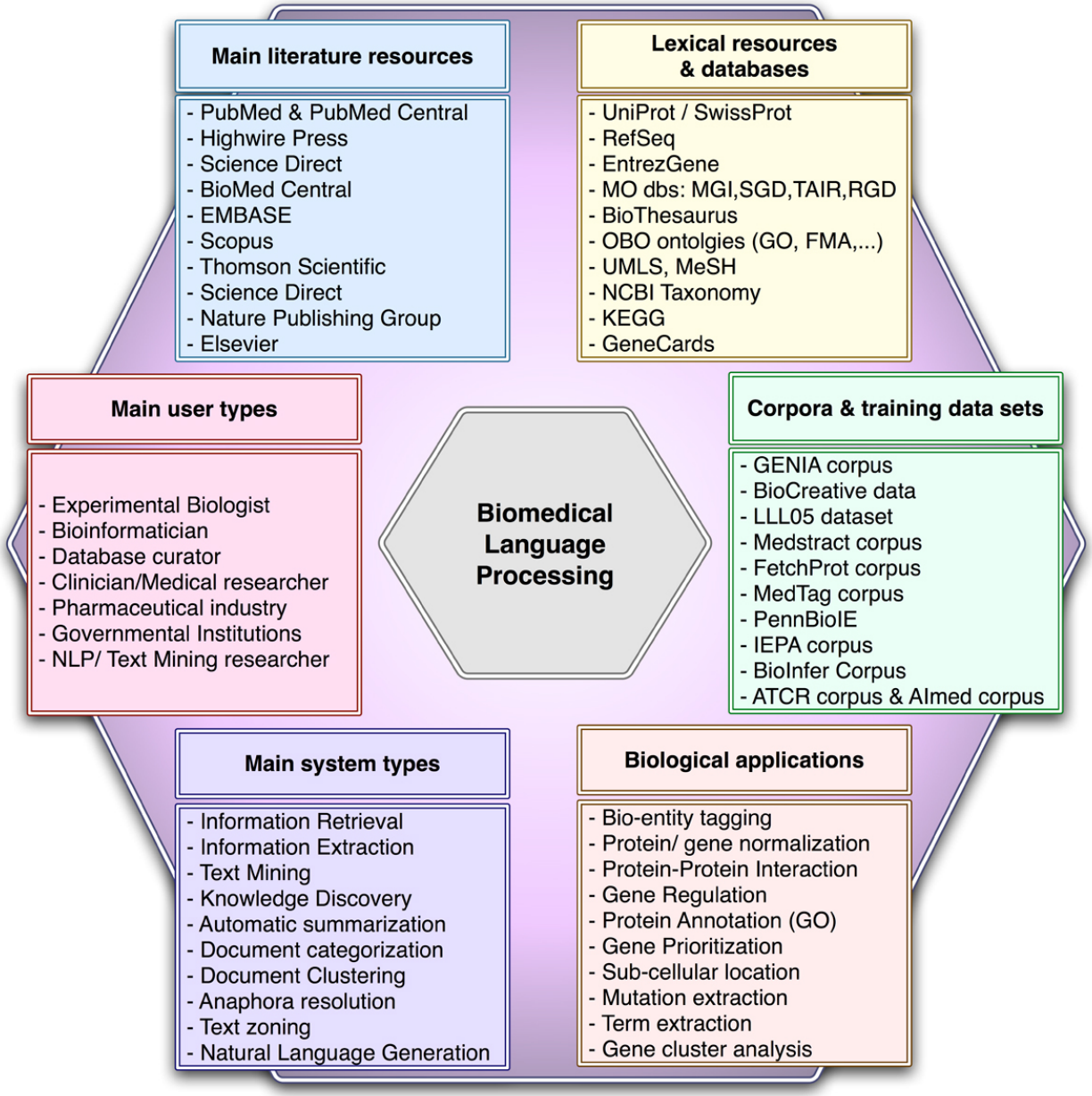
- La *fouille de texte* (en anglais *text mining*) est la découverte de connaissances nouvelles dans des données textuelles (potentiellement de volume important).
- On y combine des techniques de fouille de données, d'apprentissage artificiel, de statistique, de traitement automatique de langue.
- Avec la montée en puissance des machines toute méthode de traitement de langue peut potentiellement être utilisée pour la fouille (exemple : la traduction automatique → alignement de textes → fouille multilingue, désambiguïsation, etc.).

Applications

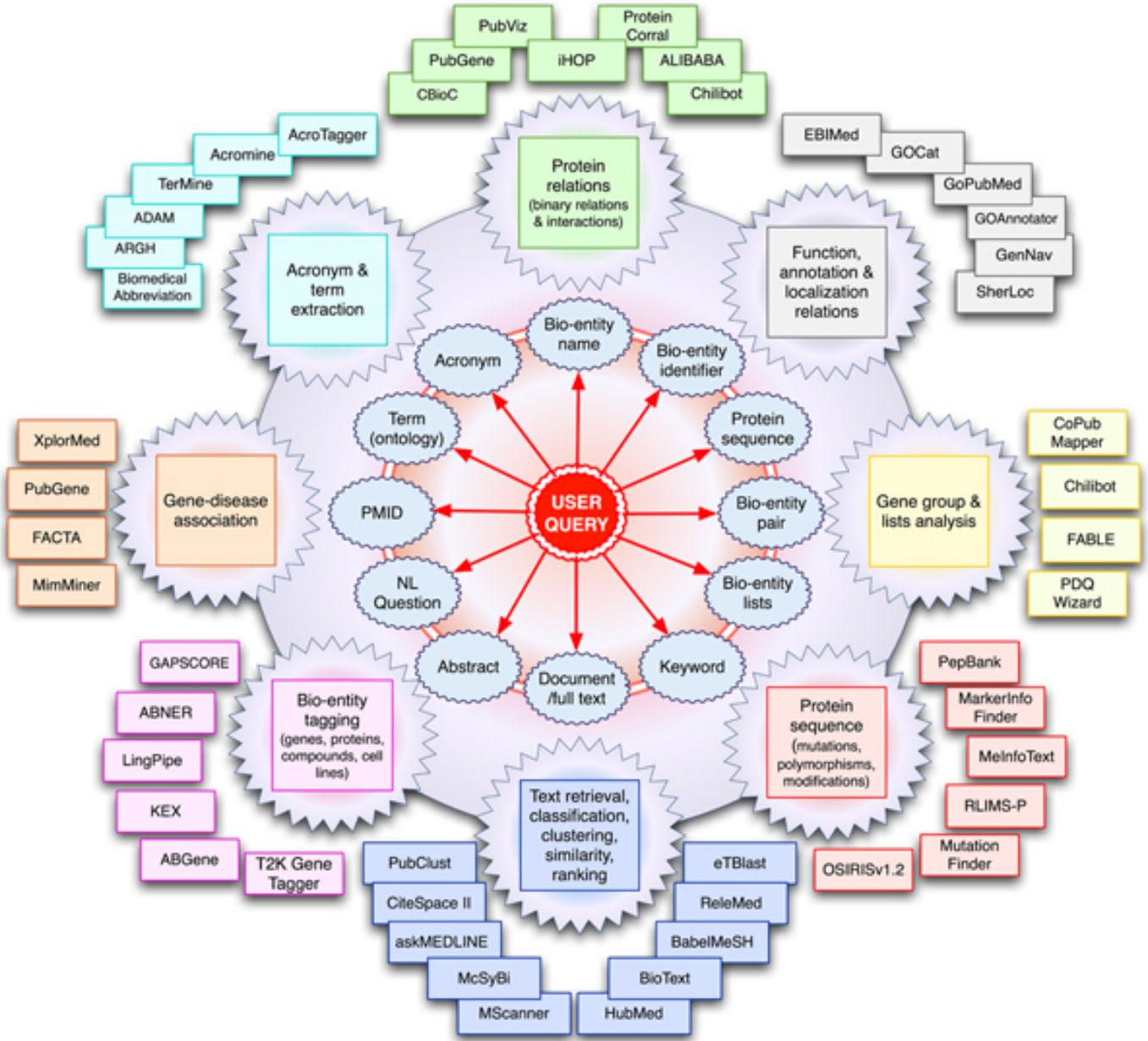
Nombreuses applications :

- sécurité : trouver des terroristes par leur twits... ;
- veille concurrentielle : analyse des articles de presse, brèves, blogs, forums, Twitter, etc. (opinion) ;
- veille scientifique : analyse et résumé automatique des publications, des brevets, des pages Web, des blogs, etc. (découvertes) ;
- économie / marketing : gestion de relation clients, analyse de messages ;
- systèmes question-réponse en langue naturelle (achats en ligne, demandes de renseignements) ;
- extraction d'information ($\longrightarrow$ Web mining, etc.) ;
- classification de textes (par exemple : SPAM) ;
- découverte scientifique, en particulier dans le secteur biomédical (40 kpublications par mois) ;
- ... tout ce qui relève de l'utilisation de la langue sur le Web, et ailleurs.

La fouille de texte - le biomédical



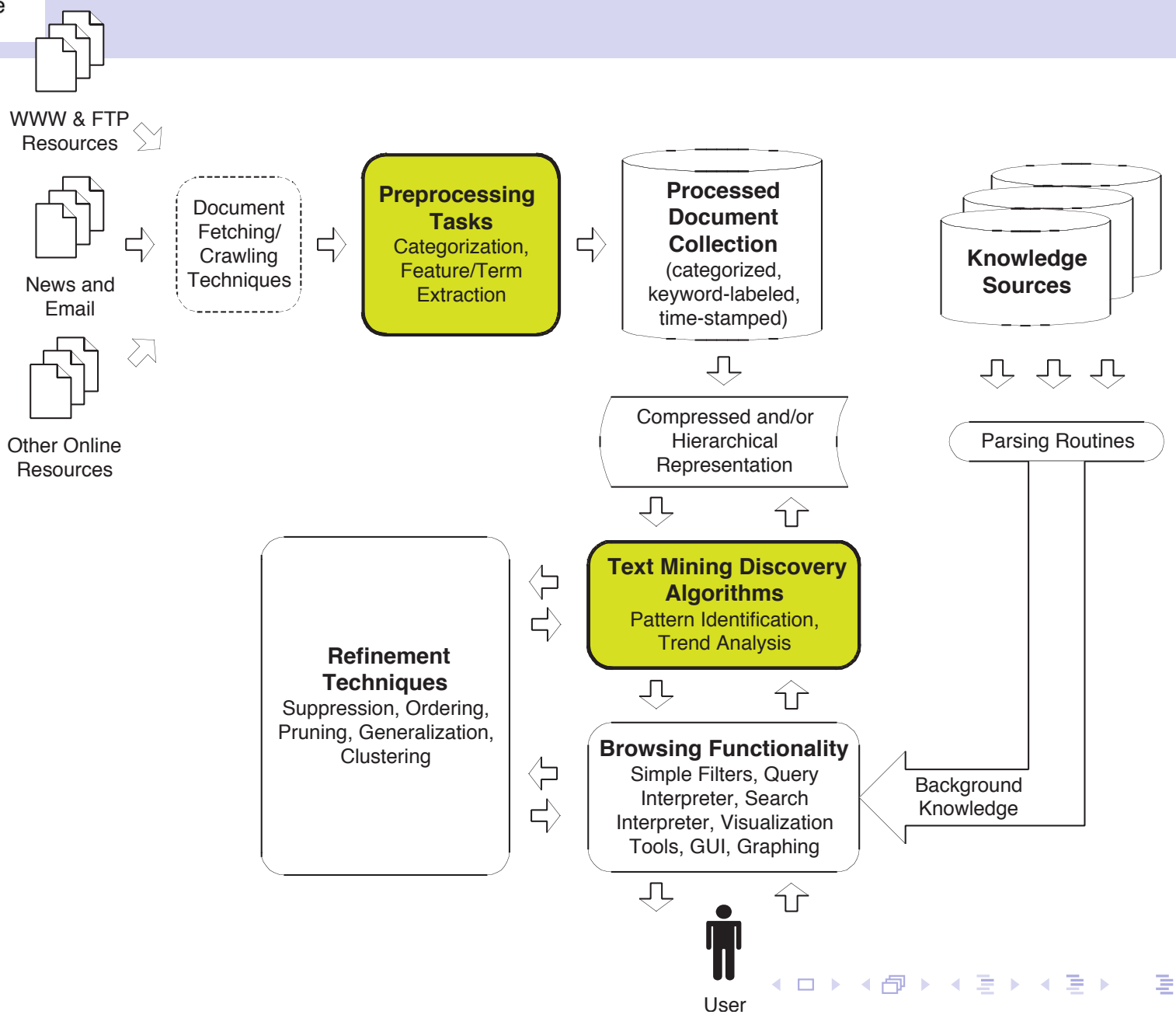
La fouille de texte - le biomédical



La fouille de texte par rapport à la fouille de données

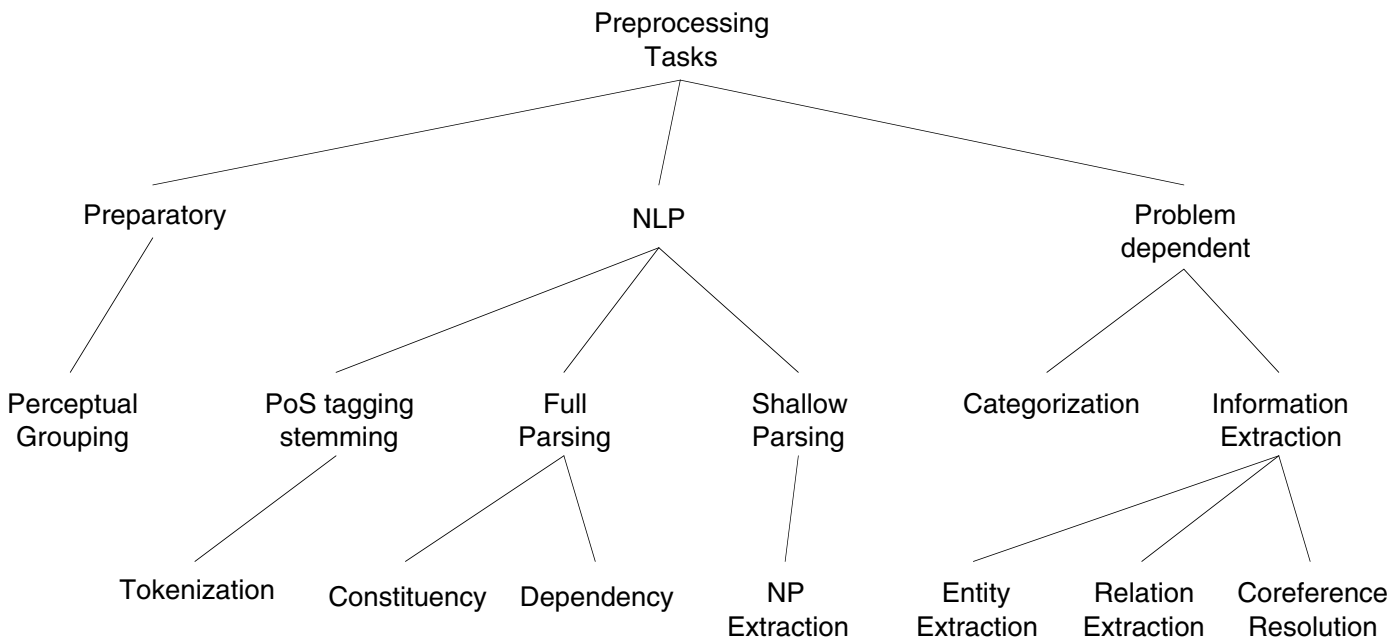
- Pour les datamineurs, la fouille de texte se caractérise par les propriétés suivantes :
  1. les données (= textes) ont une structure implicite (= la structure linguistique) ;
  2. cette structure est flexible car individuelle pour chaque locuteur, mais aussi partagée par tous les locuteurs de la langue ;
  3. elle est complexe et multicouches : la linguistique définit au moins cinq couches d'étude de la langue :
     1. phonologie / graphématique,
     2. morphologie,
     3. syntaxe,
     4. sémantique,
     5. pragmatique ;
- à l'issue des traitements linguistiques on obtient des données, que l'on peut ensuite fouiller comme d'ordinaire ;
- c'est donc la phase de « prétraitement » qui est importante.

