Ingénierie de la fouille et de la visualisation de données massives (RCP216)

Fouille de données textuelles

Michel Crucianu
(prenom.nom@cnam.fr)
http ://cedric.cnam.fr/vertigo/Cours/RCP216/

Département Informatique
Conservatoire National des Arts & Métiers, Paris, France

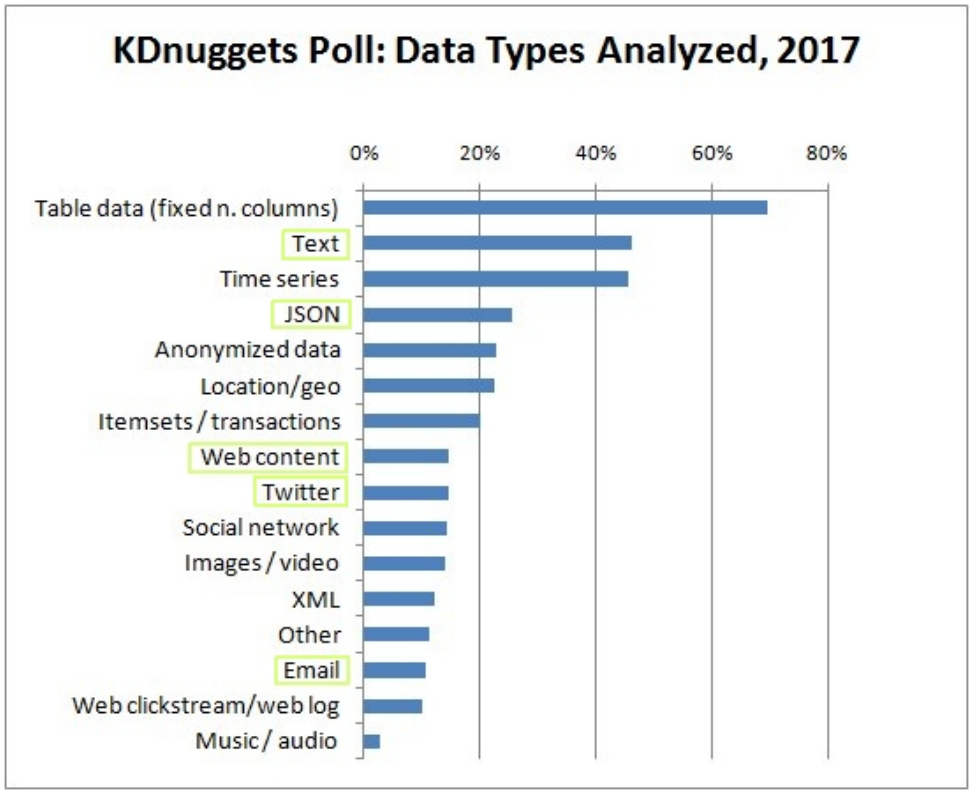
1 octobre 2021

Plan du cours

Fouille de données textuelles : définition et objectifs

- Fouille de données textuelles = processus d'extraction non triviale d'informations utiles inconnues *a priori* à partir de grands volumes de textes
- Spécificité : les données sont sous une forme qui n'est pas directement exploitable par les méthodes classiques de fouille de données
- Objectifs :
  - Identification de thèmes : regroupement de (parties de) textes en thèmes inconnus *a priori*
  - Affectation de (parties de) textes à des catégories (classes) prédéfinies
  - Recherche de « variables » explicatives utilisables ensuite conjointement avec d'autres variables (quantitatives, nominales)
  - Extraction d'informations : mise en correspondance des textes avec des « schémas » plus directement exploitables par des méthodes classiques de fouille de données
- Remarque importante : la fouille de textes protégés par le droit d'auteur (même en accès ouvert, par ex. sites web) est interdite (sans l'accord des auteurs) dans la plupart des pays européens ! Exception notable : Royaume-Uni.

