Natural Language Processing (NLP)

Introduction au traitement automatique des langues

Georges-André Silber

Juin 2024

École des mines de Paris

Introduction

Quelques sources

- Chaire annuelle 2023—2024 de Benoît Sagot au Collège de France « *Apprendre les langues aux machines* »
- Cours du master MVA « *Algorithms for speech and language processing* »
- Cours de Stanford « *CS224N : Natural Language Processing with Deep Learning* »
- Livre « *Speech and Language Processing* » (Jurafsky/Martin)
- Livre/notes « *Natural Language Processing* » (Jacob Eisenstein)
- « *Neural Networks : Zero to Hero* », Andrej Karpathy
- Bibliographie commentée

- En français : Traitement Automatique des Langues (TAL)
- Lien avec l'IA : imiter et assister l'intelligence humaine
- Discipline pluri-disciplinaire : linguistes, informaticiens, mathématiciens
- « *Talistes, taliens, taleux* » (Lebarbé)
- Challenges principaux du NLP : analyse, génération, transformation de textes, interaction humain/machine
- Applications : linguistique, humanités, droit, santé, ...
- Années 90 : passage des règles à l'apprentissage automatique (ML)
- Les grands modèles de language (LLM) réalisent aujourd'hui la plupart des tâches du NLP de manière performante : état de l'art du domaine

Cours de B. Sagot, collège de France, novembre 2023

https://www.college-de-france.fr/fr/agenda/lecon-inaugurale/apprendre-les-langues-aux-machines-0

Introduction to speech and language processing

Benoît Sagot

Inria · Collège de France 1530 · PR[AI]RIE

MVA – Speech and Language processing – Class #1 – 1st March 2024

https://github.com/rbawden/MVA_2024_SL/blob/main/Course_%231/20240301-MVA2024-1.pdf

Histoire sélective du NLP

- 1950, traduction automatique (contexte de guerre froide)
- 1950, *Computing Machinery and Intelligence* (A. Turing)
- 1954, expérience Georgetown-IBM, traduction du russe vers l'anglais
- 1966, ELIZA (Joseph Weizenbaum)
- 1968, SHRDLU (PhD de Terry Winograd au MIT)
- 1970–2000, « ontologies conceptuelles »
- 2018, BERT (Google)
- 2020, GPT-3 (OpenAI)
- 2023, ChatGPT (OpenAI)

Histoire sélective des réseaux neuronaux

- 1943, Notion de neurone artificiel (McCulloch & Pitts)
- 1957/1958, Apprentissage supervisé, Perceptron (Rosenblatt)
- 1962, Plusieurs couches en propagation avant (Rosenblatt)
- 1986, Rétropropagation du gradient (Rumelhart, Hinton, Williams)
- 1989, Réseaux convolutifs (Le Cun *et al.*)
- 1990, Réseaux récurrents (Elman)
- 1997, LSTM (Hochreiter)
- 2006, *Deep Learning*, $c \geq 3$ (Hinton, Bengio)
- 2017, Architecture *Transformer* (Vaswani *et al.*)

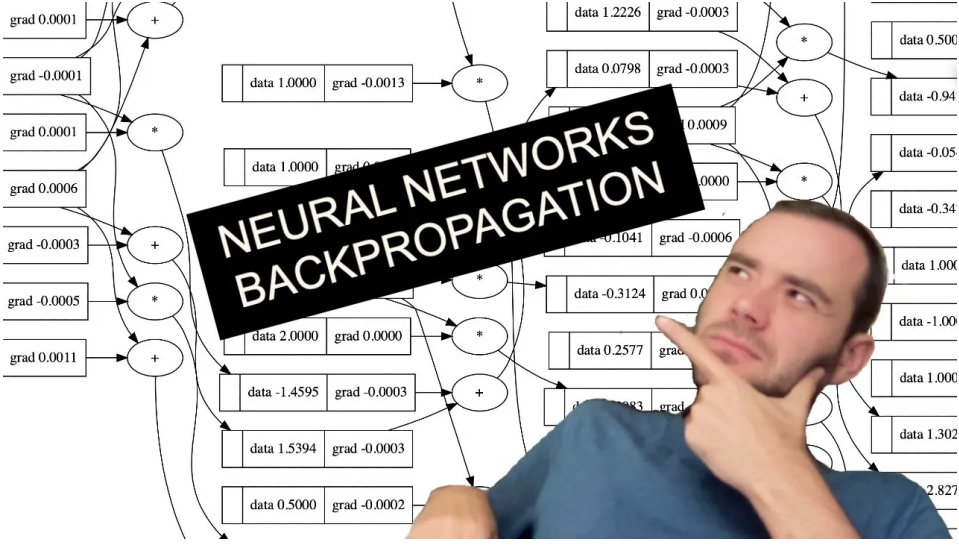
4e révolution de l'accès à l'information