Architecture d'un système de fouille de texte



Le prétraitement

- Le *prétraitement* consiste à traiter le corpus pour obtenir des *représentations de documents* ;
- pour chaque document, une telle représentation peut être simpliste (quelques termes ou concepts dans d'une base de connaissances) ou complexe (le résultat d'une analyse linguistique plus ou moins poussée du texte)



La tokénisation

- La *tokénisation* consiste à couper le texte en *tokens* (que l'on a va éventuellement classer en tokens de mot, de blanc, de ponctuation, etc.) ;
- un *token* peut être, selon le besoin, un caractère, une syllabe, un mot, une phrase, un paragraphe, une section, etc.
- pour les caractères : Unicode Standard Annex #29 ;
- pb avec écritures qui ne séparent pas les mots (chinois, thaï, khmer, mais aussi partiellement arabe, etc.) ;
- TRÈS IMPORTANT : MÉTHODE GÉNÉRALE quand on bloque sur une couche on se sert des couches supérieures ;
- on est à la base de l'édifice, si on se trompe maintenant toutes les analyses ultérieures vont se planter !

Première couche linguistique : phonologie

- La *phonétique* décrit tous les sons que la bouche de l'humain peut produire ;
- la *phonologie* définit des classes d'équivalence de sons pour une langue donnée, on les appelle *phonèmes* ;
- exemple : en japonais Larry = Rally = ラリ, il s'agit d'un seul phonème la/ra/ラ et d'un seul phonème li/ri/リ ;
- les phonèmes sont validés par la *méthode des paires distinctives* : batte/patte, pif/pouf, pif/pic ;
- la *phonologie dérivationnelle* considère qu'il existe des formes abstraites des mots qui deviennent concrètes à travers des dérivations (→ Chomsky) : in + mature → immature ;
- la *graphématique* étudie les unités élémentaires de l'écriture et leurs relations avec les phonèmes.

Deuxième couche linguistique : la morphologie

- La morphologie étudie les *morphèmes* qui sont les briques dont on se sert pour faire des phrases. Leur régularité et leurs oppositions forment une architecture mentale qui nous permet d'exprimer la réalité.
- Un morphème est la plus petite unité porteuse de sens qu'il soit possible d'isoler dans un énoncé.
- Les *morphèmes libres* (ou bases lexicales) peuvent être trouvés seuls et on peut en ajouter des nouveaux à une langue : table, marche, souvent, ...
- Les *morphèmes liés* (ou affixes) ne peuvent être seuls et leur liste est (en principe) fermée : table**s**, marche**ront**, etc.
- en japonais, les caractères chinois sont des morphèmes libres et les particules (écrits en hiragana) sont des morphèmes liés.

Deuxième couche linguistique : la morphologie

- La *morphologie flexionnelle* étudie les transformations des mots qui ne produisent pas de mot nouveau (marche → marchera, marchons, marchais, etc., table → tables).
- Principales catégories de la morphologie flexionnelle : pour les noms : le nombre (singulier / pluriel), le genre (masculin / féminin / neutre), le cas (nominatif / génitif / accusatif, etc.). Pour les verbes : la personne (1re, 2e, 3e), le mode (impératif / subjonctif / indicatif), la voix (active / passive), le temps (présent, futur, etc.). Chaque langue a sa propre configuration :
  - le français a 2 nombres, 2 genres, aucun cas, le verbe s'accorde avec le nombre du nom ;
  - l'allemand a 2 nombres, 3 genres, 4 cas (n,g,d,a) ;
  - le grec a 3 nombres, 3 genres, 5 cas (n,g,d,a,v) ;
  - le russe a 3 nombres, 3 genres, 6 cas (n,g,d,a,l,i) ;
  - l'arabe a 3 nombres, 2 genres, 3 cas (n,g,a), le verbe dépend aussi du genre du nom ;
  - le japonais n'a ni personne, ni nombre, ni genre, ni cas, tout est indiqué à l'aide de particules ;
  - etc.

Deuxième couche linguistique : la morphologie

- La *morphologie dérivationnelle* produit des nouveaux mots : table → tablette, marche → démarche, mange → mangeable, etc.
- Mécanismes dérivationnels : suffixation (manger → mangeable), préfixation (tirer → retirer), composition (canapé-lit, casse-noisettes), siglaison (CNRS, ovni), troncation (sympa, métro, psy), acronymie (franglais, alcootest), reduplication (fefille, pépère).
- En arabe tout mot peut être obtenu à partir d'une *racine sémitique*, le plus souvent trilittère. Par exemple : $\sqrt{\text{ك ت ب}}$ donne كتاب (livre) كتب (livres) كاتب (écrivain) مكتبة (bureau) يكتب (il écrit), etc.

Deuxième couche linguistique : la morphologie

- Dans un texte on trouve des *occurences* d'un mot. Quand ces occurences sont des formes du même mot, on dira qu'il s'agit du même *lexème* (ou unité lexicale). Les unités lexicales utilisées par une personne forment son *vocabulaire*, celles d'une communauté linguistique forment le *lexique* d'une langue.
- On étudie les unités lexicales dans des collections de documents qui se ressemblent par leur origine, que l'on appelle des *corpus*.

POS tagging + *stemming* : les tags

- La *désuffixation* (*stemming*) efface les terminaisons d'un mot (par exemple : le radical de « porteront » est `port`). Cela n'a rien de linguistique mais peut être pratique.
- La *lemmatisation* consiste à trouver la forme canonique d'un mot (par exemple : celle de « porteront » est `porter` : l'infinitif).
- L'*étiquetage grammatical* (*POS tagging*) est le processus qui consiste à associer aux mots d'un texte leur fonction grammaticale (sous forme de *tags*).
- Voici la liste de tags du logiciel libre *TreeTagger* `http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/`

ABR	abréviation (*CNRS, ENST, ...*)
ADJ	adjectif (*long, court, gros, ...*)
ADV	adverbe (*heureusement, dûment,...*)
DET:ART	article (*le, la, les*)
DET:POS	pronom possessif (*ma, ta, ...*)
INT	interjection (*Ah! Holà! Boum! Atchoum! Berk! ...*)
KON	conjonction (*Mais où e(s)t donc Ornicar? quand*)

Étiquetage : les tags

NAM	nom propre (*Noé*, *Meyer*, ...)
NOM	nom (*cheval*, *gâteau*, *espoir*, ...)
NUM	numéral (*15*, *trois*, *XXIX*, ...)
PRO	pronom
PRO:DEM	pronom démonstratif (*celui*, *celui-ci*, ...)
PRO:IND	pronom indéfini (*personne*, *plusieurs*, *chacun*, ...)
PRO:PER	pronom personnel (*je*, *me*, *moi*, *tu*, *te*, *toi*, ...)
PRO:POS	pronom possessif (*mien*, *tien*, ...)
PRO:REL	pronom relatif (*qui*, *lequel*, *quiconque*, ...)
PRP	préposition (*de*, *à*, *pour*, *en*, *dans*, *avec*, *sur*, *par*, ...)
PRP:det	préposition plus article (*au*, *du*, *aux*, *des*)
PUN	ponctuation
PUN:cit	ponctuation citation (guillemets)
SENT	tag de phrase
SYM	symbole

Étiquetage : les tags

VER:cond	verbe au conditionnel (*j'irais*)
VER:futu	verbe au futur (*j'irai*)
VER:impe	verbe à l'imperatif (*va*)
VER:impf	verbe à l'imparfait (*j'allais*)
VER:infi	verbe à l'infinitif (*aller*)
VER:pper	verbe au participe passé (*allé*)
VER:ppre	verbe au participe présent (*allant*)
VER:pres	verbe au présent (*je vais*)
VER:simp	verbe au passé simple (*j'allai*)
VER:subi	verbe à l'imparfait du subjonctif (*j'allasse*)
VER:subp	verbe au présent du subjonctif (*j'aille*)

Autres tags fréquemment utilisés (outre-atlantique) : le *Penn Treebank Tagset* `http://www.mozart-oz.org/mogul/doc/lager/brill-tagger/penn.html`

Étiquetage : exemple

Longtemps, je me suis couché de bonne heure. Parfois, à peine ma bougie éteinte, mes yeux se fermaient si vite que je n'avais pas le temps de me dire : « Je m'endors. »

donne :

Longtemps	ADV	longtemps
,	PUN	,
je	PRO:PER	je
me	PRO:PER	me
suis	VER:pres	suivrelêtre
couché	VER:pper	coucher
de	PRP	de
bonne	ADJ	bon
heure	NOM	heure
.	SENT	.
Parfois	ADV	parfois
,	PUN	,
à	PRP	à
peine	NOM	peine
ma	DET:POS	mon

Étiquetage : exemple

bougie	NOM	bougie
éteinte	VER:pper	éteindre
,	PUN	,
mes	DET:POS	mon
yeux	NOM	œil
se	PRO:PER	se
fermaient	VER:impf	fermer
si	ADV	si
vite	ADV	vite
que	KON	que
je	PRO:PER	je
n’	ADV	ne
avais	VER:impf	avoir

pas	ADV	pas
le	DET:ART	le
temps	NOM	temps
de	PRP	de
me	PRO:PER	me
dire	VER:infi	dire
:	PUN	:
«	PUN:cit	«
Je	PRO:PER	je
m’	PRO:PER	me
endors	VER:pres	endormir
.	SENT	.
»	PUN:cit	»

Comment fonctionnent les tagueurs ?

- À partir d'un corpus tagué, on calcule les fréquences des correspondances mot+tag. Si elles sont trop rares (inf. à 1%) on les ignore.
- (Un *n-gramme* est un n-uplet d'objets consécutifs, les objets pouvant être des caractères, des syllabes, des mots, des termes, des phrases, etc.)
- Une première approche : on utilise des n-grammes de mots (souvent $n = 2$ ou 3) et on émet une hypothèse markovienne : le tag d'un mot ne dépend que de celui du précédent, et ce comportement ne varie pas dans le temps.

PARENTHÈSE : Modèles visibles de Markov

- Soit $X = (X_1, \ldots, X_T)$ une suite de v.a. à valeurs dans un espace fini $S = \{s_1, \ldots, s_N\}$. Les *propriétés de Markov* sont :
  1. l'horizon limité : $P(X_{t+1} = s_k \mid X_1 \ldots X_t) = P(X_{t+1} = s_k \mid X_t)$,
  2. la stationnarité : $P(X_{t+1} = s_k \mid X_1 \ldots X_t) = P(X_2 = s_k \mid X_1)$;
- X est alors une *chaîne de Markov* et on peut la considérer comme un automate dont S sont les états, et dont les transitions de s_i à s_j ont des probabilités $a_{ij} = P(X_{t+1} = s_j \mid X_t = s_i)$ ($a_{*,*}$ est appelée *matrice stochastique*) ;
- les probabilités des états initiaux s_i sont $\pi_i = P(X_1 = s_i)$ (avec $\sum \pi_i = 1$) ;
- la probabilité d'avoir une suite avérée d'états X est

$$P(X_1, \ldots, X_T) = \pi_{X_1} \prod_{t=1}^{T-1} a_{X_t X_{t+1}}$$

- on appelle cela un *modèle de Markov visible*.

PARENTHÈSE : Modèles cachés de Markov

- Quand on ne connaît pas la suite d'états mais uniquement une fonction aléatoire de ceux-ci, on parle de *modèle de Markov caché*.
- On procède par apprentissage.
- Soient w_i les mots (resp. t_i les tags) de la phrase à taguer, w^ℓ les mots du lexique (resp. t^j les tags disponibles), $C(w^\ell)$ (resp. $C(t^j)$) le nb. d'occ. de w^ℓ (resp. t^j) dans le corpus d'entraînement, $C(t^j, t^k)$ (resp. $C(w^\ell : t^j)$) le nb. d'occ. de t^j suivi de t^k (resp. du mot w^ℓ avec le tag t^j),
- alors $\hat{P}(t^k \mid t^j) = \frac{C(t^j,t^k)}{C(t^j)}$ est le MV (maximum de vraisemblance) que le tag t^k suive le tag t^j,
- $\hat{P}(w^\ell \mid t^j) = \frac{C(w^\ell:t^k)}{C(t^j)}$ est le MV que le mot w^ℓ survienne dans le corpus avec le tag t^j,
- alors pour obtenir la suite optimale de tags :

$$\hat{t}_{1,n} = \arg\max_{t_{1,n}} \prod_{i=1}^{n} \hat{P}(w_i \mid t_i)\hat{P}(t_i \mid t_{i-1}).$$

Étiquetage : algorithmes

- Les modèles de Markov cachés sont essentiellement des tagueurs à bigrammes. Dans une phrase du type

  la jeune fille la porte

  impossible de dire à partir de « la » si « porte » est nom ou verbe.
- [Church 1988] décrit un tagueur à trigrammes.
- Dans un tagueur à trigrammes, on peut dire que la probabilité pour une suite de n mot $w_{1,n}$ d'avoir les tags $t_{1,n}$ peut être calculée de manière récursive par

$$p(w_{1,n}, t_{1,n}) := p(t_n|t_{n-2}t_{n-1})p(w_n \mid t_n)p(w_{1,n-1}, t_{1,n-1}).$$

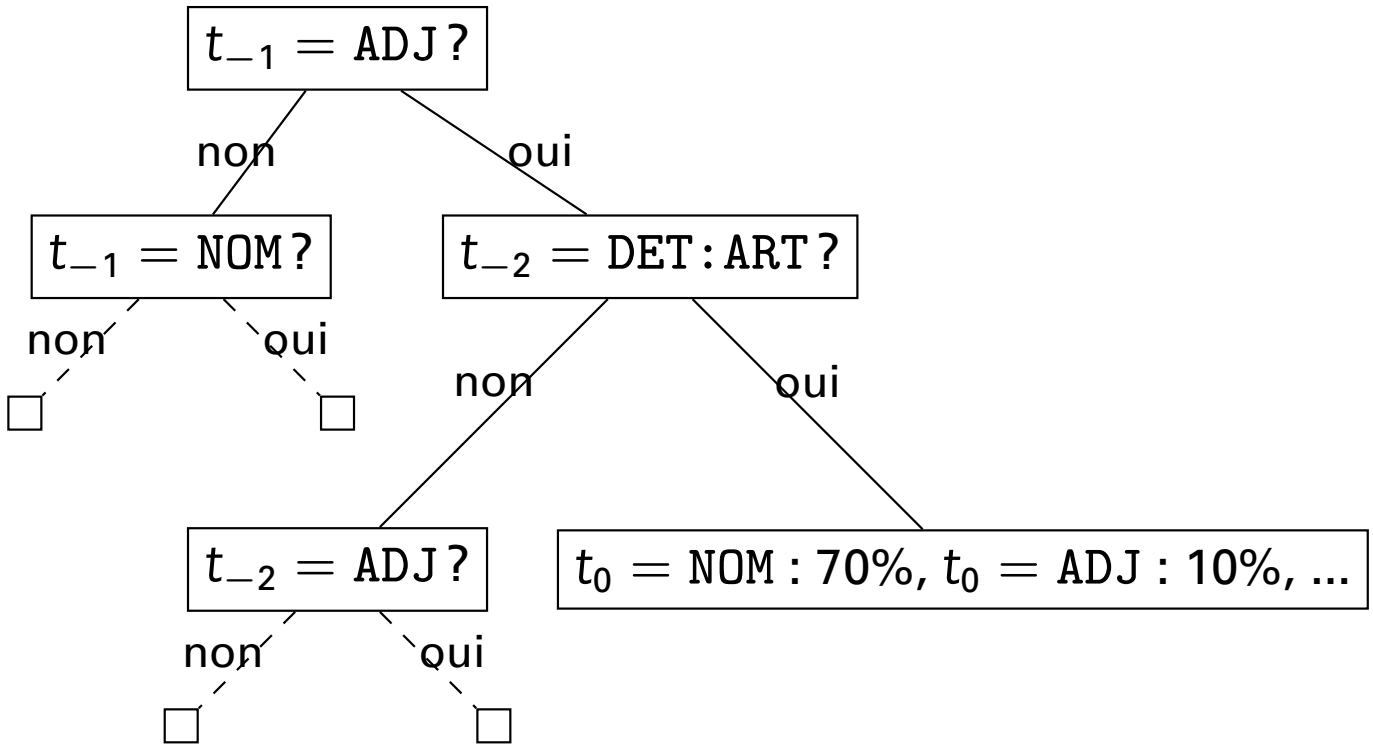
- On peut donc estimer $p(t_n|t_{n-2}t_{n-1})$, en suivant le principe du MV, par

$$\frac{\text{fréquence du trigramme } t_{n-2}t_{n-1}t_n}{\text{fréquence du bigramme } t_{n-2}t_{n-1}}.$$

- Problème : ces probas peuvent être très faibles (voire nulles si le trigramme n'apparaît pas dans le corpus d'apprentissage).