Types de données traitées (source http ://www.kdnuggets.com 2017)



Fouille de données textuelles : exemples d'applications

1. Gestion de la relation client :
   - Détermination de catégories de clients à partir de leurs échanges avec le service client, redirection des courriels mal adressés
   - Identification de l'objet des retours négatifs fréquents, détermination des causes majeures de l'attrition de clientèle
   - Détermination de l'image d'une famille de produits
   - Détermination des attentes majeures dans l'évolution des produits
2. Identification de tendances à partir de messages postés sur des médias sociaux :
   - Produits ou familles de produits recherchés, caractéristiques recherchées pour des produits d'une certaine famille
   - Quantification de l'impact affectif de caractéristiques de produits, d'événements liés à des marques

Fouille de données textuelles : exemples d'applications (2)

3. Identification de supports adaptés pour la publicité en ligne :
   - Classification thématique de blogs ou (sous-)forums
   - Identification de caractéristiques recherchées pour des produits d'une certaine famille
4. Identification de risques sécuritaires par la surveillance d'échanges sur des réseaux sociaux :
   - Classification thématique avec suivi de l'évolution thématique et de l'activité
   - Quantification de l'impact affectif d'événements
5. Identification de concepts et liens entre concepts (par ex. pour aider la recherche scientifique) :
   - Analyse de littérature scientifique trans-communautés pour trouver des liens entre faits intra-communautés
   - Analyse de compte-rendus d'examens médicaux

Problème majeur : le « fossé sémantique » ; a un impact variable suivant l'objectif

Étapes dans la fouille de données textuelles

(certaines étapes ne sont pas systématiquement présentes dans la fouille de textes)

1. Collecte des données textuelles : identification des sources, récupération des contenus à partir des sources, extraction des textes (par ex. une page HTML peut contenir d'autres composants : scripts Javascript, menus…)
2. Pré-traitement des données textuelles : uniformisation du codage, élimination éventuelle de certains caractères spéciaux (sauts de lignes, symboles, etc. suivant l'objectif), « traduction » de langage SMS…
3. Extraction d'entités primaires : mots, éventuellement locutions nominales (« chemin de fer »), verbales (« arrondir les angles »)…
4. Étiquetage grammatical : caractérisation grammaticale de chaque composante du texte par une catégorie lexicale (ex. nom commun, nom propre, verbe, adverbe…) et une fonction (ex. sujet, complément d'objet direct…) ; souvent analyse superficielle

Étapes dans la fouille de données textuelles (2)

5. Extraction d'entités nommées : noms de personnes, de lieux (« Mont Blanc »), d'organisations, dates…
6. Résolution référentielle : par ex., à qui fait référence *Il* dans « Barack Obama est le 44e président des États-Unis. *Il* est né le 4 août 1961 à Honolulu »
7. Analyse syntaxique (générale ou spécifique) : identification de la négation ($\rightarrow$ distinction entre affirmation et négation), « quantification » des adverbes (ex. « *très* abouti / *plus ou moins* abouti / *peu* abouti »)
8. Extraction d'informations : suppression des « mots ignorés » (*stop words*) ; mise en relation avec des schémas d'interprétation

Étapes dans la fouille de données textuelles (3)

9. Lemmatisation ou racinisation : remplacer chaque mot (par ex. « pensons ») par sa forme canonique (« penser ») ou par sa racine (« pense ») ; peuvent engendrer des confusions (par ex. « organ » pour « organe » comme pour « organisation »)
10. Représentation vectorielle des textes : pondération *tf*idf*, décomposition en valeurs singulières (analyse sémantique latente, *LSA*), analyse sémantique explicite (*ESA*)…
11. Développement de modèles sur la base du contenu textuel seul ou en ajoutant des variables quantitatives et nominales
12. Utilisation des modèles développés, évaluation des résultats