Extrait de la leçon inaugurale de B. Sagot (11/2023) :

1. Écriture : stockage des informations de manière externe et pérenne. Outil d'accès à l'information
2. Imprimerie : externalisation et diffusion facilités
3. Web : numérisation massive, moteurs de recherche. Automatisation de l'identification des sources.
4. IA : restitution des informations et capacité externe de raisonnement

Rétropropagation du gradient

Andrej Karpathy : Backpropagation



https://www.youtube.com/watch?v=VMj-3S1tku0

Caractères et Alphabets

Données sources du NLP : caractères et alphabets

MINES PARIS | PSL

Pourquoi s'intéresser aux caractères ?

- Donnée de base du NLP : caractère ∈ alphabet
- Qualité des données primordiale pour le NLP
- En 2023, le Mojibake existe toujours
- Diversité des caractères dans les langues humaines
- Traitements plus compliqués quand on ne traite pas de l'anglais

Représentation des caractères

- Sur un ordinateur, un caractère est un entier positif.
- Cet entier positif représente un caractère dans une table donnée.
- Si on ne connaît pas la table, on ne peut pas connaître le caractère.
- 1 caractère : 5 bits, 7 bits, 1, 2, 3 ou 4 octets (Unicode).

Émile Baudot (1845–1903) : code Baudot

- Code sur 5 bits (1878)
- Téléscripteur, Telex
- Baud (Bd)
  - unité de modulation
  - nb de symboles par *s*
- *Stateful*

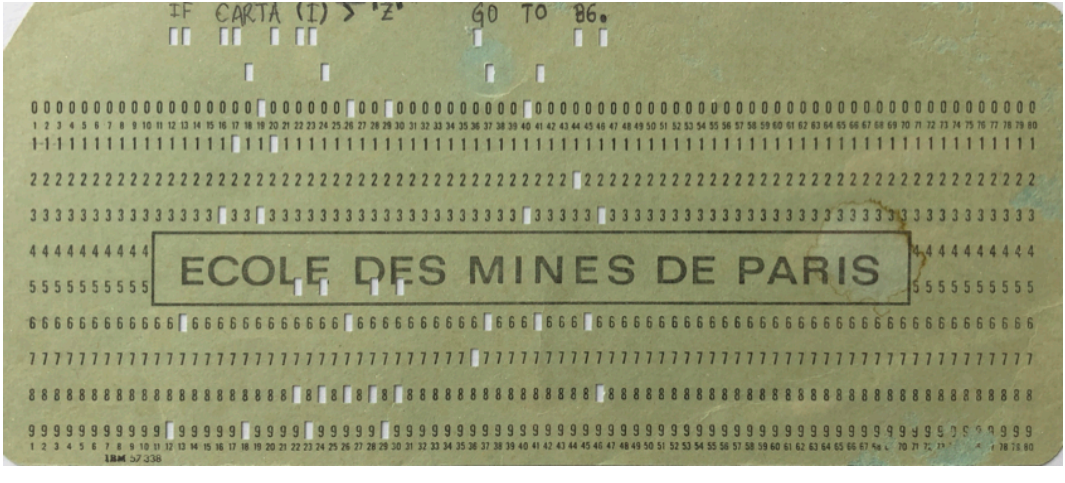
Code Baudot (US TTY)

00	01	02	03	04	05	06	07
NUL	E 3	LF	A –	SP	S ’	I 8	U 7
08	09	0A	0B	0C	0D	0E	0F
CR	D ENQ	R 4	J BEL	N ,	F !	C :	K (
10	11	12	13	14	15	16	17
T 5	Z +	L)	W 2	H £	Y 6	P 0	Q 1
18	19	1A	1B	1C	1D	1E	1F
O 9	B ?	G &	FIGS	M .	X /	V ;	LTRS

Letters	Figures	Control Chars.