Comment fonctionne *TreeTagger* ?

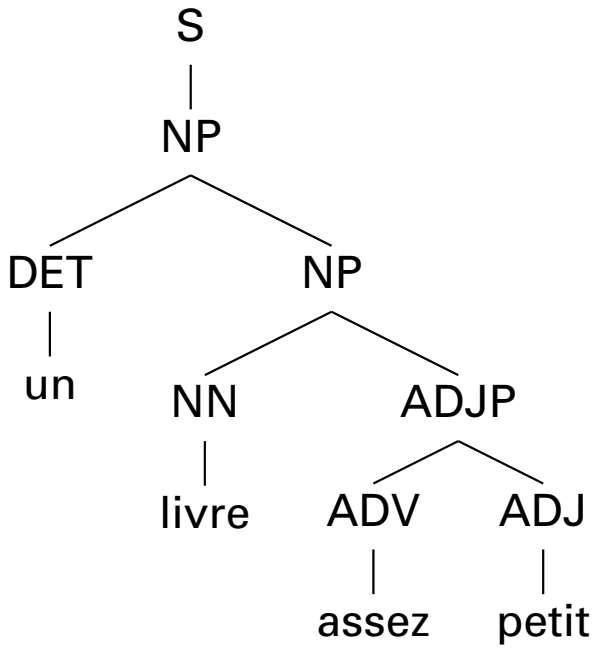
- Solution adoptée par *TreeTagger* : utiliser un arbre binaire de décision pour déterminer $p(t_n|t_{n-2}t_{n-1})$. Exemple :



- L'apprentissage se fait par ID3 sur des corpus tagués.
- *TreeTagger* dispose d'un lexique de mots avec des probabilités de tags pour le $p(w_n \mid t_n)$. Si le mot ne s'y trouve pas, il se sert d'expressions régulières sur les suffixes.

Troisième couche linguistique : la syntaxe

- Une *phrase* est une suite de mots qui obéit, dans le système d'une langue donnée, à un ordre et à des dépendances donnés.
- Une phrase se subdivise en *syntagmes* (ou *groupes*) qui sont formés de mots :



- Chaque groupe a une *tête* qui lui donne son nom. On trouve ainsi des groupes nominal NP (le petit chat noir), verbal VP (pense à mes amis), prépositionnel PP (près de la porte), adjectival ADJP (très fier de son travail), ADVP adverbial (tout aussi simplement), etc.

Troisième couche linguistique : la syntaxe

- De manière similaire aux langages formels, l'arbre syntaxique décrit la réalisation d'un mot du langage à partir d'un certain nombre de productions.
- On peut donc décrire la syntaxe d'un langage en décrivant les règles de production :

$$\begin{aligned} \mathrm{S} &\rightarrow \mathrm{NP} \\ \mathrm{NP} &\rightarrow \mathrm{DET\ NP} \\ \mathrm{NP} &\rightarrow \mathrm{NN\ ADJP} \\ \mathrm{ADJP} &\rightarrow \mathrm{ADV\ ADJ} \end{aligned}$$

  etc.
- Les feuilles de cet arbre sont les mots. Les parents des feuilles sont les tags de partie du discours (en informatique : les tags POS).

Troisième couche linguistique : la syntaxe

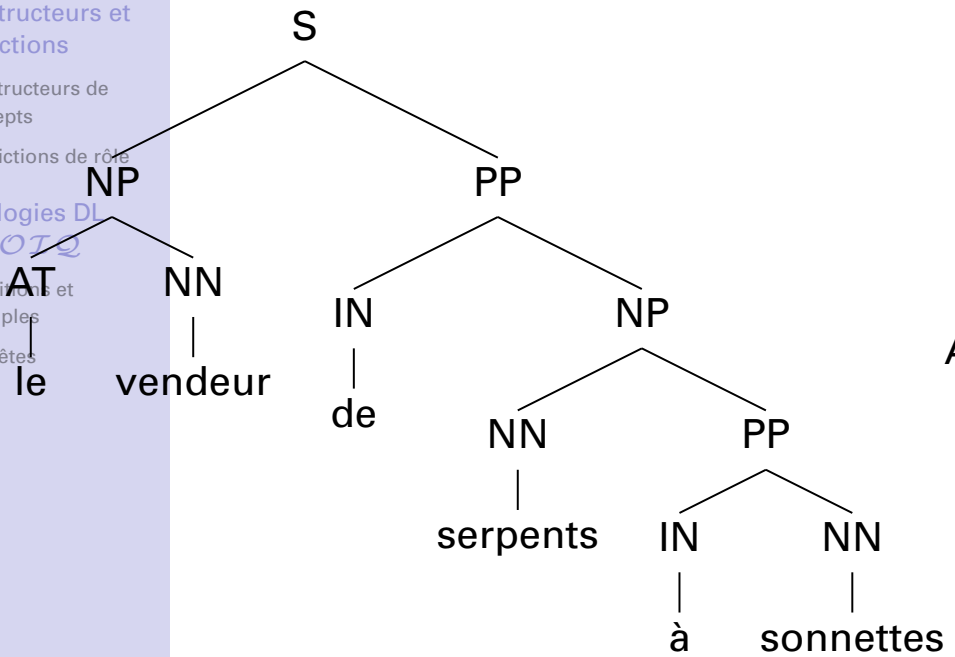
- La syntaxe agit sur la morphologie. Ainsi, en français, appliquer la règle NP $\longrightarrow$ DET ADJ NN ADJ implique que les adjectifs s'accordent en genre (et en nombre) avec le nom: « des petits livres rouges ».
- Certains traits des verbes interdisent certaines productions, par exemple lorsqu'un verbe est intransitif la production VP $\longrightarrow$ VBZ NP (la fille dort la pomme) est impossible (sauf en poésie !).
- Autre exemple : « prendre » est toujours suivi d'un COD, sauf dans l'expression « la mayonnaise prend »...
- Les phrases du monde réel sont souvent plus complexes que ces quelques exemples (cf. transparent suivant). Les phrases peuvent être *enchâssées* (Elle sait que Pierre est parti), *coordonnées* (j'ai cherché les clés mais je ne les ai pas trouvées), *transformées* (transformations négative, impérative, interrogative, passive, et leurs combinaisons), etc.

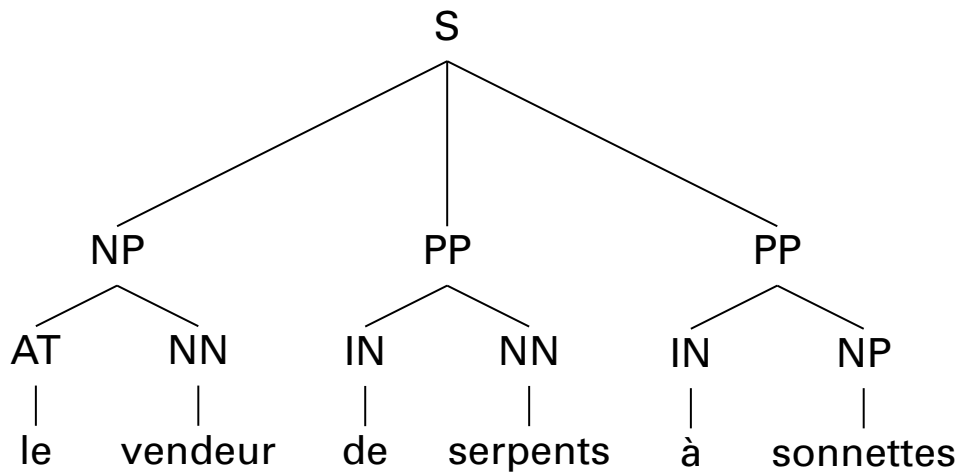
La phrase la plus longue de Proust (243 mots)

Mais au lieu de la simplicité, c'est le faste que je mettais au plus haut rang, si, après que j'avais forcé Françoise, qui n'en pouvait plus et disait que les jambes « lui rentraient », à faire les cent pas pendant une heure, je voyais enfin, débouchant de l'allée qui vient de la Porte Dauphine — image pour moi d'un prestige royal, d'une arrivée souveraine telle qu'aucune reine véritable n'a pu m'en donner l'impression dans la suite, parce que j'avais de leur pouvoir une notion moins vague et plus expérimentale, — emportée par le vol de deux chevaux ardents, minces et contournés comme on en voit dans les dessins de Constantin Guys, portant établi sur son siège un énorme cocher fourré comme un cosaque, à côté d'un petit groom rappelant le « tigre » de « feu Baudenord », je voyais — ou plutôt je sentais imprimer sa forme dans mon cœur par une nette et épuisante blessure — une incomparable victoria, à dessein un peu haute et laissant passer à travers son luxe « dernier cri » des allusions aux formes anciennes, au fond de laquelle reposait avec abandon M^{me} Swann, ses cheveux maintenant blonds avec une seule mèche grise ceints d'un mince bandeau de fleurs, le plus souvent des violettes, d'où descendaient de longs voiles, à la main une ombrelle mauve, aux lèvres un sourire ambigu où je ne voyais que la bienveillance d'une Majesté et où il y avait surtout la provocation de la cocotte, et qu'elle inclinait avec douceur sur les personnes qui la saluaient.

Arbres syntaxiques

- Des logiciels appelés *analyseurs syntaxiques* peuvent calculer le (ou les) arbre syntaxique(s) d'une phrase. Parfois il y a plus d'un arbre, correspondant à des sémantiques différentes. Pour désambiguïser on passe aux niveaux supérieurs (sémantique, pragmatique) :





- Autre exemple illustre : la jeune porte le voile.

Les SCFG

- Pour décrire ces arbres on peut utiliser des *grammaires formelles hors-contexte stochastiques* (SCFG). Ce sont des grammaires formelles hors-contexte (type 2 dans la hiérarchie de Chomsky) avec des probabilités sur les règles de production $P(N^i \longrightarrow \zeta^j \mid N^i)$ où N^i est un non-terminal et ζ^j une suite de terminaux et de non-terminaux (avec $\sum_j P(N^i \longrightarrow \zeta^j \mid N^i) = 1$ pour tout i).
- La probabilité d'une phrase par rapport à la grammaire G est $P(w_{1m}) = \sum_t P(w_{1m}, t)$ pour tout arbre t dont les feuilles sont w_{1m}.
- On note N^j_{kl} la branche sous N^j dont les feuilles sont les mots $w_k, \ldots, w_l$.
- On pose trois conditions :
  1. l'invariance spatiale : $P(N^j_{k(k+c)} \longrightarrow \zeta)$ reste la même qq soit k ;
  2. hors-contexte : $P(N^j_{kl} \longrightarrow \zeta \mid \text{mots en dehors de } w_{kl}) = P(N^j_{kl} \longrightarrow \zeta)$;
  3. hors-ancêtres : $P(N^j_{kl} \longrightarrow \zeta \mid \text{nœuds ancêtres de } N^j_{kl}) = P(N^j_{kl} \longrightarrow \zeta)$.

Les SCFG

Petit exemple d'application des conditions :

$$P\left(\begin{array}{ccc} & \text{S} & \\ \text{NP} & & \text{VP} \\ \text{l'} \quad \text{homme} & & \text{ronfle} \end{array}\right) =$$

$$\begin{aligned} &= P(\text{S} \to \text{NP VP}, \text{NP} \to \text{l'homme}, \text{VP} \to \text{ronfle}) \\ &= P(\text{S} \to \text{NP VP}) \cdot P(\text{NP} \to \text{l'homme} \mid \text{S} \to \text{NP VP}) \\ &\quad \cdot P(\text{VP} \to \text{ronfle} \mid \text{S} \to \text{NP VP}, \text{NP} \to \text{l'homme}) \\ &\overset{\text{cond. 2}}{=} P(\text{S} \to \text{NP VP}) \cdot P(\text{NP} \to \text{l'homme}) \cdot P(\text{VP} \to \text{ronfle} \end{aligned}$$

où la cond. 1 nous a permis d'utiliser des expressions générales alors qu'elles sont extraites d'un arbre spécifique.

Les SCFG

- Pour faire des calculs on passe à la *forme normale de Chomsky* : on n'admet que des règles du type
  1. $N^i \longrightarrow N^j N^k$,
  2. $N^i \longrightarrow w^j$.

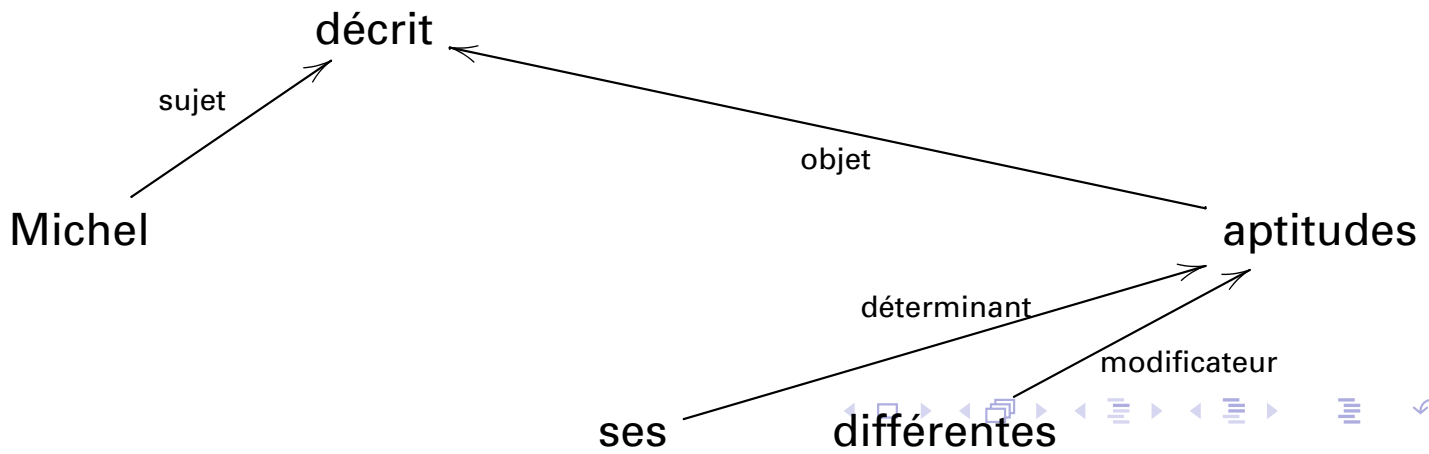
  donc, pour n non-terminaux et V terminaux, on a $n^3 + nV$ règles possibles.
- On calcule la probabilité d'une phrase selon une grammaire G en utilisant des techniques similaires à celles des HMM.
- Idem pour le calcul de l'arbre le plus probable pour une phrase. L'algorithme de Viterbi nous évite de parcourir tous les arbres possibles (complexité exponentielle).
- Les méthodes d'*inférence grammaticale* (un domaine très actif) nous permettent d'identifier les règles qui produisent un ensemble de phrases prétaguées, et leurs probabilités de manière à coller au plus au corpus d'apprentissage.