Collecte et pré-traitement des données textuelles

1. Identification des sources : possible de regrouper des données issues de plusieurs sources
   - Textes bien formés : suite de phrases structurées mais *conformité grammaticale* variable
   - Mots-clés, *tags* : simples listes de mots, en général courtes, parfois bruitées (*tag spam*)
   - Transcription de SMS : fautes de frappe ou de sélection, émoticônes, langage codé, expressions *ad-hoc*
   - Transcription à partir de séquences audio : difficile, nombreuses erreurs ; meilleurs résultats si connexion entre transcription et modèles de langage, ou *word-spotting*, ou directement extraction d'informations suivant schémas
   - Important : la conformité grammaticale de la source est compatible avec l'objectif ? Par ex., l'extraction d'entités nommées, la résolution référentielle, l'analyse syntaxique exigent une bonne conformité

Collecte et pré-traitement des données textuelles (2)

2. Récupération des contenus à partir des sources : outils variés suivant les sources, allant de `wget` à `Hive`
3. Extraction des données textuelles : outils configurables pour la suppression d'autres contenus (par ex. scripts, menus…)
4. Pré-traitements : uniformisation du codage, élimination de certains caractères spéciaux (sauts de lignes, symboles, suivant l'objectif), traduction de langage SMS

Extraction d'entités primaires

- Entités primaires : mots simples ou composés (« chauve-souris »), ponctuation, éventuellement locutions nominales (« chemin de fer »), verbales (« arrondir les angles »)…
- Outil de segmentation (*tokenizer*) : utilise des règles (dépendantes de la langue) et un lexique
- Lexique : ensemble des *lemmes* (∼mots) d'une langue et/ou d'un domaine particulier
  - Informations additionnelles : morphologiques (formes possibles, à partir de racine et suffixes, préfixes), parfois syntaxiques
  - Peut souvent être enrichi par des lemmes et des locutions spécifiques au domaine
- L'extraction de certaines locutions peut exiger des traitements plus complexes que la recherche dans un lexique

Extraction d'entités primaires (2)

- Les lemmes extraits ne sont pas toujours directement associables à des *concepts* !
  - Synonymie : un concept associé à plusieurs lemmes différents (par ex. voiture et automobile)
  - Homonymie : un lemme associé à plusieurs concepts différents (par ex. avocat)
- Les relations entre concepts (sorte_de, partie_de…) sont définies dans des *ontologies*, qui incluent aussi des informations sur l'expression des concepts par des lemmes (dans une langue particulière)
- L'identification des concepts associés aux lemmes passe par une étape parfois difficile de désambiguïsation

Étiquetage grammatical

- Caractérisation grammaticale de chaque composante du texte par une catégorie lexicale (ex. nom commun, nom propre, verbe, adverbe…), le genre, nombre, mode, temps, etc. ; souvent par analyse superficielle
- Ex. : TreeTagger (http ://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/) peut apprendre des règles à partir d'exemples

```
La          DET :ART    le
fouille     NOM         fouille
de          PRP         de
données     NOM         donnée
peut        VER :pres   pouvoir
être        VER :infi   être
définie     VER :pper   définir
comme       ADV         comme
le          DET :ART    le
"           PUN :cit    "
Processus   NOM         processus
d'          PRP         de
...
.           SENT        .
```

Extraction d'entités nommées, résolution référentielle

- Entité nommée = élément du langage qui fait référence à une entité unique, appartenant éventuellement à un domaine spécifique
  - Exemples : noms de personnes (« Barack Obama Jr. »), de lieux (« Mont Blanc »), d'organisations (« Mouvement international de la Croix-Rouge et du Croissant-Rouge », « ICRC »), dates…
  - Approches : règles explicitement définies, apprentissage à partir d'un *corpus* annoté, approches hybrides
  - Difficultés : polysémie (« Washington » → la ville ou l'état ou la personne ?), métonymie (« l'Elysée » → la présidence de la République Française ou le palais ?) ⇒ le contexte doit être pris en compte
  - Exemple d'outil qui exploite DBpedia : `http://www.nuxeo.com/blog/mining-wikipedia-with-hadoop-and-pig-for-natural-language-processing/`
  - Résultats : extraction correcte d'env. 90% des entités nommées (si des ressources adaptées sont disponibles)

Extraction d'entités nommées, résolution référentielle (2)

