

Fouille de données lexicales

Extraction 1

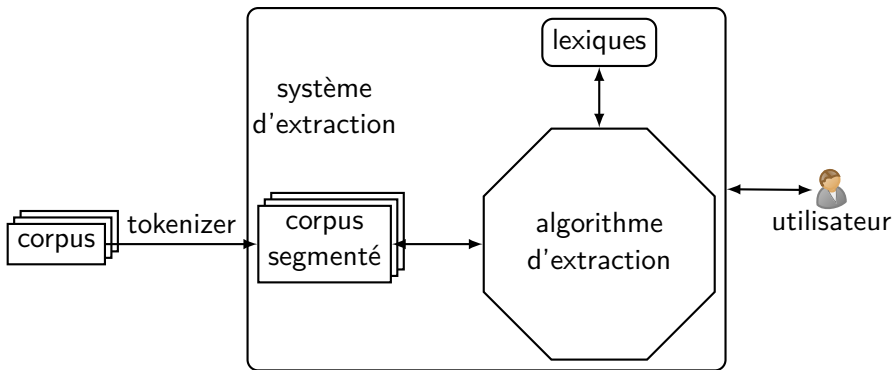
Damien Nouvel

Inalco

Contexte de la fouille de données lexicales

Extraction à partir du corpus

- **segmentation** des textes en tokens ou **tokenisation**
- construction de **représentations** du corpus
- filtrage des éléments pertinents
- approche avec une **compréhension limitée**



Segmentation en tokens

Formalisation de la **tokenisation**

Soit une chaîne S comme séquence (suite) de caractères C_i

$$S = (c_1, c_2, c_3, \dots c_n)$$

On cherche des paires d'indices $((i_1, j_1), (i_2, j_2), \dots (i_k, j_k))$

Tels que :

- il y a beaucoup moins de tokens que de caractères $k \ll n$
- un token t_l est la chaîne de caractère $(c_{i_l}, c_{i_l + 1}, \dots c_{j_l})$
- pas d'intersection entre tokens, $i_k \geq j_{k-1}$
- la concaténation des tokens « couvre » au maximum S

Exemple avec « Le chat boit du lait »

- la séquence comporte 20 caractères (espaces compris)
- on identifie les tokens $((0, 1), (3, 6), (8, 11), \dots)$
- le deuxième token $(3, 6)$ est (c_4, c_5, c_6, c_7) soit « chat »
- la chaîne est reconstruite avec les tokens (modulo les espaces)

Importance et difficultés de la tokenisation

La segmentation détermine tous les autres traitements TAL

- les segments seront les **unités minimales** de traitement
- les **représentations** sont construites à partir des segments

Deux exigences contradictoires

- associer un **sens** à chaque segment
- **limiter** raisonnablement la taille du vocabulaire

Problèmes linguistiques de la tokenisation

- **morphologie** et **agglutination**
 - possibilité de calcul par dérivation
 - exemple : « anticapitalisme »
- **unités polylexicales**
 - impossible de construire le sens à partir de sous-segments
 - exemple : « cordon bleu »

Recherche de la taille **optimale** de segments

Segmentation par repérage de séparateurs

Recherche des séparateurs entre tokens

- focalisée sur la recherche des éléments entre les tokens
- modélisation à partir d'**expressions régulières**
- avec `grep` utilisation de `\b`

Avantages

- **simple** à implémenter
- très **rapide** (automates)
- relativement **robuste**

Inconvénients

- inopérante pour les **langues sans séparateurs** (chinois)
- pas d'**analyse** des tokens
- erreurs pour les tokens avec **ponctuations** (sigles, etc.)

Segmentation par modélisation linguistique

Approches **prescriptives** de la segmentation

- exploitation d'un **lexique** de la langue
- **paradigmes** de flexions et de conjugaisons
- utilisation d'**expressions régulières**
- avec `grep`, utilisation de `\w`

Avantages

- segmentation motivée **linguistiquement**
- **informations** sur les mots (catégorie, racine, famille, etc.)