Problèmes avec les SCFG

- Les SCFG ne tiennent pas compte des mots et de leurs propriétés. Une phrase du type « L'homme dort la lune » sera considérée comme correcte puisque la SCFG ignore l'intransitivité du verbe dormir.
- (Parenthèse : Ainsi, le Lefff [Sagot 2010] nous donne les infos suivantes sur la forme « dort » :
`dort [pred='dormir_____1<Suj:cln|sn>',@pers,cat=v,@P3s]`
alors que pour le verbe « prendre » il nous aurait dit :
`prend [pred='prendre_____2<Suj:cln|scompl|sinf|sn,Obj:(cla|sn),Objà:(cld|à-sn)>',@pers,cat=v,@P3s])`
- D'autre part, il paraît que l'hypothèse d'invariance spatiale est fausse dans la réalité : par exemple, les noms propres apparaissent bcp plus souvent en tant que sujets qu'en tant qu'objets.

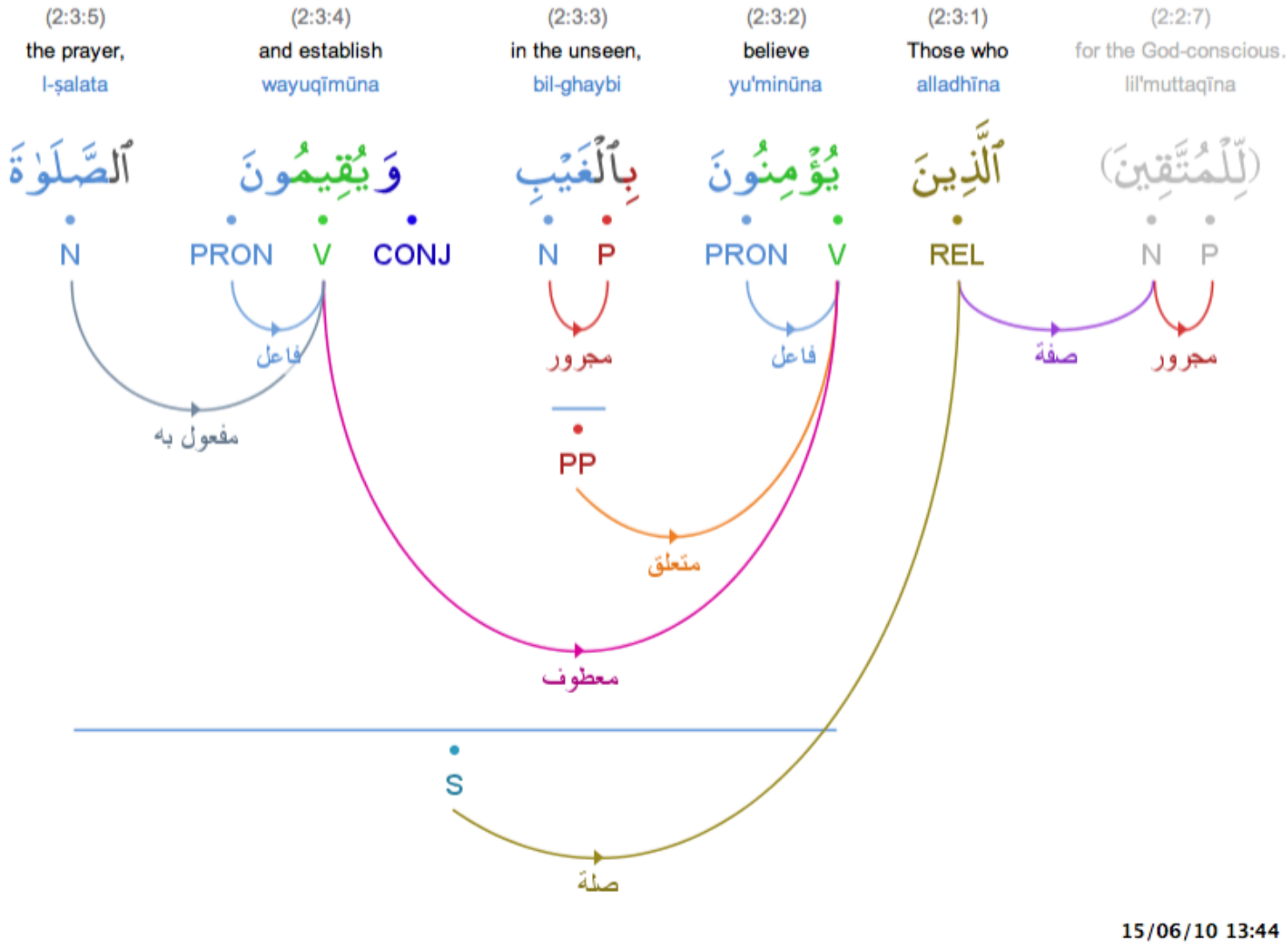
Les grammaires de dépendances de Tesnière

- La syntaxe que nous avons apprise à l'école se base sur une grammaire de constituants : le sujet (ou le GN), le verbe (ou le GV), l'objet, etc.
- Il existe une autre manière de décrire la structure d'une phrase : la *grammaire des dépendances* de Tesnière.
- On considère une phrase comme un mot (la *tête* : le plus souvent, le verbe) auquel sont attachés des *modificateurs*, qui peuvent avoir des modificateurs à leur tour. Le concept fondamental est celui de *relation* entre les mots, relation fléchée entre *dépendant* et *gouverneur*.
- Exemple d'arbre de dépendances :



Autre exemple d'arbre de dépendances

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

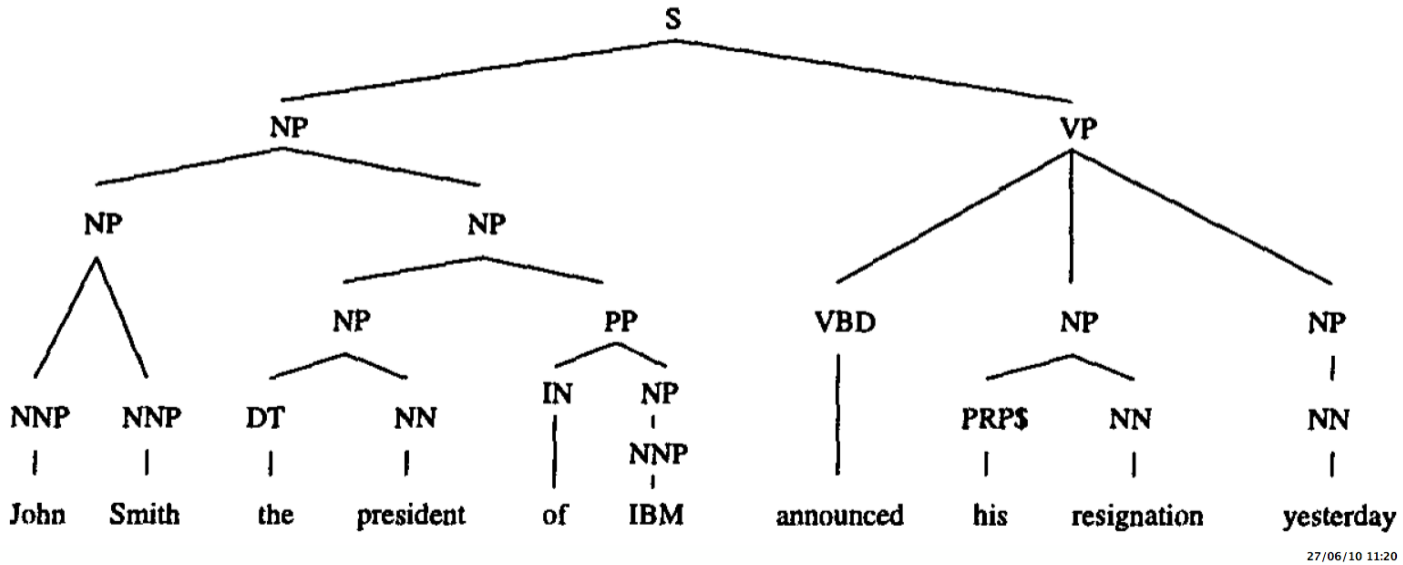


Source :

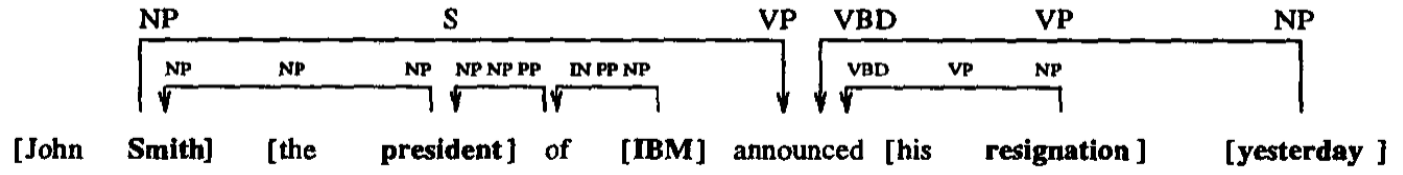
http://corpus.quran.com/treebank.jsp?chapter=2&verse=3&token=1

Autres approches : les dépendances

- [Collins 1996, 1997] présente un modèle statistique et un parseur lexicalisé, basés sur la grammaire des dépendances.
- Le modèle statistique calcule, pour tout arbre T de la phrase S, la probabilité $P(T \mid S)$. Pour cela, S est réduit en $\bar{S} = (B, D)$ (bases et dépendances).
- Exemple :



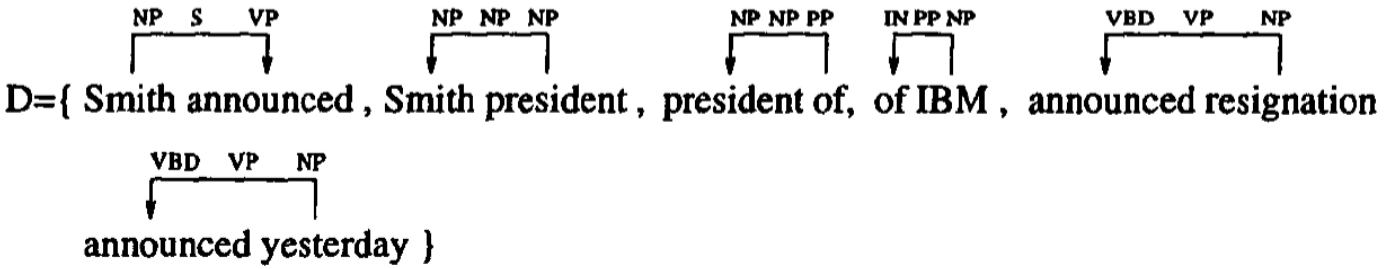
- Voici les dépendances qui en découlent :



Autres approches : les dépendances

- et voici les ensembles B et D :

B={ [John Smith] , [the president], [IBM], [his resignation], [yesterday] }



- Pour réduire l'arbre, il va :
  1. pour chaque nœud intermédiaire trouver le *head-child* par un arbre de décisions [Jelinek *et al.* 1994] ;
  2. faire remonter les *head-child* aussi haut que possible ;
  3. extraire les dépendances.
- Chaque constituent à n enfants produit $n - 1$ dépendances.
- Ayant mis l'arbre à plat, le parseur fait de l'apprentissage sur des triplets des dépendances.

Le *shallow parsing / chunking*

- Le *shallow parsing* (ou *chunking*) est la version allégée de l'analyse syntaxique : au lieu de trouver un arbre syntaxique complet, on se contente d'une structure superficielle. Il convient parfaitement à l'extraction d'information ou à la classification de documents.
- [Abney 1994] définit la notion de *chunk* et la justifie en se basant sur des études psychologiques.
- Pour lui, un chunk est tout mot non-grammatical w accompagné de mots grammaticaux qui l'entourent, *sauf* quand w se trouve entre un autre mot w' et un mot grammatical dépendant de w'. Ainsi, alors que « big », « proud » et « man » sont des noms, « a big proud man » est un seul chunk, puisque « a » dépend de « man ».

Le *shallow parsing / chunking*

- [Sha & Pereira 2003] définissent les chunks comme étant les noyaux non-récursifs de différents groupes de la phrase.
- Les chunks étant non-récursifs, le découpage de la phrase est toujours plat. Exemple :

  $[\text{Indexing}]_N$ $[\text{for the most part}]_N$ $[\text{has involved simply buying}]_V$ $[\text{and then holding}]_V$ $[\text{stocks}]_N$ $[\text{in the correct mix}]_N$ $[\text{to mirror}]_V$ $[\text{a stock market barometer}]_N$.
- L'*Illinois Chunker*
  `http://cogcomp.cs.illinois.edu/page/software_view/13`
  trouve des chunks de 10 types (ADJP, ADVP, CONJP, INTJ, LST, NP, PP, PRT, SBAR, VP):

  [NP Jack and Jill] [VP went] [ADVP up] [NP the hill] [VP to fetch] [NP a pail] [PP of] [NP water]

Différentes approches

- Le problème de base (on se restreint aux chunks nominaux) consiste à associer à chaque mot un « label »
  1. `O` s'il est à l'extérieur de tout chunk ;
  2. `B` s'il est le premier mot d'un chunk ;
  3. `C` s'il appartient à un chunk, sans en être le premier mot.
- Pour faire cela, [Ramshaw & Marcus 1995] utilisent des règles de transformations faisant intervenir jusqu'à 3 mots ou tags POS à gauche et à droite d'un mot donné.
- Entre 1992 et 2001, plusieurs techniques ont été utilisées : les HMM, les MEMM, Winnow, AdaBoost, les SVM, le perceptron généralisé.
- [Sha & Pereira 2003] utilisent des CRF pour attaquer ce problème. Pour un mot w_i ils font intervenir $w_{i-2}, \ldots, w_{i+2}$, $t_{i-2}, \ldots, t_{i+2}$ (les tags POS), y_i et y_{i+1} (les labels de chunk).

Quatrième couche linguistique : la sémantique

- La sémantique est l'étude (scientifique) de la signification.
- Dans un processus d'abstraction à partir des choses qui nous entourent on arrive aux *concepts* et aux *relations* entre les concepts.
- Au-delà des considérations philosophique, on peut se demander comment décrire, classer, calculer les concepts ?
- Une méthode classique consiste à énumérer leur *traits* (= catégorie + valeur) :

français	allemand	échange	donner qqch	recevoir qqch	pour argent	permanent
acheter	kaufen	+	-	+	+	+
vendre	verkaufen	+	+	-	+	+
prêter	ausleihen	+	+	-	-	-
emprunter	leihen	+	-	+	-	-
louer	vermieten	+	+	-	+	-
louer	mieten	+	-	+	+	-

Cette méthode a inspiré la discipline FCA (analyse de concepts formels).

Polysémie et *nymies

On a un certain nombre de phénomènes :

- la *polysémie* : le même mot a plusieurs sens, liés entre eux (exemple : la construction du pont, cet édifice est une belle construction) ;
- l'*homonymie* : plusieurs sens sans rapport entre eux (exemple : lire un livre, une livre sterling) ;
- l'*hyponymie* : quand on a une relation d'implication dans un sens (exemple : le concept de pomme est un hyponyme du concept de fruit, puisque toute pomme est fruit, mais tout fruit n'est pas pomme) ;
- l'*hypéronymie* : l'inverse de l'*hyponymie* ;
- la *synonymie* : quand les implications vont dans les deux sens (exemple : soulier et chaussure). Attention : il peut y avoir des différences stylistiques ou historiques, on n'a jamais de synonymie parfaite (on parle aussi de *cohyponymie*) ;
- la *méronymie* : quand le premier est une partie du second (exemple : le doigt est une partie de la main qui est une partie du corps).

Analyse de concepts formels

- Voici comment cette théorie définit les notions de *contexte*, *concept*, *extension* et *intension*.
- Un *contexte* est un triplet (G, M, I) où G (objets) et M (traits ou attributs) sont des ensembles et $I \subseteq G \times M$. $(g, m) \in I$ signifie que « g possède l'attribut m »;
- pour $G_1 \subset G$, $M_1 \subset M$ les dérivés G_1^I et M_1^I sont définis par

$$G_1^I = \{m \in M \mid \forall g \in G_1, (g, m) \in I\}$$
$$M_1^I = \{g \in G \mid \forall m \in M_1, (g, m) \in I\}$$

  autrement dit : G_1^I est l'ensemble des attributs partagés par tous les éléments de G_1 ; M_1^I est l'ensemble des objets possédant tous les attributs de M_1 ;
- un *concept* (X, N) de (G, M, I) est un couple $X \subseteq G, N \subseteq M$ tel que $X^I = N$ et $N^I = X$. On dit que X est l'*extension* du concept et N son *intension*.