- Résolution référentielle (coréférence) : relier entre elles les différentes formes sous lesquelles une même entité apparaît dans un texte (ou un ensemble de textes)
  - Exemples : « Barack Obama est le 44e président des États-Unis. *Il* est né le 4 août 1961 à Honolulu » ; « Google [...]. La société de Mountain View [...] »
  - Approches : règles spécifiques à des catégories d'entités nommées, voire à des entités nommées individuelles, analyse syntaxique
  - Résultats : très variables car les problèmes sont de nature diverse

Analyse syntaxique

- Complète : mettre en évidence la structure hiérarchique des phrases d'un texte (⇒ ensemble d'arbres syntaxiques)
  - Difficultés : mots hors lexique, structures non répertoriées par la grammaire, diverses formes d'ambiguïté

  ⇒ analyse « robuste » : proposer une analyse pour chaque élément et peu d'analyses alternatives

  ⇒ souvent analyse « de surface » qui ne met pas en évidence la structure complète et ne cherche pas à enlever toute ambiguïté ; extraction de *chunks* (groupes nominaux, groupes verbaux) et de relations simples entre eux
- Partielle : traiter des fragments spécifiques, qui présentent des constructions simples et avec peu de variabilité
  - Identification de la négation (→ distinction entre affirmation et négation)
  - Analyse limitée au voisinage de certains lemmes répertoriés (ex. entités nommées)
  - Traitement de certains adverbes (ex. « *très* utile / *plus ou moins* utile / *peu* utile »)

Extraction d'informations

- Objectif : extraire d'un texte des éléments ciblés par rapport à l'application ; exemples :
  - Faits : « Tôt le 25 avril 1974, au Portugal, des capitaines en rupture avec le système de Salazar se révoltent et prennent le pouvoir. »
  - Avis : « J'ai été terriblement déçu par ce roman. Je trouvais l'écriture fade et le style pseudo-éthérique très agaçant. »
- Approche générale : mise en relation avec des schémas d'interprétation (ou patrons sémantiques)

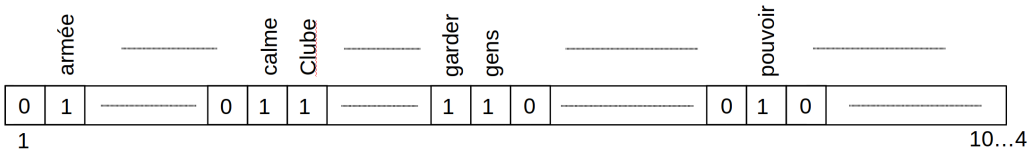
  → {Fait : *événement politique*}{{Où : *Portugal*}{Quand : *25/04/1974*}{Qui : *forces armées*}{Nature : *prise de pouvoir*}}
- Approche :
  - Définition du lexique et des entités nommées spécifiques, des schémas d'interprétation, des règles d'inférence spécifiques
  - Étiquetage grammatical, extraction d'entités nommées, résolution référentielle, analyse syntaxique locale, règles d'inférence issues d'une ontologie ou définies spécifiquement, règles pragmatiques (ex. mise en forme)…
- Exige en général un travail spécifique conséquent

Lemmatisation ou racinisation

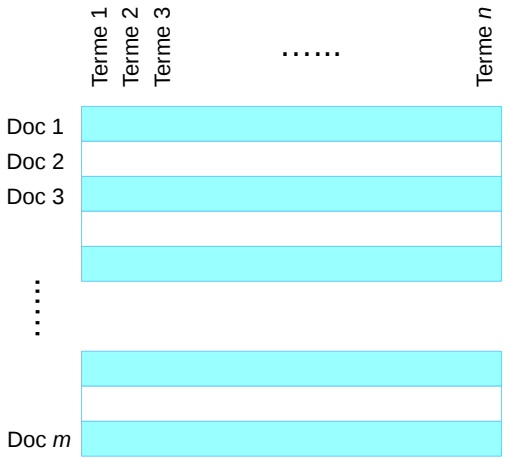
- Objectif : représentation unique pour les formes fléchies d'un même lemme
- Racinisation :
  - Remplacer un mot (par ex. « pensons ») par sa racine (« pense »)
  - En général basée sur des règles et un lexique
  - Peut engendrer des confusions, par ex. « mange » pour « mangeable » comme pour « immangeable », « organ » pour « organe » comme pour « organisation »)
- Lemmatisation :
  - Remplacer un mot (par ex. « pensons ») par sa forme canonique (« penser »)
  - Basée sur l'analyse lexicale, utilise l'étiquetage grammatical
- Racinisation suffisante pour l'anglais, lemmatisation mieux adaptée au français