Inconvénients

- mise en œuvre difficile pour la segmentation en **morphèmes**
- **ressources** à constituer pour chaque langue
- limitées au **vocabulaire** du lexique
- spécifier ou détecter la langue est nécessaire

Segmentation statistique en sous-mots

Algorithmes BPE (Byte Pair Encoding, Gage 1994) et WordPiece

- exploitation d'un **corpus volumineux**
- initialisation, chaque caractère est un token
- jusqu'à une **taille de vocabulaire** souhaitée
 - recherche dans le corpus des « **paires** » les plus
 - fréquentes (BPE)
 - probables (WordPiece)
 - **fusion** et **remplacement** des paires trouvées

Avantages

- capacité à traiter des **tokens hors-vocabulaire**
- modélisation (empirique) de la **morphologie**

Inconvénients

- pas de justification **linguistique**
- pas d'**informations linguistiques** explicites

Utilisés par les LLMs (BERT: WordPiece 30K, GPT4 : BPE 200K)

Comparaison des tokenizers

Outil	Tokens
NLTK	Charles de Gaulle n' était pas anticapitaliste
Spacy	Charles de Gaulle n' était pas anticapitaliste
Stanza	Charles de Gaulle n' était pas anticapitaliste
BERT	Charles de Gaulle n' était pas antica pital iste
GPT	Charles de Gaulle n' était pas ant icap ital iste

Quelques critiques

- traitements très différents des **ponctuations**
- aucune **expression polylexicale**
- pas d'analyse **morphologique** bien construite

Fouille textuelle à base de motifs

Définition d'un **motif** (en anglais *pattern*)

- recherche de sous-ensemble des données selon des critères
 - élément statique individuel
 - ensemble d'éléments
 - séquences d'éléments (ou d'ensembles d'éléments)
- **fréquence** par décompte de son nombre d'occurrence
- pour le TAL, motifs sur des **séquences** ou des **graphes**
- capacité à **généraliser**
 - expressions régulières
 - appel au lexique
 - caractéristiques linguistiques

Application d'un motif

- recherche dans les données des **occurrences**
- algorithmes rapides à base d'automates
- récupération du **contexte** ou des **concordances**

Recherche des motifs séquentiels

Formellement, recherche de **sous-séquences** de tokens

- soient
 - un **motif séquentiel** $M = (m_1, m_2, m_3 \dots m_n)$
 - une séquence de **tokens** $T = (t_1, t_2, t_3 \dots t_m)$
 - $t_i \in m_j$ si le token t_i est reconnu par l'élément de motif m_j
- la paire d'indices (k, l) est une **occurrence** de M dans T si $\forall i \in \{0 \dots l - k\}, t_{k+i} \in m_i$
- on peut associer à l'occurrence (k, l) les contextes de taille c
 - gauche $(t_{k-c} \dots t_{k-1})$
 - droit $(t_l \dots t_{l+c})$

On peut rechercher à l'aide de motifs

- les **séquences** qui ont au moins une occurrence
- les **occurrences**
- les **occurrences** et leurs **contextes** (gauches et droits)

Approches de recherches par motifs

Différentes méthodes pour rechercher des motifs

- utilisation d'**expressions régulières**
 - la donnée source n'est pas enrichie
 - l'expression régulière est convertie en **automate**
- exploitation de **lexiques** de la langue
 - enrichissement de la recherche (requête ou données) par le lexique
 - les motifs s'appuient sur des **informations** du lexique
 - génération de formes par **paradigmes** (flexion, conjugaison)
 - pas désambiguation automatique (en général, sauf Unitex)
- inférences par **prétraitement** des données
 - la donnée source est enrichie par des **algorithmes**
 - une **désambiguation** (souvent **contextuelle**) est réalisée
 - recherche sur les informations calculées

Approches assez **contraintes**

- recherche **exacte** (*string matching*)
- les lexiques sont **finis**

Approximations des recherches par motifs

Approximations déterministes

- insensibilité à la **casse** (encodages)
- insensibilité aux **diacritiques** (encodages)
- équivalences entre **encodes de caractères**
- extension d'éléments lexicaux aux **synonymes**
- éléments **optionnels**
- **répétition** (quantifiée) d'éléments
- tokens libres **intercalés** dans le motif