Analyse de concepts formels

- À partir d'un objet g on peut obtenir son *concept d'objet* $(\{g\}^{II}, \{g\}^{I})$;
- à partir d'un attribut m on peut obtenir son *concept d'attribut* $(\{m\}^{I}, \{m\}^{II})$;
- on a $X \subseteq X^{II}$, $N \subseteq N^{II}$, $X^{I} = X^{III}$, $N^{I} = N^{III}$;
- on a un ordre partiel $(X_1, N_1) \leq (X_2, N_2)$ induit par l'inclusion dans G ($\Leftrightarrow$ l'inverse de l'inclusion dans N) : plus un concept est grand (= large), plus il contient des objets, moins il contient des attributs ;
- le concept d'objet de g est le plus petit concept dont l'extension contient g / le concept d'attribut de m est le plus grand concept dont l'intension contient m ;
- muni de l'ordre $\leq$, un contexte a une structure de *treillis* (plus précisément, de *treillis de Galois* pour la relation binaire I).

Les ontologies et les domaines de connaissance

- En informatique, pour associer des mots à des concepts et définir des relations entre eux, on utilise les *ontologies*.
- Une ontologie comporte des *concepts*, des *relations* entre les concepts, des *attributs*, une *hiérarchie de concepts*, une *hiérarchie de relations* définie à partir de la hiérarchie de concepts, et un ensemble de types (chaîne, entier, etc.).

  Mathématiquement parlant :

  1. Une ontologie $\mathcal{O} := (C, \leq_C, R, \sigma_R, \leq_R, A, \sigma_A, T)$
  2. où C, R, A, T sont les concepts, relations, attributs, types ;
  3. $\sigma_R : R \to C^+$, $\sigma_A : A \to C \times T$ sont les signatures des relations et des attributs ;
  4. $\leq_C$ munit C d'une structure de treillis ;
  5. $\leq_R$ est un ordre partiel de R, défini de la manière suivante : $r_1 \leq_R r_2 \Rightarrow |\sigma_R(r_1)| = |\sigma(r_2)|$ et $\pi_i(\sigma_R(r_1)) \leq_C \pi_i(\sigma_R(r_2))$ pour tout i (ici π_i est la i-ème projection).
- Exemple : concepts `rivière`, `ville`, relation `traverse()`, signature $\sigma_R(\texttt{traverse}) = (\texttt{rivière}, \texttt{ville})$.

Les ontologies et les domaines de connaissance

- À l'aide du λ-calcul on définit des schémas d'axiomes, et on en munit l'ontologie. Exemple : l'axiome $\lambda P, Q.\mathrm{disjoint}(P, Q)$, peut être décrit dans le formalisme de logique du premier ordre $\lambda P, Q.\forall x(P(x) \to \neg Q(x))$ et donc, par exemple, si x est rivière il ne peut pas être montagne.
- Si on veut peupler une ontologie à partir du texte, il faut associer des mots aux concepts. On définit un *lexique* $\mathcal{L}$ pour une ontologie $\mathcal{O}$ comme étant $\mathcal{L} := (S_C, S_R, S_A, \mathrm{Ref}_C, \mathrm{Ref}_R, \mathrm{Ref}_A)$ où S_C, S_R, S_A sont des lexèmes pour les concepts, relations et attributs, et $\mathrm{Ref}_* \subseteq S_* \times *$ des *références lexicales* pour les concepts, relations et attributs.
- Ainsi, par exemple, quand on a des synonymes (voiture, auto, bagnole, caisse) on pourra dire que $\mathrm{Ref}_C^{-1}(\texttt{voiture}) = \{\text{voiture, auto, bagnole, caisse}\}$, où `voiture` dénote le concept de voiture.

Les ontologies et les domaines de connaissance

- Une *base de connaissances* $\mathcal{K}$ pour une ontologie $\mathcal{O}$ est un quadruplet $(I, \iota_C, \iota_R, \iota_A)$ où I est un ensemble d'*instances* (ou *objets*) et ι_* des fonctions des instantiation : $\iota_C : C \longrightarrow 2^I, \iota_R : R \longrightarrow 2^{I^+}, \iota_A : A \longrightarrow 2^I \times \text{valeurs}(T)$;
- Même question que tout à l'heure : trouver des entités lexicales pour nos instances. Même réponse :
- Un *lexique d'instances* pour une base de connaissances $\mathcal{K}$ est une paire (S_I, R_I) où S_I sont des signes pour les instances et $R_I \subseteq S_I \times I$ est une relation qui associe des références lexicales aux instances.

Les ontologies et les domaines de connaissance

- Il est temps de profiter de la structure de treillis de l'ontologie pour obtenir plus d'instances (exemple : on sait qu'un labrador est un toutou, et on a Bill, une instance de labrador, il faudrait que ce soit également une instance de toutou), idem pour les relations et les attributs. Si $c \in C$ est un concept, on définit son extension $[\![c]\!]_{\mathcal{K}}$ dans $\mathcal{K}$ comme $[\![c]\!]_{\mathcal{K}} \subseteq I$ par la construction récursive suivante :
  - condition initiale : $[\![c]\!]_{\mathcal{K}} \leftarrow \iota_C(c)$;
  - pour tout $c' <_C c$, $[\![c]\!]_{\mathcal{K}} \leftarrow [\![c]\!]_{\mathcal{K}} \cup [\![c']\!]_{\mathcal{K}}$.
- On peut également définir un concept ou une relation intensionnellement. Exemple : le concept `nombre_pair` se définit par $\{n \in \mathbb{Z} \mid \exists p$ tel que $n = 2p\}$.

Alimentation d'ontologie

- Extraire de la connaissance à partir de textes dans le but d'alimenter une ontologie revient à extraire les couches successives suivantes d'information :
  1. des termes (fleuve, rivière, pays, ville, capitale, ...) et les entités nommées (Brest, IMT Atlantique, Laury Thilleman, etc.) ;
  2. des synonymes { fleuve, rivière, cours d'eau, ... } ;
  3. des concepts `rivière`, `capitale`, `ville`, ... (un concept est une paire de définitions intentionnelle et extensionnelle, ainsi qu'un lexème référent) ;
  4. des hiérarchies de concepts $\texttt{capitale} \leq_C \texttt{ville}$;
  5. des relations entre concepts $\texttt{traverse}(\texttt{rivière},\texttt{pays})$;
  6. des hiérarchies de relations $\texttt{est\_capitale} \leq_R \texttt{est\_situé\_dans}$;
  7. des schémas d'axiomes $\texttt{disjoint}(\texttt{rivière},\texttt{ville})$;
  8. des axiomes logiques généraux : $\forall x(\texttt{pays}(x) \rightarrow \exists y\, \texttt{capitale\_de}(y,x)) \wedge \forall z(\texttt{capitale\_de}(z,x) \rightarrow y = z)$.

Les bases

- Les *logiques de description* sont des variantes de la logique du 1er ordre qui forment des compromis sur deux tableaux :
  1. elles ajoutent des nouvelles notations pour améliorer l'expressivité du langage (par ex. la possibilité de dire qu'il existe exactement n éléments ayant telle propriété) ;
  2. elles sont moins puissantes que la logique du 1er ordre, ce qui les rend décidables.

Les bases

- On a trois types d'objets :
  1. des *individus* (ce qui correspond aux constantes de la logique du 1er ordre),
  2. des *concepts* ou *classes* (des prédicats unaires, dont l'interprétation ensembliste correspond à des ensembles d'individus),
  3. des *rôle* ou *propriétés* (des prédicats binaires sur les individus) ;

Les bases

- On définit trois types de formules :
  1. celles de l'*ABox*, « A » comme « assertion », qui décrivent des concepts et des rôles (prédicats unaires et binaires sur des constantes) ;
  2. celles de la *TBox*, « T » comme « terminologie », qui décrivent des relations entre concepts ;
  3. celles de la *RBox*, « R » comme « relation », qui décrivent une hiérarchie des rôles, des compositions de rôles et des relations entre rôles, comme l'exclusion mutuelle, la réflexivité, la symétrie, la transitivité, etc.

ABox

- Une *assertion de concept* est un prédicat unaire du type ÉlèveTélécom(xavier).
- Une *assertion de rôle* est un prédicat binaire Père(andré,mathilde).
- Dans les logiques de description on ne fait pas l'*hypothèse de l'unicité des noms* : deux individus de même nom peuvent avoir le même référent, on écrira cloclo $\approx$ claudeFrançois, et samsonFrançois $\not\approx$ claudeFrançois.

TBox

- Puisqu'on peut interpréter les *concepts* (prédicats unaires) comme des ensembles, on peut aussi utiliser des relations de théorie d'ensembles :
- Mère notations]$\sqsubset$ (inclusion de concept)$\sqsubset$ Parent, qui équivaut à $\forall X\,\mathrm{Mere}(X) \rightarrow \mathrm{Parent}(X)$;
- Personne notations]$\equiv$ (équivalence de concept)$\equiv$ Humain, qui équivaut à $\forall X\,\mathrm{Personne}(X) \leftrightarrow \mathrm{Humain}(X)$.

RBox

- La *hiérarchie des concepts* induit une *hiérarchie des rôles* :
- ParentDe $\sqsubset$ AncêtreDe, qui équivaut à $\forall X, Y\, \mathrm{ParentDe}(X, Y) \rightarrow \mathrm{AncêtreDe}(X, Y)$.
- Une relation binaire peut se combiner avec une autre : sachant que le frère d'un père est un oncle, on peut écrire
- FrèreDe $\circ$ ParentDe $\sqsubset$ OncleDe, qui équivaut à $\forall X, Y, Z\, \mathrm{FrereDe}(X, Y) \wedge \mathrm{ParentDe}(Y, Z) \rightarrow \mathrm{OncleDe}(X, Z)$.
- Enfin, on peut donner des caractéristiques générales des concepts : *Disjoint*(ParentDe, EnfantDe), ce qui équivaut à $\forall X, Y\, \mathrm{ParentDe}(X, Y) \rightarrow \neg \mathrm{Parent}(Y, X)$.

Restriction existentielle

- Toujours en mimant la théorie des ensembles, on peut écrire notations]$\sqcap$ (intersection de concept)$\mathrm{Mère} \equiv \mathrm{Femme} \sqcap \mathrm{Parent}$, qui équivaut à $\forall X\, \mathrm{Mère}(X) \leftrightarrow \mathrm{Femme}(X) \wedge \mathrm{Parent}(X)$,
- ou notations]$\sqcup$ (union de concept)$\mathrm{Parent} \equiv \mathrm{Père} \sqcup \mathrm{Mère}$, qui équivaut à $\forall X\, \mathrm{Parent}(X) \leftrightarrow \mathrm{Mère}(X) \vee \mathrm{Père}(X)$,
- la négation correspond au complémentaire d'un ensemble : ¬Célibataire est équivalent à Marié.
- L'ensemble complet correspond à un prédicat qui est toujours vrai, on l'écrit notations]$\top$ (ensemble complet)$\top$, pour exprimer une partition on écrira $\top \sqsubset \mathrm{Homme} \sqcup \mathrm{Femme}$.
- L'ensemble vide correspond à un prédicat qui est toujours faux, on l'écrit notations]$\bot$ (ensemble vide)$\bot$, pour exprimer une exclusion mutuelle on écrira $\mathrm{Homme} \sqcap \mathrm{Femme} \sqsubset \bot$.

Restriction existentielle

- Imaginons qu'on a un rôle Parent(X, Y) et que l'on cherche à caractériser les X qui sont parents. Il s'agit donc de dire « je cherche les X pour lesquels $\exists Y$ tel que Parent(X, Y) », on écrira $\exists \mathrm{Parent}.\top$;
- le $\top$ signifie qu'on prend les X pour lesquels il existe un Y, sans les filtrer davantage.
- Si je cherchais ceux qui sont des parents d'au moins une fille, j'écrirais $\exists \mathrm{Parent}.\mathrm{Fille}$.
- De même, ceux qui n'ont que des filles : $\forall \mathrm{Parent}.\mathrm{Fille}$.
- Mais que signifie alors $\forall \mathrm{Parent}.\top$?

Restriction existentielle

- Pour le savoir, prenons les définitions formelles des sémantiques de ces notations.
- Soit R un rôle, et R^{I} son interprétation. Rappelons que dans une interprétation ensembliste, R^{I} devient un ensemble de paires d'éléments $R^{\mathsf{I}} = \{(x^{\mathsf{I}}, y^{\mathsf{I}}), \ldots\}$ du domaine Δ. On dira que y^{I} est un R-*successeur* de x^{I} si la paire $(x^{\mathsf{I}}, y^{\mathsf{I}})$ appartient à R^{I}.
- Soit C un concept (et donc C^{I} est un sous-ensemble de Δ). Alors l'interprétation de $\exists R.C$notations]$\exists R.C$ (quelques successeurs) est $\{x^{\mathsf{I}} \mid$ quelques successeurs de x^{I} sont dans $C^{\mathsf{I}}\}$.
- Et celle de $\forall R.C$notations]$\forall R.C$ (tous les successeurs) est $\{x^{\mathsf{I}} \mid$ tous les successeurs de x^{I} sont dans $C^{\mathsf{I}}\}$.
- Donc, quelle est l'interprétation de $\forall \mathrm{Parent}.\top$?

Restrictions au-plus et au-moins, réflexivité

- De même que l'on peut demander au moins un successeur ou tous les successeurs, on peut aussi spécifier « au moins n successeurs » : $\geq nR.C$notations]$\leq nR.C$ (au moins n successeurs), ainsi qu'« au plus n successeurs » : $\leq nR.C$notations]$\leq nR.C$ (au plus n successeurs).
- Une autre notation permet de décrire l'ensemble des éléments pour lesquels un rôle R est réflexif : $\exists R.Self$notations]$\exists R.Self$ (rôle réflexif).
- Une manière de décrire un concept est en donnant explicitement ses membres : au lieu de Parent(jacques,julie), on peut aussi écrire $\{\text{jacques}\} \sqsubset \exists \text{Parent}.\{\text{julie}\}$.
- On note R^- le *rôle inverse* de R, par exemple Parent$^-$ est équivalent à Enfant puisque $\forall X, Y\, \text{Parent}(X, Y) \leftrightarrow \text{Enfant}(Y, X)$.
- Enfin, on note U le *rôle universel*, c'est-à-dire celui qui associe chaque élément à chaque autre élément (y compris lui-même).
- Enfin, on note R^+ la *clôture transitive* de R, c'est-à-dire le plus petit rôle transitif contenant R.

Ontologies DL $\mathcal{SROIQ}$

- Une *ontologie* est la formalisation d'un domaine de connaissances dans un langage de représentation de connaissances.
- Une *ontologie DL* est une ontologie représentée dans le langage d'une logique de description.
- $\mathcal{SROIQ}$ est une logique de description spécifique que nous allons décrire dans la suite.
- Si N_C est un concept quelconque, N_R un rôle quelconque et N_I un individu quelconque, alors on définit de manière itérative dans $\mathcal{SROIQ}$:
  1. une *expression de concept* $\mathbf{C}$ par $\mathbf{C} ::= N_C \mid (\mathbf{C} \sqcap \mathbf{C}) \mid \mathbf{C} \sqcup \mathbf{C}) \mid \neg\mathbf{C} \mid \top \mid \bot \mid \exists \mathbf{R}.\mathbf{C} \mid \forall \mathbf{R}.\mathbf{C} \mid \geq n\,\mathbf{R}.\mathbf{C} \mid \leq n\,\mathbf{R}.\mathbf{C} \mid \exists \mathbf{R}.Self \mid \{N_I\}$,
  2. une *expression de rôle* $\mathbf{R}$ par $\mathbf{R} ::= U \mid N_R \mid N_R^-$.
- Les axiomes d'une ontologie $\mathcal{SROIQ}$ sont des types suivants :
  1. ABox : $\mathbf{C}(N_I)$, $\mathbf{R}(N_I, N_I)$, $N_I \approx N_I$, $N_I \not\approx N_I$,
  2. TBox : $\mathbf{C} \sqsubset \mathbf{C}$, $\mathbf{C} \equiv \mathbf{C}$,
  3. RBox : $\mathbf{R} \sqsubset \mathbf{R}$, $\mathbf{R} \equiv \mathbf{R}$, $\mathbf{R} \circ \mathbf{R} \sqsubset \mathbf{R}$, $Disjoint(\mathbf{R}, \mathbf{R})$.