Lab : Code Baudot

```
$ python encode_hello.py
***.**
* *.
    . *
*  .*
*  .*
** .
  *.
*  .**
[...]
```



Extended Binary Coded Decimal Interchange Code (8 bits)

Code EBCDIC

```
$ python fortran.py
I -> 0xc9
F -> 0xc6
  -> 0x40
C -> 0xc3
A -> 0xc1
R -> 0xd9
T -> 0xe3
A -> 0xc1
[...]
```

USASCII code chart

Bits b_7 b_6 b_5 →	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b_4 ↓ b_3 ↓ b_2 ↓ b_1 ↓ Row ↓ Column →	0	1	2	3	4	5	6	7
0 0 0 0	0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w
1 0 0 0	8	BS	CAN	(	8	H	X	h	x
1 0 0 1	9	HT	EM	)	9	I	Y	i	y
1 0 1 0	10	LF	SUB	*	:	J	Z	j	z
1 0 1 1	11	VT	ESC	+	;	K	[	k	{
1 1 0 0	12	FF	FS	,	<	L	\	l	|
1 1 0 1	13	CR	GS	-	=	M	]	m	}
1 1 1 0	14	SO	RS	.	>	N	^	n	~
1 1 1 1	15	SI	US	/	?	O	_	o	DEL

American Standard Code for Information Interchange (7 bits)

Code ASCII

```
#include <stdio.h>
int main(int ac, char *av[])
{
        printf("hell\x08o\n");
}
```

Caractères étendus : ISO 8859

- Ajouts aux 96 caractères affichables de l'US-ASCII
- Encodage sur 8 bits
- 128 nouveaux caractères (utilisation du bit restant d'un octet)
- 16 parties différentes (de 8859-1 à 8859-16)
- Pas utilisable pour certaines langues d'Asie (CJCV)

Code ISO-8859-1

```
(nlp) cervinia:latin1 gasilber$ cat latin1.py
s="Hé ôh"
with open("doc.txt", "w", encoding="iso-8859-1") as f:
    f.write(s)
(nlp) cervinia:latin1 gasilber$ python latin1.py
(nlp) cervinia:latin1 gasilber$ hexdump -C doc.txt
00000000  48 e9 20 f4 68                                    |H. .
00000005
(nlp) cervinia:latin1 gasilber$ cat doc.txt
H? ?h
```

Caractères pour le monde entier : Unicode

- Développé depuis 1991 (première version).
- Standard Unicode 13 (mars 2020) : 144 697 caractères.
- Relié à la norme ISO/CEI 10646 (un sous-ensemble du standard Unicode).
- Un caractère ISO/CEI 10646 : couple nom unique / numéro unique (point de code).
- 245 000 points de codes assignés dans un espace pouvant contenir 1 114 112 codes différents (21 bits).
- 17 zones de 65 536 points de code : plans de code.
- Chaque plan de code est divisé en 4096 colonnes de code de 16 points de code.
- Codes : de `0` à `0x10FFFF` (1 114 112 - 1).
- De `0x0` à `0xFF` : ISO/CEI 8859-1.

Unicode

```
(nlp) cervinia:unicode gasilber$ python wat.py
ê == ê -> False
(nlp) cervinia:unicode gasilber$ python unicode.py
--> Aéêê 5
0 0041 Lu LATIN CAPITAL LETTER A
1 00e9 Ll LATIN SMALL LETTER E WITH ACUTE
2 00ea Ll LATIN SMALL LETTER E WITH CIRCUMFLEX
3 0065 Ll LATIN SMALL LETTER E
4 0302 Mn COMBINING CIRCUMFLEX ACCENT
(nlp) cervinia:unicode gasilber$ python spaces.py
0x2000 '\u2000' EN QUAD Zs ' '
0x2001 '\u2001' EM QUAD Zs ' '
0x2002 '\u2002' EN SPACE Zs ' '
0x2003 '\u2003' EM SPACE Zs ' '
```

UTF-8

- Universal Character Set Transformation Format sur 8 bits.
- Encodage de ISO/CEI 10646 sur 8 bits, compatible avec l'US-ASCII (`0x0` à `0x7F`).
- Stockage d'un point de code sur 1 à 4 octets consécutifs.
- Le décodage des *strings* devient *stateful*.
- Souvent compatible avec le code existant, insensible à l'*endianness*.

Définition du nombre d'octets utilisés dans le codage (attention ce tableau de principe contient des séquences invalides)

Caractères codés	Représentation binaire UTF-8	Premier octet valide (hexadécimal)	Signification
U+0000 à U+007F	0bbb·bbbb	00 à 7F	1 octet, codant jusqu'à 7 bits
U+0080 à U+07FF	110b·bbbb 10bb·bbbb	C2 à DF	2 octets, codant jusqu'à 11 bits
U+0800 à U+FFFF	1110·bbbb 10bb·bbbb 10bb·bbbb	E0 à EF	3 octets, codant jusqu'à 16 bits
U+10000 à U+10FFFF	1111·0bbb 10bb·bbbb 10bb·bbbb 10bb·bbbb	F0 à F3	4 octets, codant jusqu'à 21 bits
	1111·0100 1000·bbbb 10bb·bbbb 10bb·bbbb	F4	

```
(nlp) cervinia:utf8 gasilber$ python count.py
...
0 feff Cf ZERO WIDTH NO-BREAK SPACE
1 0048 Lu LATIN CAPITAL LETTER H
2 00e9 Ll LATIN SMALL LETTER E WITH ACUTE
(nlp) cervinia:utf8 gasilber$ wc -c Classeur1.csv
      XX Classeur1.csv
```