Représentations vectorielles de textes

- Objectif : pouvoir manipuler des données textuelles avec les nombreux outils disponibles pour les espaces vectoriels
- Au préalable, possible suppression des « mots ignorés » (*stop words*) : prépositions, conjonctions, articles, verbes auxiliaires…
  - L'ensemble des mots à ignorer peut dépendre de l'objectif de l'analyse !
  - Doit être appliquée seulement *après* l'extraction de locutions (ex. « chemin *de* fer »)
- Modèle vectoriel de texte [19] : affecter une dimension de l'espace à chaque terme (lemme, entité nommée…) trouvé dans la base de documents → chaque texte (document, requête) est représenté par un vecteur de grande dimension, (très) creux :



Modèle vectoriel de texte



- Comparaison des vecteurs avec la distance cosinus : la norme du vecteur étant proportionnelle à la longueur du texte, mieux vaut mesurer l'angle entre vecteurs que la distance euclidienne
- Evolutions de la représentation vectorielle de base : pondérations (par ex. *TF-IDF*), sélection de termes (test du χ^2), analyse sémantique latente (*LSA*), ...

Modèle vectoriel de texte avec pondérations *TF-IDF*

- Des pondérations spécifiques, prédéfinies, peuvent être utilisées, par ex. sur-pondérer les termes des titres de sections et sous-sections, sur-pondérer certaines entités nommées, etc.
- Objectif *TF-IDF* : pondérer les termes suivant leur « importance » déterminée automatiquement ; surtout pour la recherche d'informations
- Fréquence d'un terme dans un document (*term frequency*, TF) : l'importance d'un terme pour un document est proportionnelle au nombre d'occurrences du terme dans le document

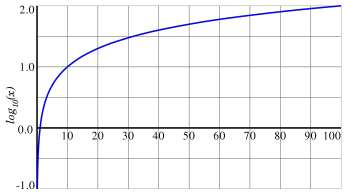
$$tf_{ij} = \frac{n_{ij}}{\|d_j\|}$$

n_{ij} étant le nombre d'occurrences du terme i dans le document j et $\|d_j\|$ la longueur du document d_j (la somme des occurrences de tous les termes qu'il contient)

Modèle vectoriel de texte avec pondérations *TF-IDF* (2)

- Inverse de la fréquence dans les documents (*inverse document frequency*, IDF) : l'importance d'un terme pour tous les documents est inversement proportionnelle au nombre de documents dans lequel il apparaît (les termes présents dans peu de documents sont plus discriminants que les termes présents dans beaucoup de documents)

$$idf_i = \log\left(\frac{n}{n_i}\right)$$



n étant le nombre total de documents, n_i le nombre de ceux contenant le terme i

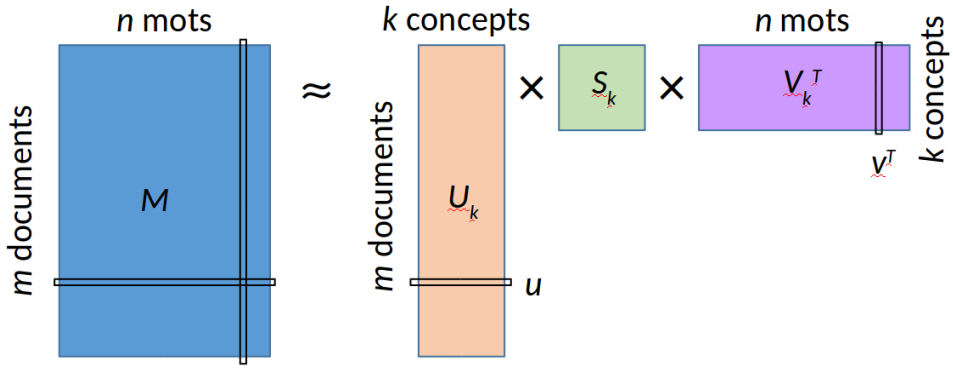
- Le logarithme permet d'éviter une sur-pondération des termes très rares, d'ailleurs en général on exclut les termes dont le nombre d'occurrences dans la collection de documents est inférieur à un seuil
- Des variations existent pour *tf* et surtout pour *idf*