Propriétés de $\mathcal{SROIQ}$

- Voici les propriétés de la logique de description $\mathcal{SROIQ}$:
  1. $\mathcal{S}$ (également appelé $\mathcal{ALCR}^+$) : ($\mathcal{AL}$) présence de noms de concepts et de rôles, de $\top$, du constructeur $\sqcap$ (conjonction), du quantificateur universel, ($\mathcal{C}$) de la négation de concept, ($\mathcal{R}^+$) de la clôture transitive ;
  2. $\mathcal{R}$: inclusion de rôles, réflexivité, irréflexivité, exclusion de rôles ;
  3. $\mathcal{O}$: possibilité de décrire un concept extensionnellement ($\{N_{I,1}, \ldots, N_{I,n}\}$) ;
  4. $\mathcal{I}$: possibilité d'avoir des rôles inverses (R^-) ;
  5. $\mathcal{Q}$: possibilité d'avoir des restrictions pleinement quantifiées ($\leq n$, $\geq n$).
- D'autres logiques de description sont éventuellement dotées des propriétés suivantes :
  6. $\mathcal{H}$: hiérarchie des rôles $R_1 \sqsubset R_2$, $\mathcal{H}$ est plus faible que $\mathcal{R}$;
  7. $\mathcal{N}$: restrictions de cardinalité plus faibles que $\mathcal{Q}$.

- La norme *OWL* 2 du consortium *WWW* correspond à la logique $\mathcal{SROIQ}$, alors que la norme plus faible OWL-DL correspond à $\mathcal{SHOIN}$.
- Le logiciel *Protégé* d'édition d'ontologies est basé sur cette dernière.

Donnée ≠ Information ≠ Connaisssance

- Nous utilisons tous les termes « donnée », « information », « connaissance » dans la vie courante.
- Mais ils ont aussi un sens plus technique, utilisé en intelligence artificielle :

- les *données* sont des ensembles ordonnés de nombres, provenant de capteurs ou de programmes de simulation, de générateurs de données aléatoires, etc. ;
- les *informations* sont des données auxquelles on a attaché un sens : on sait ce que représente tel nombre ou tel ensemble d'octets ou telle chaîne de caractères ;
- les *connaissances* sont des motifs ou des tendances qu'on extrait des informations, dans le but de faire des prédictions.

- Exemple : le départ d'un train de la gare de Brest.

Donnée ≠ Information ≠ Connaisssance

- Un *domaine de connaissances* est un ensemble de données, informations et connaissances autour d'un thème donné (par exemple : la géographie est un domaine de connaissances).
- Une *base de connaissances* est une structure informatique/mathématique qui permet de stocker des données, informations et connaissances. Point commun avec les SGBD : on peut interagir avec une base de connaissances par le biais de requêtes/réponses.

Graphes conceptuels

- Dans ce chapitre, qui est un avant-goût du module de logique INF 424, on va représenter une base de connaissances sous forme de graphe.
- On utilisera pour cela un genre de graphe bien particulier : les *graphes conceptuels*.
- Pour faire le lien entre le domaine que l'on veut décrire et la structure de graphe, on définit la notion de *vocabulaire* :

Graphes conceptuels

Definition

Un *vocabulaire de graphe conceptuel* est un triplet $(T_C, T_R, \mathcal{I})$, où T_C est l'ensemble de *types de concept*, il est partiellement ordonné et a un plus grand élément $\top$; T_R est l'ensemble de *symboles de rélation*, il est également partiellement ordonné et chacun de ses éléments a une *arité* ≥ 1 ; $\mathcal{I}$ est l'ensemble des *marqueurs d'individu*. On ajoute à $\mathcal{I}$ l'élément $*$ (*marqueur générique*) avec la propriété $\forall i \in \mathcal{I}, i \geq *$. Ces ensembles sont mutuellement disjoints.

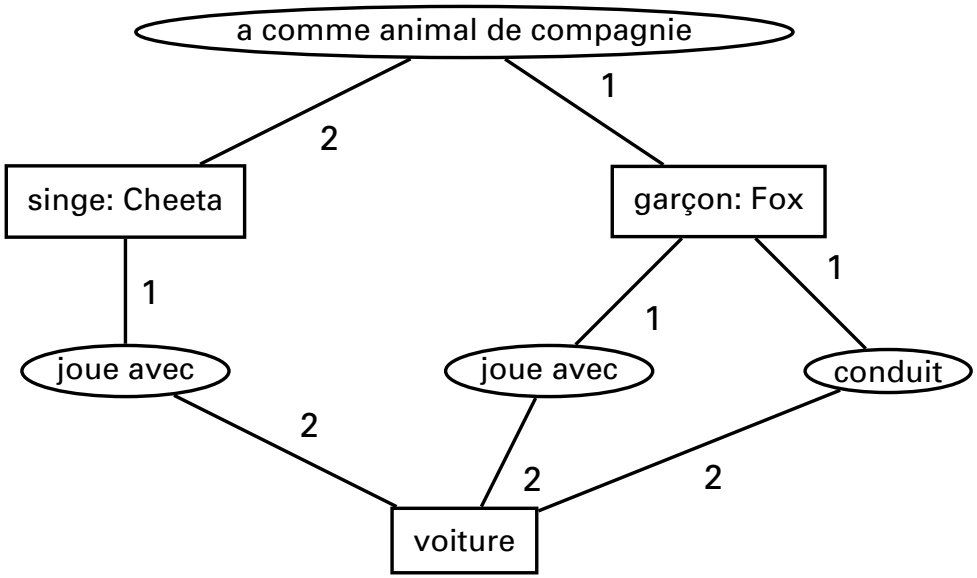
Definition

Un *graphe conceptuel de base* sur un vocabulaire $\mathcal{V}$ donné, est la donnée d'un graphe G biparti, de partitions C et R, non-orienté avec éventuellement des arêtes multiples et d'une fonction ℓ, définis de la manière suivante :

— les sommets sont des *concepts* C et des *rélations* R
— l'image de $c \in C$ par ℓ est une paire (t, i) avec $t \in T_C$ et $i \in \mathcal{I}$;
— pour $r \in R$, $\ell(r) \in T_R$;
— le degré de $r \in R$ est égal à l'arité de $\ell(r)$;

Graphes conceptuels

Prenons un exemple :



La photo est interprétée par « Fox est un garçon, Cheeta est un singe. Cheeta est l'animal de compagnie de Fox. Ils jouent ensemble avec une voiture-jouet, conduite par Fox. »
Ici, « singe », « garçon » et « voiture » sont des types de concepts, « joue avec » et « conduit » sont des rélations d'arité 2 (sujet, COD), « Cheeta » et « Fox » sont des marqueurs d'individu, alors que le sommet du bas est en fait « voiture : $*$ » (on ne note pas le marqueur générique).

Graphes conceptuels

Definition

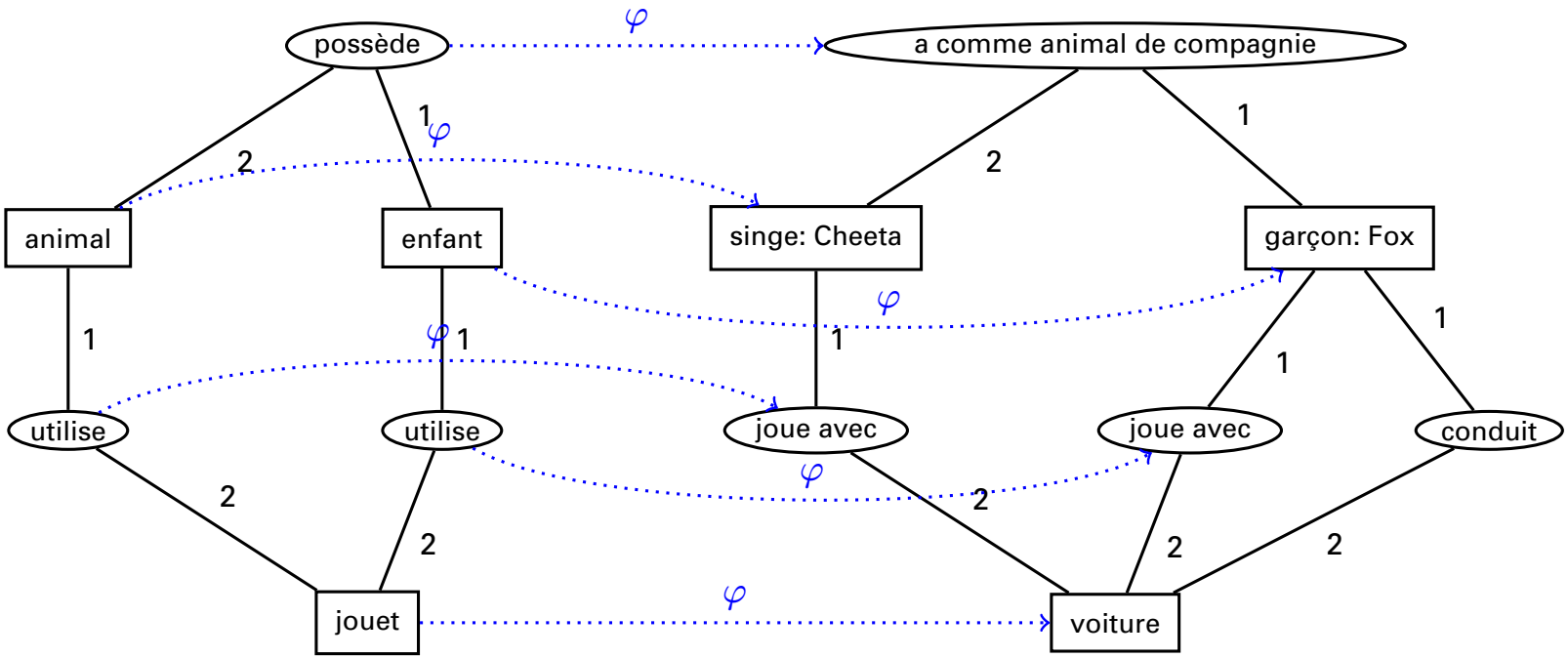
On définit un *ordre des concepts* à partir de l'ordre des types de concept : si $(t,i),(t',i') \in C$, $(t,i) \leq (t',i')$ si et seulement si $t \leq t'$ et $i \leq i'$. Si $t \leq t'$ on dira que t' est un *hypéronyme* de t.

(À noter que $i \leq i'$ ne peut arriver que si $i = i'$ ou si $i' = *$, puisque les i, i' différents de $*$ ne peuvent être comparés.)

Definition

Si G et G' sont des graphes conceptuels sur le même vocabulaire, un *homomorphisme de graphes conceptuels* est un homomorphisme de graphes φ qui envoie C dans C', R dans R', $\mathcal{I} \cup \{*\}$ dans $\mathcal{I}' \cup \{*\}$ et qui est tel que $\varphi(c) \leq c$, $\varphi(r) \leq r$ et φ respecte les numéros des arêtes.

Graphes conceptuels



Graphes conceptuels

Definition

Soient G et G' deux graphes conceptuels sur le même vocabulaire. On définit la *rélation de subsumption* $\succeq$ de la manière suivante : $G \succeq G'$ ssi il existe un homomorphisme de graphe conceptuel $\varphi : G \to G'$.

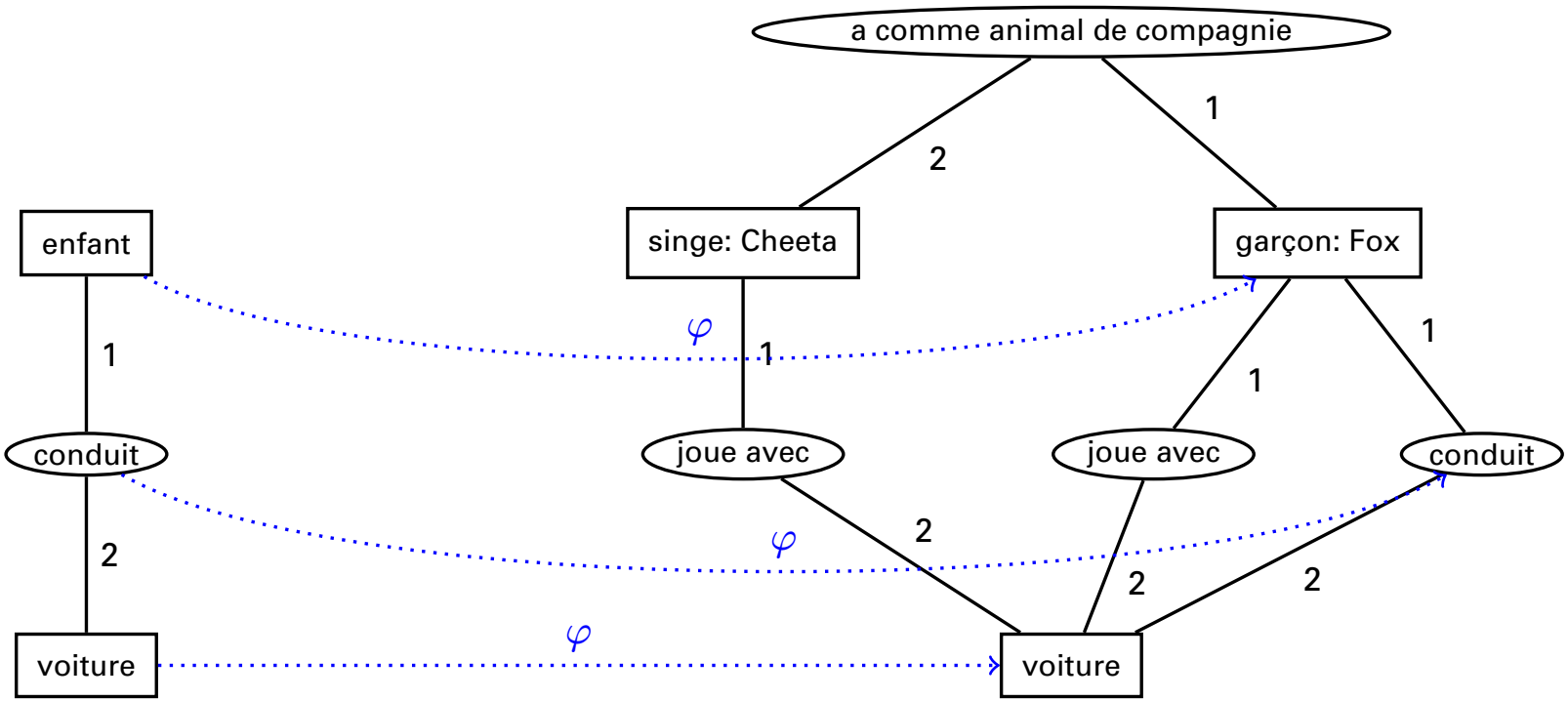
Definition

Soit Q et G des graphes conceptuels sur le même vocabulaire. On dira que Q est une requête acceptée par G si $Q \succeq G$. Les résultats de la requête sont les images $\varphi(Q)$ pour tout homomorphisme $\varphi : Q \to G$.