UTF-16 et UTF-32

- UTF-16 : codage sur 2 ou 4 octets (16 ou 32 bits).
- UTF-32 : codage fixe sur 4 octets (32 bits).
- Codages sensibles à l'*endianness*.
- Utilisation d'un BOM, `0xFFFE` (*little endian*) ou 0xFEFF (*big endian*).
- Dans certains fichiers UTF-8 (par ex. CSV généré par Excel) utilisation d'un BOM pour indiquer que le fichier est UTF-8.
- Il existe également un format UTF-5 (unicode sur téléscripteur).
- Bush hid the facts

Formats de fichiers

- Format texte
- Comma-Separated Values (CSV)
- JavaScript Object Notation (JSON)
- YAML Ain't Markup Language
- TOML Tom Obvious, Minimal Language
- eXtensible Markup Language (XML)
- Portable Document Format (PDF)
  - Langage de description de page (Turing complet)
  - $b^2 - 4ac \rightarrow$ `b b mul 4 a mul c mul sub`

Mots

Qu'est-ce qu'un mot ?

- Notion mal définie
- Problème de la séparation des mots d'une phrase
- Espaces, ponctuations ?
- Lemmatisation, racinisation pour commencer à "classifier"

Lemmatisation (lemmatization)

- Réduction des mots à leur forme canonique (le lemme)
- « *avoir* » depuis « *eussions eu* »
- « *des avions* » vs « *nous avions* »

continu	continu
continua	continuer
continuait	continuer
continuant	continuer
continuation	continuation
continuations	continuation
continue	continu | continuer

Racinisation (stemming)

- Regroupement des mots par racine commune
- ”Lemmatisation” simplifiée

continu	continu
continua	continu
continuait	continu
continuant	continu
continuation	continu
continuations	continu
continue	continu

Délimitation d'une phrase

Séparation des phrases d'un texte. À l'écrit, la ponctuation ou la casse permet en général de séparer les phrases, mais des complications peuvent être causées par les abréviations utilisant un point, ou les citations comportant des ponctuations à l'intérieur d'une phrase, etc.

Séparation des mots

Dans la langue parlée, les phrases ne sont qu'une chaîne de phonèmes, où l'espace typographique n'est pas prononcé. Par exemple, « *un bon appartement chaud* » et « *un Bonaparte manchot* » sont identiques d'un point de vue phonétique.

Outil de base : les expressions régulières

Application : expressions régulières

- Expressions régulières par génération d'un automate fini (Ken Thompson).
- `grep`, `lex`, analyseur lexical
- `https://regexcrossword.com`
- Python : `import re`
- hyperscan

Exemple 1 : utilisation dans un IDE

```
#define MAX_URI_COUNTRY 3
#define MAX_URI_CORPUS 5
#define MAX_URI_NATURE 70
#define MAX_URI_YEAR 5
#define MAX_URI_MONTH 3
#define MAX_URI_DAY 3
#define MAX_URI_NUMBER 30
#define MAX_URI_VERSION 9
#define MAX_URI 256
```

```
MAX_(\w+)
$1_MAX
```

Exemple 2 : découpage d'un arrêt de cour d'appel d'Agen

```
intro_re = re.compile(
  r'^(?P<intro>.*?)(?=('
  r'<p>\s*A\s+rendu\s+l.arrêt\s+((réputé\s+)?
  r'contradictoire|par\s+défaut)'
  r'|<p>\s*EXPOS(É|E)\s*DU\s*LITIGE'
  r'|<p>A rendu réputé l.arrêt réputé contradictoire'
  r'))',
  re.UNICODE|re.DOTALL|re.MULTILINE|re.IGNORECASE)
decision_re = re.compile(
  r'(?P<decision>(<p>par\s*ces\s*motifs).*)$',
  re.U|re.DOTALL|re.MULTILINE|re.IGNORECASE)
```

Exemple 3 : numéros d'alinéas (Legistix)

```
alinea_number = (
  r"("
  r"\w\)(?=\s+)"
  r"|\d{1,2}°(\s+bis)?(?=\.?\s+)"
  r"|\d{1,2}(\s+bis)?(?=\.?\s+)"
  r"|[IVX]+(?=\.\s+)"
  r")"
)
```