Quelques exemples d'approximations non-déterministes

- degrés de **synonymie** (similarités)
- variations d'**orthographe** ou de **typographie**
 - variations linguistiques
 - variations de forme
- **probabilités** sur les informations linguistiques

Concordances

Exploration de corpus par recherche des **concordances**

Concordances

- **occurrences** du motif
- affichage du **contexte** (gauche et droit)
- **lien** vers la donnée source pour la consulter

Importance du **retour au texte**

- vérification des **occurrences**
- amélioration des **motifs**
- difficultés pour la **complétude** des résultats

Outil basé sur les **automates**

- utilisation d'automates
 - application des **lexiques** disponibles (par langue)
 - phrases du corpus comme graphes (sans cycle)
 - requête utilisateur comme graphe
- **concordances** par méthode d'appariement de graphes

Construction des requêtes par interface graphique

- **nœuds** : conditions sur les tokens à rechercher
- **arcs** : syntagmes et expressions linguistiques

Recherches par automates **optimisés** (compilés)

Langage CQL

Spécification de motifs par langage standardisé

- standard *Corpus Query Language* (processeur CQP)
- recherches basées sur
 - **paradigmes** : tokens, expressions régulières, linguistique (POS)
 - **syntagmes** : liste de paradigmes avec possibilité de variables liées

Principes généraux

- recherche d'un token avec les **doubles guillemets**
- utilisation d'**expressions régulières**
- possibilité de neutraliser la **casse** et les **diacritiques**
- recherche des attributs avec les **crochets**
- opérateurs **logiques** (et &, ou |, négation!)
- **répétitions** de tokens (potentiellement vides)
- utilisation de données de la **structure** des documents

Exemple

```
("un"|"des") [ ]{0,3} [lemma="homme" & tag="N.*"]
```

Outils implémentant CQL

SketchEngine

- Outil en ligne <https://app.sketchengine.eu/#open>
- Documentation
<https://www.sketchengine.eu/documentation/corpus-querying>

TXM

- Outil en ligne <http://portal.textometrie.org/demo>
- Documentation
<https://txm.gitpages.huma-num.fr/txm-manual/analyser-un-corpus.html>
- Memo
https://txm.gitpages.huma-num.fr/textometrie/files/documentation/memo_cql.pdf

Recherche de motifs de graphes

Description formelle des graphes

- ensembles de nœuds (*vertices*) N
- ensembles d'arêtes (*edges*) $A \subset N \times N \times L$ (avec L les étiquettes)

Pour les graphes de **dépendances**

- les nœuds sont les tokens
- les arêtes sont les relations syntaxiques **orientées**
- recherche de **sous-graphe**
- exploitation d'informations morphologiques ou syntaxiques

Avantages

- notion de séquence contigüe remplacée par les **relations**
- informations linguistiques et sémantiques

Inconvénients

- nécessite un analyseur syntaxique (*parser*) **robuste**
- plus grande **complexité** de recherche

Représentation lexicale des documents et du corpus

Représentation par **occurrences de termes**

- pour un **document**
 - un document D_i comme séquence de tokens
 $D = (t_1, t_2, t_3 \dots t_m)$
 - vocabulaire du document
 $V_D = \{m | \exists k, t_k = m\}$
 - fréquence d'un terme dans le document (nombre d'occurrences)
 $F_{m,D} = |\{k, t_k = m\}|$
- pour un **corpus** $C = \{D_1, D_2 \dots D_n\}$
 - vocabulaire du corpus par union sur les documents
 $V_C = \cup_{D \in C} V_D$
 - fréquence d'un terme dans le corpus par somme sur les documents
 $F_{m,C} = \sum_{D \in C} F_{m,D}$

Quelques remarques

- utilisation de **termes** (généralement les tokens)
- pas d'encodage sur la **position des termes**
- le **vocabulaire** du modèle dépend du corpus

Matrice terme par documents

Représentation lexicale sous forme de **matrice**

- chaque **document** comme (vecteurs) lignes
- apparitions des **termes** (tokens) comme (vecteurs) colonnes