On peut imaginer diverses situations où la notion de requête peut être utile :
(1) on considère que G est un graphe conceptuel qui décrit la réalité et on voudrait savoir si Q est « conforme » à cette réalité (on dira en INF 424 que Q est une « conséquence » de G) ;
(2) G décrit l'image ci-dessus, Q correspond à une requête d'image

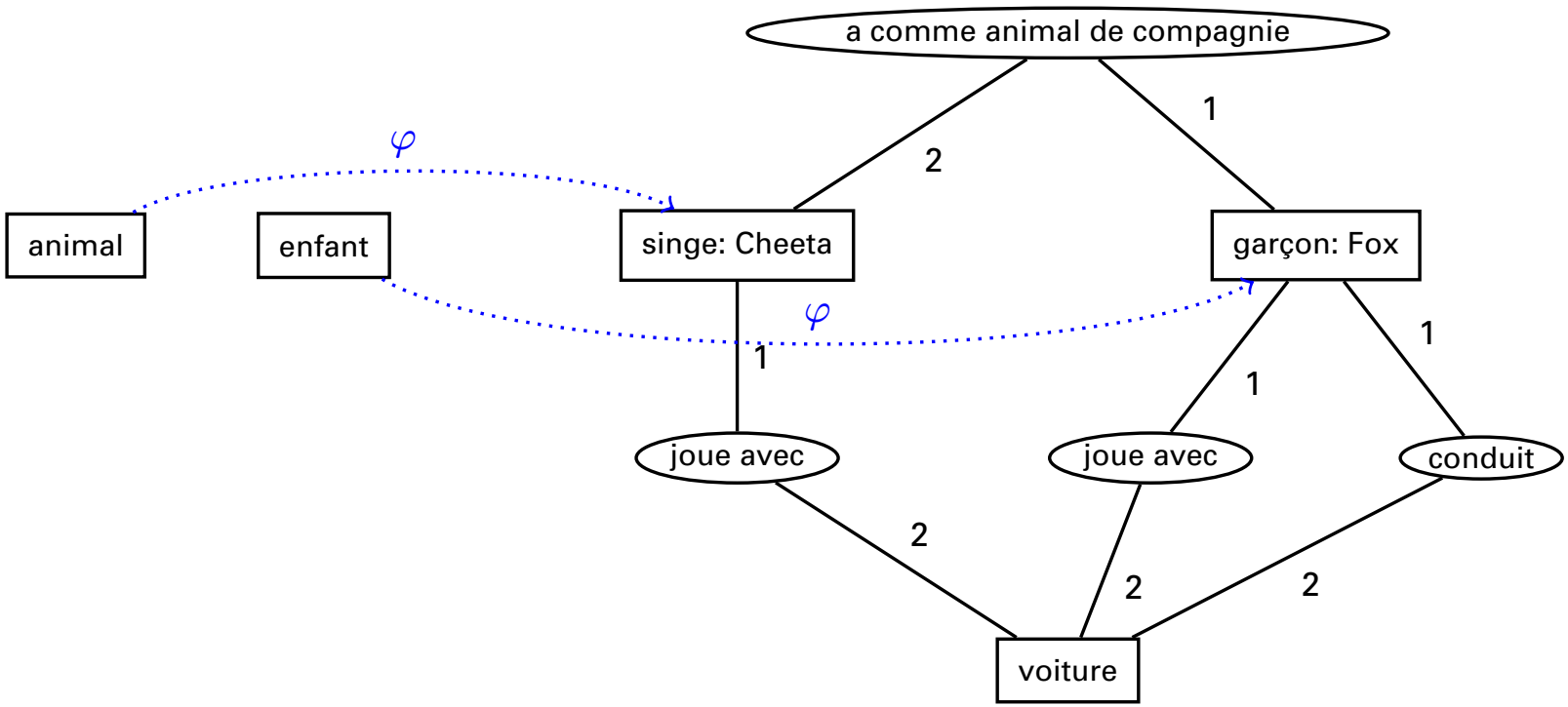
Graphes conceptuels

Premier exemple : « Un enfant conduit une voiture ».



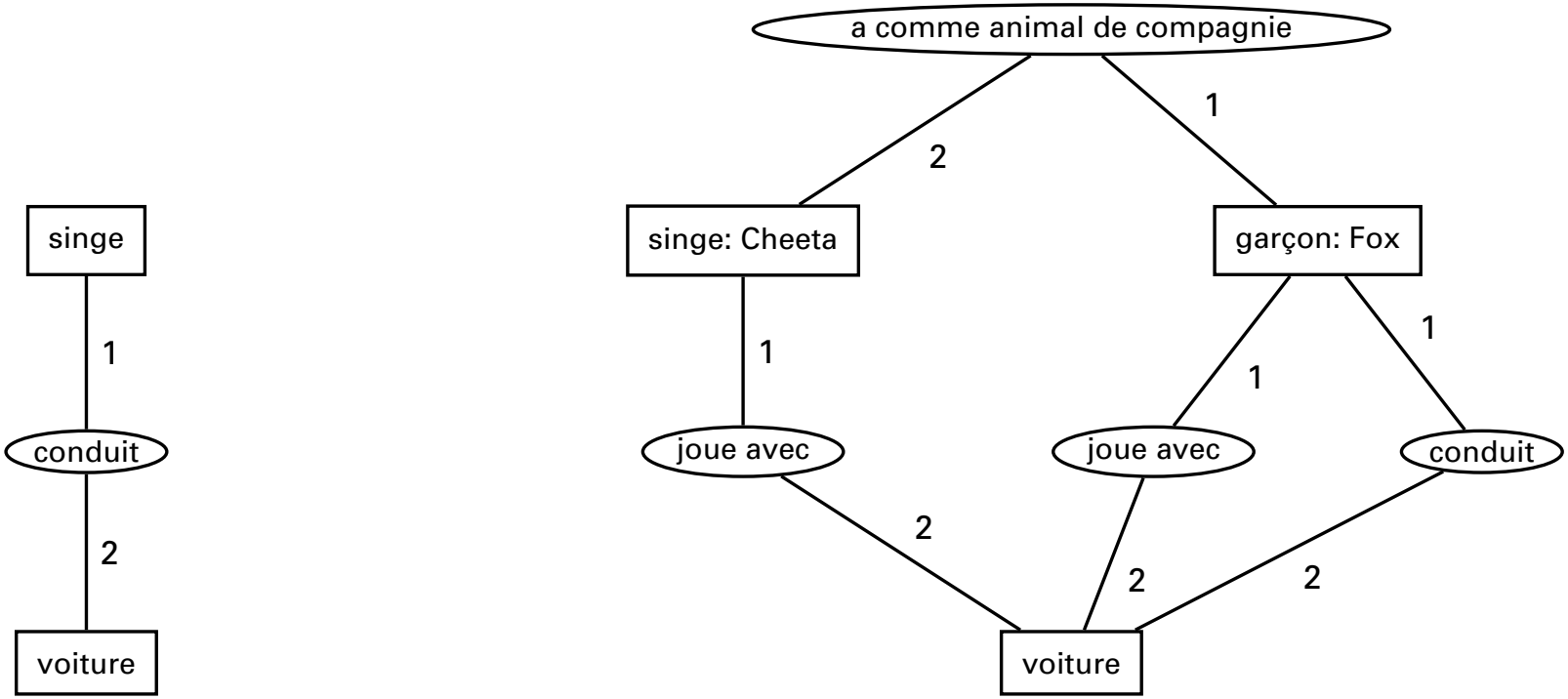
Graphes conceptuels

Deuxième exemple : « Un enfant possède un animal ».



Graphes conceptuels

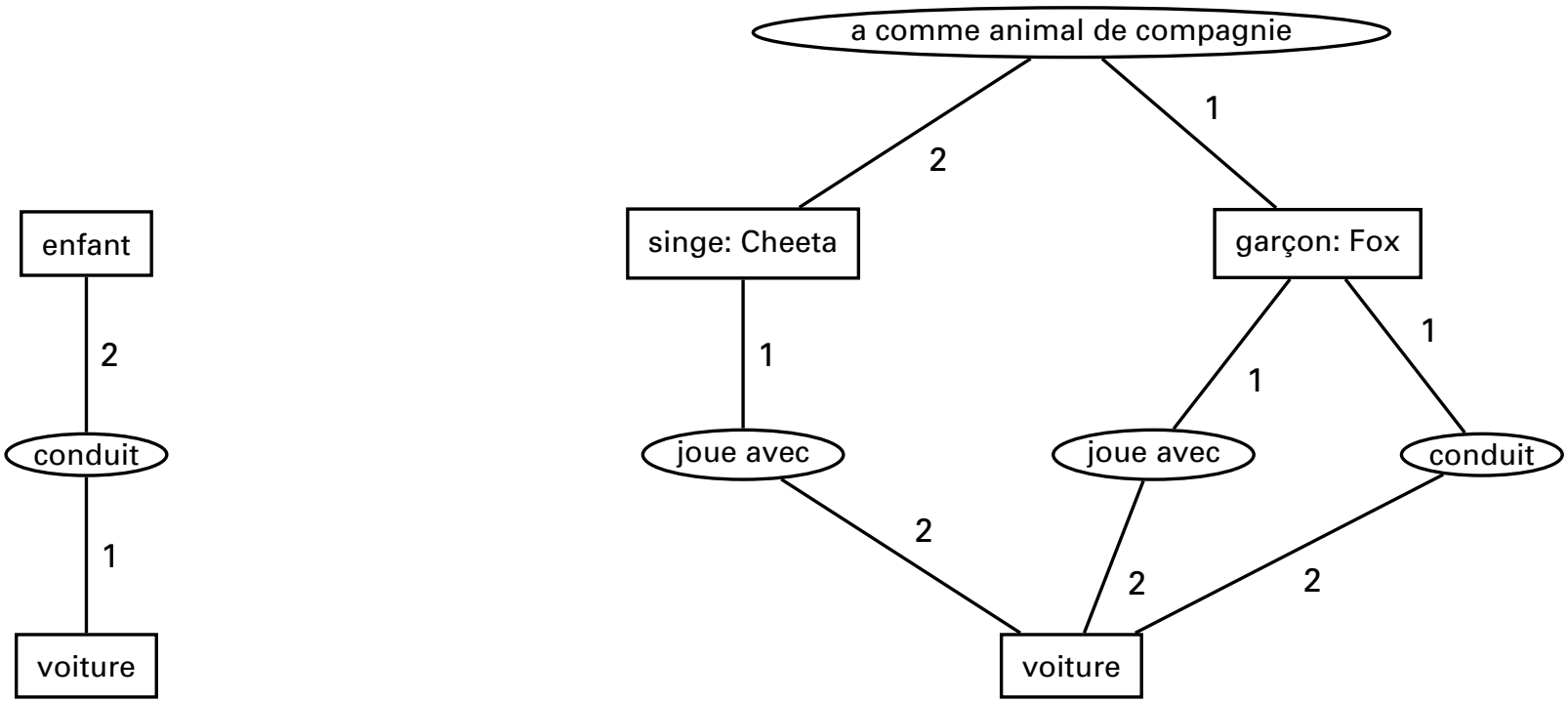
Troisième exemple : « Un singe conduit une voiture ».



Impossible puisque l'arête entre « singe » et « conduit » ne peut être envoyée à « singe : Cheeta »—« conduit » qui n'existe pas. De même, on ne peut pas envoyer « conduit » vers « joue avec » puisqu'elles sont incomparables.

Graphes conceptuels

Quatrième exemple : « Une voiture conduit un enfant ».



Impossible puisque l'application qui envoie « enfant » à « garçon : Fox », « conduit » à « conduit » et « voiture » à « voiture », n'est pas un homomorphisme de graphes contextuels puisqu'elle ne respecte pas les numéros des arêtes.

Exemple de problème : la désambiguïsation de mot

- (Officiellement, depuis 1990 ce mot s'écrit « désambigüisation ».)
- Il s'agit de choisir entre les différents sens d'une instance de mot polysémique (*hier mon fils a volé pour la première fois* : voleur ou aviateur ?).
- Première approche [Gale *et al.* 1992] : *classification bayésienne*. On considère w un mot ambigu, s_* ses sens, c_* les contextes de w dans un corpus, v_* les mots environnants. Pour un c donné, on cherche à maximiser $P(s_i \mid c)$.
  Approche bayésienne : on cherche

$$\arg\max P(s_* \mid c) = \arg\max [\log P(c \mid s_*) + \log P(s_*)].$$

  Hypothèse « naïve Bayes » : $P(c \mid s_k) = \prod_{v_j \in c} P(v_j \mid s_k)$.
  On procède par estimation de la vraisemblance maximale :

$$\hat{P}(v_j \mid s_k) = \frac{\#\{v_j \in \text{contexte}(s_k)\}}{\#\{w\}},\ \hat{P}(s_k) = \frac{\#\{s_k\}}{\#\{w\}}$$

  dans le corpus d'apprentissage.

La désambiguïsation de mot

- Autre approche [Lesk 1986] : se servir des différentes définitions du mot dans un dictionnaire. On prend les mots qui apparaissent dans chacune des définitions d'un mot, et on les compare avec ceux des définitions des mots voisins. [Walker 1987] utilise un thésaurus. [Resnik 1995] utilise les synsets de WordNet.
- WordNet est une base de données lexicale libre. Pour chaque mot, WordNet fournit des synsets (= ensembles de synonymes) et une courte description. Exemple (pour *car*) :
  - car, auto, automobile, machine, motorcar – (4-wheeled motor vehicle; usually propelled by an internal combustion engine; he needs a car to get to work),
  - car, railcar, railway car, railroad car – (a wheeled vehicle adapted to the rails of railroad; three cars had jumped the rails),
  - car, gondola – (car suspended from an airship and carrying personnel and cargo and power plant),
  - car, elevator car – (where passengers ride up and down; the car was on the top floor),
  - cable car, car – (a conveyance for passengers or freight on a cable railway; they took a cable car to the top of the mountain).

Extraction de l'information

- IE (*Information Extraction*) : recherche d'une information dans un document ou dans un corpus, à ne pas confondre avec IR (*Information Retrieval*) : recherche de documents contenant une certaine information.
- Il s'agit de récupérer des entités, des relations entre entités, des qualificatifs d'entités, ainsi que des structures plus élaborées comme des tableaux et des listes.
- Une *entité* est typiquement une phrase nominale comportant entre un et un petit nombre de mots. Une *entité nommée* est un nom propre, un acronyme, une abréviation, un terme temporel, une expression monétaire ou numérique. En médécine une EN sera un nom de maladie, de protéine, de gène, etc.
- Une *relation entre entités* est un prédicat. Certains parlent de *faits* et d'*évènements*.
- Une branche du TAL, le *Semantic Role Labeling* fait l'inverse : étant donné un prédicat, trouver ses différents arguments sémantiques (les réponses aux questions qui-quoi-quand-où-comment, etc.).

IE, méthodes basées sur les règles

- Les règles peuvent être des simples expressions régulières ou alors des motifs plus complexes, faisant intervenir les mots, les tags POS, des dictionnaires externes, des annotations provenant d'extractions précédentes.
- Une règle typique est du type « motif contextuel $\longrightarrow$ action ».
- L'action peut être le balisage de l'entité, ou d'un groupe d'entités et de relations entre elles.
- L'ordre d'applications des règles est important.
- Les règles peuvent être écrites à la main ou apprises à partir d'un corpus d'apprentissage.

Apprentissage de règles d'IE

- Première approche : le bottom-up.
- [Califf & Mooney 2003] On prend une instance d'entité qui ne soit pas couverte par une règle. À partir de cette instance on crée une règle du type $x_{i-w} \dots x_{i-1} x_i \dots x_{i+w} \longrightarrow T$ où T est le tag de x_i. On enlève des tokens ou on les remplace par des tokens plus généraux, et on voit ce que ça donne au niveau du corpus d'apprentissage.
- Deuxième approche : le top-down.
- On part d'une règle très générale et on la spécialise. Par exemple, on part d'une règle R_0 qui pose des conditions au niveau de $2w$ tokens. On va considérer progressivement des règles
  1. $\mathcal{R}_1$ qui diffèrent de R_0 en un des tokens ;
  2. pour $L = 2$ à $2w$, des règles $\mathcal{R}_L$ formées par l'intersection de deux règles de $\mathcal{R}_{L-1}$ qui concordent sur $L-2$ conditions et diffèrent en une.

  À chaque fois on regarde la couverture des règles et on ne garde que celles qui dépassent un seuil minimum s.
- [Krishnamurty *et al.* 2008] définissent un modèle de données relationnel et une algèbre d'opérateurs qui permettent l'optimisation des requêtes. Ils valident leurs résultats sur des corpus de blogs.