Extraction de texte structuré depuis un PDF

- Github, Gitlab
- Étape 0 : installer Docker, faire fonctionner le Dockerfile
- Étape 1 : extraire le texte du PDF (poppler), récupérer des données
- Étape 2 : compléter le script Python `text2md` (regexps)
- Étape 3 : compléter le script Python `md2xml` (regexps)
- À rendre avant dimanche 7 janvier à 23h59
- Livrable : pull request (Github), merge request (Gitlab) ou patch (`git diff`)
- Voir les instructions

Langages formels et NLP

Informatique et langages

- Informatique = science du traitement automatique des données.
- Données = suite d'informations représentées dans un certain langage.
- Théorie des langages formels : étude des structures internes formelles des langages (niveau syntaxique).
- Auparavant, les systèmes d'analyse syntaxique utilisaient principalement des grammaires formelles, aidées éventuellement par des statistiques.
- Aujourd'hui, dans ces systèmes, les grammaires formelles sont complétées, voire remplacées, par des techniques d'apprentissage automatique.

Hiérarchie des grammaires formelles de N. Chomsky et M. P. Schützenberger :

- les grammaires de type 3 génèrent la famille des *langages rationnels* ou *langages réguliers*. Langages reconnaissables par les *automates finis* ;
- les grammaires de type 2 génèrent la famille des *langages algébriques*. Langages reconnaissables par les *automates à pile* ;
- les grammaires de type 1 génèrent la famille des *langages contextuels*. Langages reconnaissables par les *automates linéairement bornés* ;
- les grammaires de type 0, dites grammaires générales, génèrent la famille des *langages récursivement énumérables*. Langages reconnaissables par une *machine de Turing*.

Langages rationnels

- A : ensemble fini de symboles (*alphabet*) ;
- A^* : ensemble des mots que l'on peut former avec A ;
- Partie de A^* : un *langage* ;
- Opérations rationnelles, avec X et Y deux parties de A^* :
  - Concaténation : XY

$$\{ab, c\}\{ba, c\} = \{abba, abc, cba, cc\}$$

  - Union : $X \cup Y$

$$\{ab, c\} \cup \{ba, c\} = \{ab, ba, c\}$$

  - Étoile de Kleene (notée X^*) : le plus petit langage contenant ϵ, X et qui est clos pour l'opération de concaténation.

$$\{a, ab\}^* = \{\epsilon, a, aa, ab, aaa, ababaa, ...\}$$

Application pratique : **expressions régulières** par génération d'un automate fini (Ken Thompson). `lex`, analyseur lexical.

Grammaires algébriques

- Grammaires non-contextuelles / hors-contexte
- Règles de la forme $X \to \alpha$ où α est un terminal ou un non terminal
- Un langage est algébrique si il ∃ une grammaire algébrique le décrivant
- La plupart des langages de programmation ont une grammaire algébrique
- Peuvent être décrits sous la forme Backus-Naur (BNF)[1]
- Exemples de langages algébriques (et non rationnels) :

$$S \to aSb \mid \epsilon$$

$$\{a^n b^n \mid n \geq 0\}$$

$$S \to x \mid y \mid z \mid S + S \mid S - S \mid S * S \mid S/S \mid (S)$$

- Outils : YACC (LALR), ANTLR (LL)

1. https://fr.wikipedia.org/wiki/Forme_de_Backus-Naur

Grammaires algébriques (2)

- Panini, un grammairien de l'Inde antique a formalisé une grammaire du Sanskrit (IVe siècle avant J.C.), avec près de 4000 dans une notation proche de la BNF.
- Langages de Dyck : ensemble des mots bien parenthésés sur un ensemble fini de parenthèses ouvrantes et fermantes.

Définition (Morphisme alphabétique)
$h : A^* \to B^*$ avec A et B des monoïdes libres et h un morphisme. h est *alphabétique* si l'image d'une lettre de A est une lettre de B ou ϵ.

Théorème (Chomsky–Schützenberger)
Un langage L est algébrique ssi il existe un langage de Dyck D, un langage rationnel K et un morphisme alphabétique h tels que

$$L = h(D \cap K)$$

Grammaires algébriques (3)

- Soit la grammaire :

$$S \rightarrow S + S \mid a \mid 1$$

- Dérivations de $1 + a + a$

Dérivation gauche	Dérivation droite
S	S
$S + S$	$S + S$
$1 + S$	$S + a$
$1 + S + S$	$S + S + a$
$1 + a + S$	$S + a + a$
$1 + a + a$	$1 + a + a$

⇔ Machine de Turing non déterministe linéairement bornée (avec n la taille de l'entrée, ruban de taille kn où k est une constante indépendante de n).

$$L = \{a^n b^n c^n \mid n \geq 1\}$$

La machine de Turing

Objet mathématique abstrait composé :

- d'une bande infinie découpé en cases pouvant contenir un symbole ;
- d'une tête de lecture pouvant à chaque étape lire un symbole, écrire un symbole, puis se déplacer sur la bande d'une case à gauche ou à droite ;
- un registre fini d'états dans lesquels peut se trouver la machine ;
- une table d'action indiquant pour un état et un symbole l'action à effectuer.