Exemple avec trois phrases

- d1 : « un beau chat »
- d2 : « un chat et un gros chat »
- d3 : « un beau chien et un beau chat et un gros lion »

	un	beau	chat	et	gros	chien	lion
d1	1	1	1	0	0	0	0
d2	2	0	2	1	1	0	0
d3	3	2	1	2	1	1	1

- vocabulaire du corpus : (*un, beau, chat, et, gros, chien, lion*)
- document *d2* représenté par le vecteur : (2, 0, 2, 1, 1, 0, 0)

Mesures de distance ou de similarité

Mesures de **distance** ou de **similarité** entre deux vecteurs

- $X = (x_1, x_2, x_3 \dots x_n)$
- $Y = (y_1, y_2, y_3 \dots y_n)$

Nom	Domaine	Formule
Similarité de Jaccard	$[0, 1]$	$\frac{ X \cap Y }{ X \cup Y } = \frac{ \{i x_i > 0\} \cap \{i y_i > 0\} }{ \{i x_i > 0\} \cup \{i y_i > 0\} }$
Distance de Manhattan	$[0, +\infty]$	$\sum_i x_i - y_i $
Distance Euclidienne	$[0, +\infty]$	$\sqrt{\sum_i (x_i - y_i)^2}$
Similarité cosinus	$[-1, 1]$	$\frac{X \cdot Y}{ X * Y } = \frac{\sum_i x_i * y_i}{\sqrt{\sum_i x_i^2} * \sqrt{\sum_i y_i^2}}$

Comparaison des **documents** avec une **requête** (ou entre eux)

Transformation TF-IDF

Représentation des documents par la matrice

- correspond à la **fréquence** des mots
- ne tient pas compte de la **rareté** des mots
- comparaison de documents repose par les mots fréquents

Re-calcul du **poids** des mots dans les documents

- poids de **fréquence inverse** d'un mot dans le corpus :

$$FI_{m,C} = \log\left(\frac{|C|}{|\{D \in C | m \in V_D\}|}\right)$$

avec

- nombre de documents dans le corpus : $|C|$
- nombre de documents qui contiennent un mot : $|\{D \in C | m \in V_D\}|$
- multiplication des fréquences des mots par leurs poids

Donne du poids aux mots **apparaissant dans peu de documents**

Extraction de termes

Exploitation des termes et des documents

- **fréquence** des termes dans les corpus et sous-corpus (TDF-IDF)
- présence de **collocations**
- liens entre les termes et les documents (graphes)
- **similarités** entre les termes ou entre les documents
- **position** des termes dans le texte
- répartition des termes dans les **phrases**

Quelques **calculs statistiques** utilisés

- TF-IDF
- vraisemblance (*log-likelihood*)
- test du chi-deux (χ^2)
- information mutuelle
- loi hypergéométrique

Calcul des spécificités

Reformulation de la loi hypergéométrique

$$P(X = k|T, P, N) = \frac{\binom{T}{k} * \binom{C-T}{P-k}}{\binom{C}{P}}$$

Avec

- fréquence observée du terme dans la partie k
- nombre total d'occurrence du terme T
- taille de la partie (pour tous les termes) P
- taille totale du corpus (pour tous les termes) C

Calcul de la spécificité

- pour les probabilités faibles de fréquences observées
- logarithme de la probabilité
 - spécificités **positives** : fréquence forte, sur-représentation
 - spécificités **négatives** : fréquence faible, sous-représentation

Allocation de Dirichlet latente

Recherche non-supervisée de thèmes (*topics*) dans les documents

Allocation de Dirichlet latente (Blei, Ng, Jordan, 2002)

- hypothèses **génératives**
 - les documents sont des mélanges des thèmes
 - les thèmes sont représentés par des termes
- initialisation aléatoire des thèmes dans les documents
- pour chaque terme de chaque document
 - probabilité que le document s soit généré par le thème t
 $P(t|d)$
 - probabilité que le terme m soit généré par le thème t
 $P(m|t)$
 - attribution du terme à un thème selon $P(t|d) * P(m|t)$

Calcul sur un corpus réparti en documents

- chaque document a une distribution des K thèmes
- chaque thème a une distribution de mots