IE, méthodes statistiques

- On peut procéder par tokens (mots) ou par segments.
- Dans le premier cas, on part d'un suite de tokens $\vec{x} = x_1 \dots x_n$ et on se propose d'associer à chaque token un label $y_i \in \mathcal{Y}$.
- Dans le deuxième cas, on cherche une suite de segments $s_1 \dots s_p$ et à chaque segment $s_j = l_j \dots u_j$ on associe un label y_i.
- Diverses méthodes ont été employées (HMM, MEMM, CRF). Il paraît que les CRF marchent assez bien.

IE, les relations

- Premier cas : on part de deux entités connues et on cherche leur relations : soit E_1, E_2 entités dans $\vec{x}$, trouver toutes les relations dans $\mathcal{Y}$ entre E_1 et E_2.
- On peut écrire des règles (de logique propositionnelle ou de premier ordre) pour détecter les relations.
- On peut utiliser des méthodes à base de propriétés (*features*) :
  1. utiliser des listes de mots prédéfinies ou des classes dans des ontologies ;
  2. utiliser des tags POS (chercher en particulier : les verbes) ;
  3. établir l'arbre syntaxique de la phrase (difficile, et fragile en cas de bruit) ;
  4. établir un graphe de dépendances.

  Une fois les propriétés choisies, on peut utiliser des arbres de décision, ou des SVM.
- Ou alors, on peut comparer directement des arbres ou des graphes en utilisant des méthodes de noyaux.

IE, résolution d'anaphores

- *Le président de la république*$_1$ *a rencontré le serial-killer*$_2$ *dans sa*$_{1 \text{ ou } 2?}$ *cellule. Il*$_{1 \text{ ou } 2?}$ *a avoué éprouver des remords pour ses actes.*
- Une *anaphore* est un mot ou un syntagme qui, dans un énoncé, assure une reprise sémantique d'un précédent segment appelé *antécédent*. (La *cataphore* va dans l'autre sens.) On parle aussi de *coréférences*.
- La *résolution d'anaphores* consiste à identifier les entités du texte qui se réfèrent à la même entité du monde réel.
- Il y a différents types de renvois anaphoriques : pronoms personnels (*il*, *elle*,...), possessifs (*le nôtre*,...), adverbes (*là*, *ainsi*,...), ordinaux (*le second*,...), etc.
- L'approche générale :
  1. identifier/délimiter les phrases/paragraphes où l'on va chercher des antécédents ;
  2. utiliser une série de tests de cohérence (genre, nombre, cas, etc.) pour éliminer un premier lot de candidats ;
  3. pondérer, selon certaines règles, les candidats restant ;
  4. choisir le candidat de plus grand poids.

IE, résolution d'anaphores

- Approches à base de règles : CogNIAC [Baldwin 1995] pose six règles, dans un ordre précis. [Kennedy & Boguraev 1996] pondèrent (positivement) selon dix facteurs, impliquant le tag POS et la fonction syntaxique (sujet, coi, cod, etc.). [Mitkov 1998] applique des poids positifs et négatifs selon certains indicateurs (POS, syntaxe, mais aussi distance entre anaphore et antécédent).
- Approches statistiques : [Soon *et al.* 2001] reprennent l'idée des indicateurs (ils appellent cela des *markables*) et font un apprentissage sur un corpus prétagué.
- Approches sémantiques : selon [Markert & Nissim 2005], alors qu'il y a eu des travaux utilisant un apport sémantique (par exemple : recherche de proximité sémantique dans WordNet, du type *quand mon chien a vu l'os, il l'a mangé*) ceux-ci ne donnent pas de très bons résultats (manque de finesse et de complétude des ontologies).

  Exemple : *Will Quinlan had not inherited a damaged retinoblastoma supressor gene and, therefore, faced no more risk than other children.*

IE, résolution d'anaphores

Réponse à la question : oui, dans Wordnet on a « child < juvenile < person < ... ».
— S: (n) child, kid, youngster, minor, shaver, nipper, small fry, tiddler, tike, tyke, fry, nestling (a young person of either sex) « she writes books for children »; « they're just kids »; « 'tiddler' is a British term for youngster »
direct hypernym / inherited hypernym / sister term
— S: (n) juvenile, juvenile person (a young person, not fully developed)
— S: (n) person, individual, someone, somebody, mortal, soul (a human being) « there was too much for one person to do »
— S: (n) organism, being (a living thing that has (or can develop) the ability to act or function independently)
— S: (n) living thing, animate thing (a living (or once living) entity)
— S: (n) whole, unit (an assemblage of parts that is regarded as a single entity) « how big is that part compared to the whole? »; « the team is a unit »
— S: (n) object, physical object (a tangible and visible entity; an entity that can cast a shadow) « it was full of rackets, balls and other objects »
— S: (n) physical entity (an entity that has physical existence)
— S: (n) entity (that which is perceived or known or inferred to have its own distinct existence (living or nonliving))

Classification supervisée

- Après la phase de prétraitement, on a dégagé les informations souhaitées des documents et on les a intégrées dans leurs représentations.
- Il s'agit maintenant de faire du mining sur ces représentations.
- La *classification supervisée* ou *catégorisation* est une classification des documents dans des classes prédéfinies (par exemple : SPAM ou ¬SPAM ? sport, culture ou économie ?).
- On représente, le plus souvent, les documents par des *vecteurs de propriétés* (features), qui peuvent être pondérés.
- Le modèle du « sac de mots » (*bag-of-words*) consiste à considérer, pour chaque mot w, sa présence (ou sa fréquence) comme une propriété (une dimension de l'espace vectoriel). Dans certains cas, on ajoute aux mots (ou on les remplace par) leurs synonymes / hypéronymes, etc.
- Comme poids du terme t_k dans le document d_j on peut utiliser la fonction *tfidf* :

$$\mathrm{tfidf}(t_k, d_j) = \#\{t_k \in d_j\} \cdot \log \frac{\#\{d_*\}}{\#\{d_* \mid t_k \in d_*\}}.$$

Réduction de la dimensionnalité

- Pb : si chaque mot (ou terme ou concept) du document est une dimension du vecteur, on risque d'en avoir *beaucoup*. Différentes méthodes ont été appliquées pour réduire le nombre de dimensions.
- Une méthode linéaire est le LSI (*latent semantic indexing*) [Deerwester *et al.* 1990] :
- un résultat d'algèbre linéaire dit que :
  1. toute matrice X de rang n peut s'écrire USV^t, où U et V sont orthogonales ($UU^t = I$, $VV^t = I$) et S est diagonale, de termes diagonaux $s_1 \geq s_2 \geq \cdots \geq s_n$;
  2. soit $k < n$, en prenant S' diagonale avec les k premiers termes de S, U' les k premières colonnes de U, V' les k premières colonnes de V, on obtient $X' = U'S'V'$;
  3. alors X' est la matrice de rang k la plus proche (norme de Frobenius) de X.
- On applique cette méthode (appelée « décomposition en valeurs singulières ») à la matrice S des termes $\times$ documents, et on obtient S' de rang k, optimale parmi toutes les matrices de ce rang.

Réduction non-linéaire de la dimensionnalité

- LSI est une méthode *linéaire* de réduction de la dimensionnalité.
- Autres approches, provenant du traitement de l'image : [Tenenbaum *et al.* 2001] utilisent, au lieu de la distance euclidienne, des distances géodésiques sur des variétés topologiques qui collent au mieux aux données (*manifold learning*).
- L'approximation de ces variétés se fait à l'aide de graphes obtenus en liant par des arêtes des points de l'échantillon avec les voisins se trouvant à distance (euclidienne) inférieure à ε. Ensuite, une fois le graphe obtenu, on considère comme distance entre deux points, la longueur du plus court chemin (Dijkstra). C'est l'algorithme *ISOMAP*.
- Après, on réduit la dimension du graphe de manière classique.

Classification supervisée de documents

- Une fois les documents représentés par des vecteurs de dimension raisonnable, on utilise les méthodes classiques de data mining.
- Approche par règles logiques (DNF) écrites manuellement [Hayes *et al.* 1990].
- Approches statistiques : le naïve Bayes est assez robuste, malgré l'hypothèse que les mots soient tous indépendants [Domingos & Pazzani 1997].
- Regression logistique, arbres de décision (ID3, C4.5, CART), règles DNF apprises sur un corpus, méthode de Rocchio, réseaux de neurones, kNN, SVM, AdaBoost, etc.

Classification non supervisée

- La *classification non supervisée* ou *clustering*, consiste à classifier les documents sans connaissance préalable des catégories.
- Il s'agit de comparer toutes les partitions possibles d'un ensemble de documents à l'aide d'une *mesure de similarité* : les documents dans le même groupe doivent être aussi similaires que possible, les groupes entre eux aussi dissimilaires que possible.
- Première approche : *Hierarchical Agglomerative Clustering* :
  1. on met chaque objet dans un cluster séparé ;
  2. on fusionne les clusters les plus similaires ;
  3. on répète le (2) jusqu'à ce que tout soit dans le m̂ cluster.
- [El-Hamdouchi & Willet 1989] Il y a différentes manières de mesurer la similarité des clusters : *single-linkage* (max de similarité entre leurs obj), *complete-linkage* (min), *centre de gravité* (similarité entre centroïdes), *average group* (similarité moyenne), *Ward* (plus petite augmentation de distance du centroïde du nouveau cluster), etc.
- Le pb est que l'on ne peut pas revenir en arrière pour corriger

Clustering

- Deuxième approche : l'algorithme *K-means* :
  1. on prend au hasard k vecteurs et on considère qu'ils sont les centres des clusters : tout vecteur appartient au cluster dont le centre est le + proche ;
  2. on calcule les *centroïdes* : les moyennes des tous les vecteurs de cluster, ils deviennent les nvx centres de cluster et on répartit de nouveau les vecteurs ;
  3. on répète le (2) jusqu'à convergence.
- Le K-means dépend bcp du choix initial.
- À l'aide du LSI (latent semantic indexing) on réduit le nombre de dimensions et on obtient, non pas des centroïdes, mais des *médoïdes* : des documents artificiels qui jouent le même rôle.

Clustering avec des ontologies

- [Hotho *et al.* 2003] décrivent un clustering enrichi par des accès à des ontologies ou à Wordnet. Ils décrivent plusieurs stratégies :
  1. « add » à tout vecteur de termes (un vecteur représentant le document, dont les coordonnées correspondent à la fréquence des termes) on ajoute un vecteur de concepts pour ceux qui ont pu être identifiés,
  2. « repl » quand un terme renvoie à un concept, on remplace les coordonnées de termes par celles des concepts,
  3. « only » on n'utilise que des concepts.
- La correspondance entre termes et concepts étant ambigüe, ils donnent également des stratégies de désambigüisation :
  1. « all » on prend tous les concepts liés au terme,
  2. « first » si on a un ordre d'importance des concepts, on ne prend que le premier,
  3. « context » on désambiguïse par la méthode de la « densité de concepts » : on prend les sur- et sous-concepts et les termes qui les représentent et on regarde leurs fréquences dans le document.

Exemple d'application : le résumé automatique 1/3

- Première approche : par la *compréhension* ;
- inspirée des sciences cognitives et de l'IA ;
- on construit une représentation sémantique du texte, on la réduit et à partir de la réduction on génère un résumé ;
- pour la réduction on utilise des *marco-règles* :
  1. l'*élimination* : Pierre a vu une balle bleue ⟶ Pierre a vu une belle. La balle était bleue ⟶ Pierre a vu une balle,
  2. la *généralisation* : Pierre a vu un faucon ⟶ Pierre a vu un oiseau ; Pierre a vu un faucon. Pierre a vu un vautour ⟶ Pierre a vu des oiseaux,
  3. la *condensation* : Pierre a creusé le fondations, construit les murs, posé le toit... ⟶ Pierre a construit une maison ;
- problèmes : construire la représentation sémantique peut être couteux ; choisir ce qui est important lors de la phase de réduction peut être délicat ; on est au sommet de l'édifice, il ne faut pas se planter dans les couches inférieures.

Exemple d'application : le résumé automatique 2/3

- Deuxième approche : par *extraction* ;
- on trouve les phrases les plus importantes et on les extrait.
- Première idée : trouver les phrases dont les mots on le td-idf global le plus élevé (suppose l'existence d'un corpus). Facile à calculer, mais est-ce bien les phrases les plus intéressantes ?
- problèmes : pour des raisons stylistiques, les auteurs cultivent la synonymie ; les anaphores ne sont pas résolues ; il faut au moins une lemmatisation pour que les calculs soient cohérents ; quid de l'ordre des phrases ?
- Deuxième idée : se baser sur des phrases prototypiques : *dans cet article nous allons nous intéresser à ...* en affectant un score selon la présence ou non de certains mots indicateurs ;
- problèmes : demande une adaptation à chaque type de texte ; les anaphores ne sont pas résolues ; cohérence du résultat ?

Exemple d'application : le résumé automatique 2/3

- Troisième idée : par repérage de chaînes lexicales ;
- on considère, un par un, les noms. On cherche leurs distances dans WordNet et on cherche un graphe de relations entre eux qui corresponde au mieux à ces distances : il y a des fortes chances à ce que ce soient les relations qu'ils ont dans le texte ;
- ensuite, à l'aide de ces relations on attribue un score aux phrases et on prend celles de plus grand score.
- Quatrième idée : en analysant les relations entre les phrases ;
- après des analyses morphologique et syntaxique, on cherche certains patrons (par exemple, alors, par conséquent, etc.) qui permettent d'établir le graphe des relations entre phrases, et puis entre paragraphes ;
- ensuite, dans ce graphe, on trouve les nœuds (phrases) qui ont des « rôles sémantiques » précis, par exemple celui de conclusion.

Exemple d'application : le résumé automatique 2/3

- Cinquième idée : en construisant un « analyseur rhétorique »;
- [Marcu, 1997] a construit une base de 450 marqueurs discursifs qui permettent d'établir des relations discursives entre les propositions d'un texte ;
- il a développé un algorithme qui construit un arbre optimum dont les flèches correspondent à des relations rhétoriques (du type élaboration, justification, exemplification, concession, antithèse, contraste, évidence, etc.) ;
- la racine de l'arbre sera la proposition la plus saillante et on fait un parcours en largeur d'abord jusqu'à atteindre la taille de résumé souhaitée.

Exemple d'application : le résumé automatique 3/3

- Troisième approche : par *apprentissage* ;
- dans le cas de l'apprentissage supervisé, on prend un corpus, on en extrait des phrases (ou on leur attribue un score d'extraction) et on écrit des critères d'extraction. Ceux-ci peuvent être
  1. positionnels (la première phrase est plus importante, etc.),
  2. morphologiques et quantitafis (comme les fréquences de certains termes),
  3. discursifs ;
- à partir des critères d'extraction, on construit pour chaque phrase du corpus d'entraînement un vecteur de valeurs de critères ;
- l'algorithme compare ces vecteurs avec le score d'extraction donné manuellement, pondère les règles, et évalue les résultats sur un corpus de test ;
- en introduisant des nouvelles règles, on peut améliorer les performances du système ;
- on peut même imaginer des apprentissages non supervisés ou semi-supervisés.
- Problème : sans analyse morphosyntaxique ni résolution d'anaphores ; l'instance de l'apprentissage est la phrase, les relations entre phrases

Conclusions

- À travers quelques notions de base et quelques exemples, nous avons fait le tour des enjeux et méthodes de base de la fouille de texte ;
- mis à part la complexité du phénomène « langue », ce qui rend le sujet encore plus vaste est le fait qu'il n'y a pas de véritable frontière entre les différentes disciplines : linguistique, traitement automatique de langue, apprentissage artificiel, statistique, algorithmique, etc.
- ce qu'il faut retenir de ce cours : les différents angles d'attaque, les avantages et inconvénients des différentes méthodes et la tendance générale de les essayer toutes, en les comparant par des techniques d'évaluation ;
- plus spécifiquement, concernant la langue, les deux approches complémentaires : celle des méthodes formelles (logique, λ-calcul, etc.) et celle des méthodes statistiques, souvent avec apprentissage. Les meilleurs résultats s'obtiennent en les combinant.