La machine de Turing

Une machine de Turing déterministe est un septuplet $M = \langle Q, \Gamma, b, \Sigma, \delta, q_0, F \rangle$ où :

- Q est l'ensemble fini non vide des *états* ;
- Γ est l'ensemble fini non vide des *symboles de la bande* ;
- $b \in \Gamma$ est le symbole *blanc* ;
- $\Sigma \subseteq \Gamma \setminus \{b\}$ est l'ensemble des *symboles d'entrée*, les seuls symboles autorisés initialement sur la bande ;
- $\delta : (Q \setminus F) \times \Gamma \nrightarrow Q \times \Gamma \times \{\leftarrow, \rightarrow\}$ est la fonction partielle de *transition*. Si δ n'est pas définie sur l'état courant et le symbole courant, la machine s'arrête ;
- $q_0 \in Q$ est l'état initial ;
- $F \subseteq Q$ est l'ensemble des *états acceptants* : le contenu initial de la bande est *accepté* par M si elle s'arrête dans un état de F.

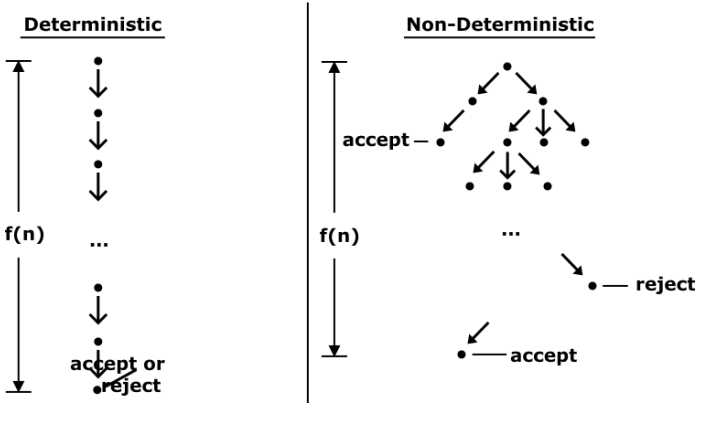
Exemple de partie de δ : $\delta(q_1, x) = (q_2, y, \leftarrow)$ indique que dans l'état q_1 quand x est lu sur la bande, on passe en état q_2, on écrit y et on se déplace à $\leftarrow$.

La machine de Turing non déterministe

Une machine de Turing non déterministe est un septuplet $M = \langle Q, \Gamma, b, \Sigma, \delta, q_0, F \rangle$ où :

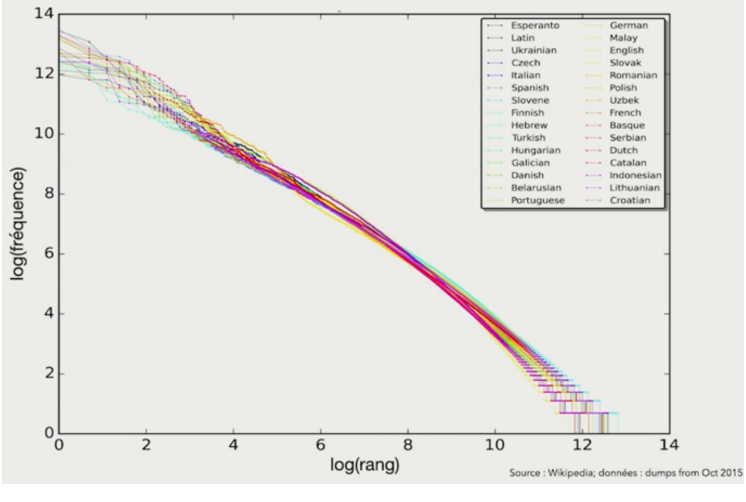
- Q est l'ensemble fini non vide des *états* ;
- Γ est l'ensemble fini non vide des *symboles de la bande* ;
- $b \in \Gamma$ est le symbole *blanc* ;
- $\Sigma \subseteq \Gamma \setminus \{b\}$ est l'ensemble des *symboles d'entrée*, les seuls symboles autorisés initialement sur la bande ;
- $\delta \subseteq (Q \setminus F \times \Gamma) \times (Q \times \Gamma \times \{\leftarrow, \rightarrow\})$ est la relation de *transition* ;
- $q_0 \in Q$ est l'état initial ;
- $F \subseteq Q$ est l'ensemble des *états acceptants* : le contenu initial de la bande est *accepté* par M si *une branche* s'arrête dans un état de F.

Machine de Turing déterministe vs non-déterministe



Représentation des tokens

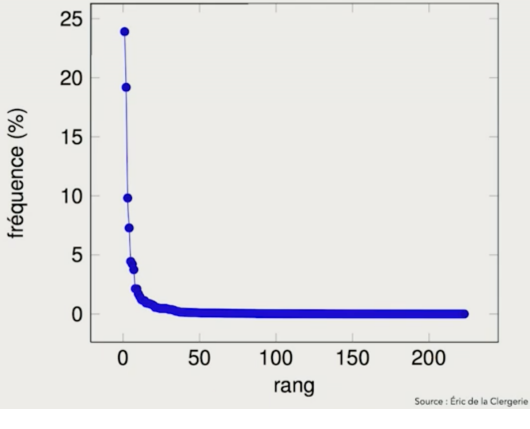
Loi de Zipf sur les mots d'un corpus



Rang / fréquence pour les 10 premiers millions de mots de 30 Wikipedia.

Source : cours de B. Sagot 2023

Loi de Zipf sur les phrases d'un corpus



Fréquence de constructions syntaxiques dans un corpus de 10 000 phrases analysées automatiquement.

Source : cours de B. Sagot 2023

Représenter les propriétés des tokens par des vecteurs

- Chaque token, dans le contexte où il apparaît, porte des propriétés (morpholoqiques, syntaxiques, sémantiques)
- On représente généralement les tokens par des vecteurs :
  - représentation utilisée depuis longtemps par les réseaux de neurones
  - permet également de tenter d'encoder la sémantique dans un espace vectoriel

Représentation *One hot*

1	a
2	à
3	abaca
4	abacas
5	abacule
6	abacules
7	abaissa
8	abaissable
9	abaissables
10	abaissai
11	abaissaient
12	abaissais
...	
29 000	zythum
30 000	zythums

	1	2	3	4	5	6	7	8	...	29 000	30 000
abaissa	0	0	0	0	0	0	1	0	...	0	0

- variables catégorielles
- taille du vecteur = nombre de tokens dans le modèle
- chaque token est représenté par un vecteur de 0 où une seule composante est à 1
- pas de notion de proximité

Source : cours de B. Sagot 2023

Hypothèse distributionnelle

- Le contexte donne des informations sur un token.
- Exemple [tiré de Nida 75, Lin 78, Sagot 23] :

  Il y a une bouteille de *tesgüino* sur la table.
  Tout le monde aime le *tesgüino*.
  Le *tesgüino* rend ivre.
  On produit le *tesgüino* à partir de maïs.
- Hypothèse : deux tokens sont similaires s'ils apparaissent dans un même contexte
- Firth (1957) : *you shall know a word by the company it keeps*.

Embeddings non contextuels

- *Word embedding* : plongement lexical
- Représentation vectorielle des tokens
- Étant donné un mot on lui assigne une représentation vectorielle unique sur la base de toutes ses apparitions dans un grand corpus
- Approches par comptage ou statistiques
- Approches prédictives par modèle neuronal
- Voir Embedding projector

Approche par comptage

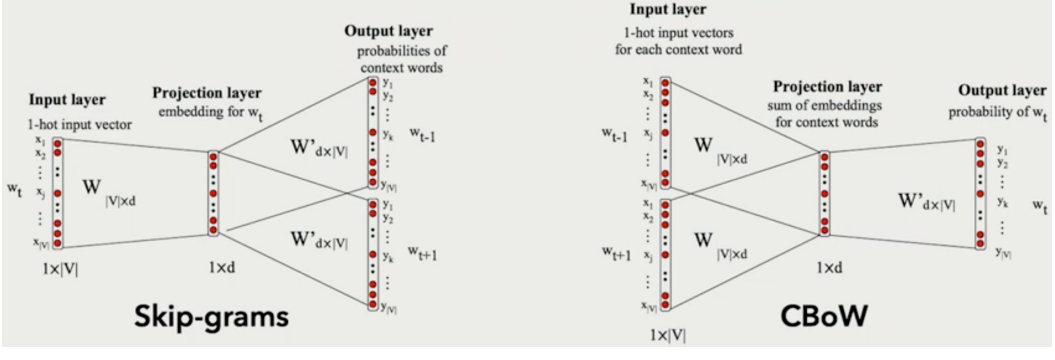
	autruche	ordinateur	donnée	salade	résultat	sucre	...	...
abricot	0	0	0	1	0	1		
ananas	0	0	0	1	0	1		
numérique	0	2	1	0	1	0		
information	0	1	6	0	4	0		

	As you like it	Twelfth night	Julius Caesar	Henry V	...	...
battle	1	0	7	13		
good	114	80	62	89		
fool	36	58	1	4		
wit	20	15	2	3		

- Matrices de co-occurrence mot/mot et mot/document.
- Vecteurs ”creux” et de grande dimension
- Mots rares et fréquents pèsent autant (1 dimension)
- Solutions : TF-IDF, Latent Semantic Analysis, Latent Dirichlet Allocation

Source : cours de B. Sagot 2023, adapté du cours de Jurafsky et Martin

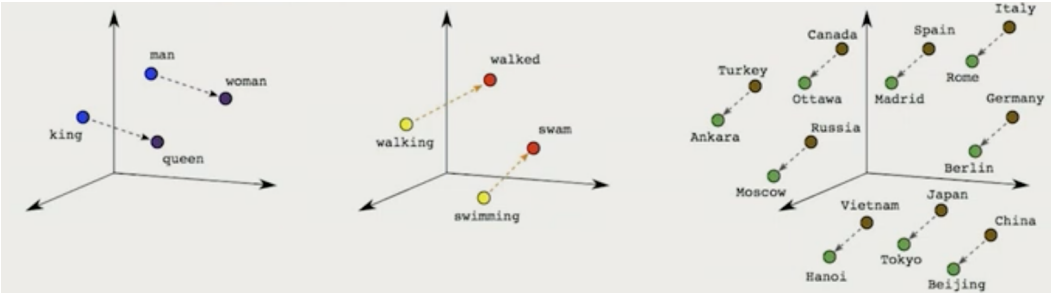
Approche prédictive neuronale



- Word2vec, fastText, GloVe
- Vecteurs One-hot
- La couche cachée est lue comme un word embedding

Source : cours de B. Sagot 2023

Structuration de l'espace



- Succès de word2vec : structuration de l'espace
- Calculs par analogie : Paris - France + Italy ≡ Rome

Source : cours de B. Sagot 2023, d'après Irina Sigler

Limite des embeddings non contextuels

Portrait de Frédéric Thomas (avocat, littérateur, journaliste). Musée Carnavalet

Pierre est un excellent avocat

Source : wiktionary

Pierre mange un excellent avocat

- ”avocat” est représenté par le même vecteur quel que soit son contexte
- Les modèles de langues et leurs *embeddings contextuels* permettent de lever cette limitation

Source : cours de B. Sagot 2023

Retour sur TF-IDF : moteurs de recherche

- Problème : pour une recherche q, dans quel ordre renvoyer les documents ?
- TF-IDF

$$\text{tfidf}(t, d, D) = \text{tf}(t, d) \cdot \log \frac{N}{1 - |\{d \in D : t \in d\}|}$$

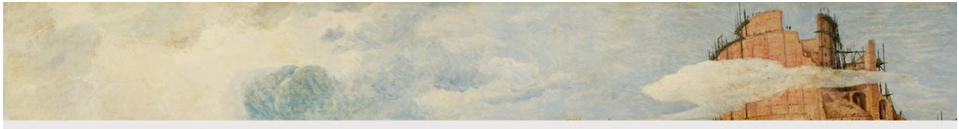
- Okapi BM25 (sacs de mots)

$$\text{score}(D, Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}$$

$$\text{IDF}(q_i) = \ln\left(\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} + 1\right)$$

- Démonstration d'indexation du JORF avec Solr
- Relevant Search

Modèles de langues



Apprendre les langues aux machines

Modèles de langue

Benoît Sagot

COLLÈGE DE FRANCE 1530

Inria

PR[AI]RIE

Cours 3/8 – chaire « Informatique et sciences numériques » – Collège de France – 22 décembre 2023

https://www.college-de-france.fr/fr/agenda/cours/apprendre-les-langues-aux-machines/modeles-de-langue

Intro to LLMs

Andrej Karpathy
Nov 2023

https://www.youtube.com/watch?v=zjkBMFhNj_g

Collège de France

NEW YORK UNIVERSITY Meta AI

L'IA Pilotée par Objectifs
(Objective-Driven AI)

Vers des machines capables d'apprendre, de mémoriser, de raisonner, et de planifier, qui sont fiables et contrôlables.

Yann LeCun
New York University
Meta – Fundamental AI Research

Cours de Benoît Sagot
Collège de France
2024-02-09

https://www.youtube.com/watch?v=7ilel2dZ_j8