$\Rightarrow$ Au total, la pondération du terme i dans le document j sera $tf_{ij} \cdot idf_i$

Représentation vectorielle : analyse sémantique latente [6]

- Objectif : recherche de « concepts », en (relativement) faible nombre, correspondant à des corrélations entre termes, pour représenter les documents d'une collection
  - Remplacer les lemmes individuels par des « concepts » correspondant à des groupes de lemmes souvent présents ensembles (identifiés par LSA et non *a priori*)
  - Réduction du « bruit » engendré par les mots rares, l'utilisation accidentelle de mots inappropriés, etc.
- Approche : décomposition en valeurs singulières (SVD) appliquée à la matrice documents-termes, suivie par une réduction de rang
  - Matrice documents-termes $\mathbf{M}$: 1 ligne par document, 1 colonne par terme
  - SVD : $\mathbf{M} = \mathbf{U} \cdot \mathbf{S} \cdot \mathbf{V}^T$, $\mathbf{U}$ et $\mathbf{V}$ étant des matrices orthogonales et $\mathbf{S}$ une matrice diagonale
  - En calculant $\mathbf{M}\mathbf{M}^T$ et $\mathbf{M}^T\mathbf{M}$ on constate que $\mathbf{S}$ contient les racines carrées des valeurs propres de $\mathbf{M}\mathbf{M}^T$ (ou $\mathbf{M}^T\mathbf{M}$), $\mathbf{U}$ contient les vecteurs propres de $\mathbf{M}\mathbf{M}^T$ et $\mathbf{V}$ les vecteurs propres de $\mathbf{M}^T\mathbf{M}$
  - Réduction de rang : on considère les k plus grandes valeurs propres et les vecteurs propres associés, alors $\mathbf{M}_k = \mathbf{U}_k \cdot \mathbf{S}_k \cdot \mathbf{V}_k^T$

Représentation vectorielle : analyse sémantique latente (2) [6]



- Concept (latent) = groupe de mots (lemmes) présents souvent ensemble dans des documents (= groupe de variables corrélées dans la matrice $\mathbf{M}$), par ex. :

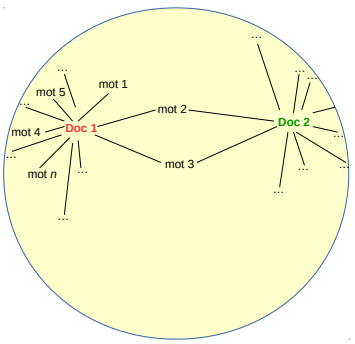
  {procès, juge, tribunal, plainte, inculpé, procureur, avocat, condamnation}
  {assiette, couteau, chef, mayonnaise, avocat, crevettes, plateau}

- Évolutions de l'idée : analyse sémantique latente probabiliste (*PLSA*, [9]), allocation de Dirichlet latente (*LDA*, [1])

Représentation vectorielle : analyse sémantique explicite [8]

- Représentations basées sur la matrice documents-termes → issues de relations entre documents (et entre termes), très dépendantes de l'ensemble de documents dont la matrice est issue → nombreux biais de modélisation
- Analyse sémantique explicite (*Explicit Semantic Analysis*, ESA [8]) :
  - Utiliser un corpus très grand (Wikipedia) pour construire la matrice documents-termes, représenter chaque terme par sa colonne dans la matrice

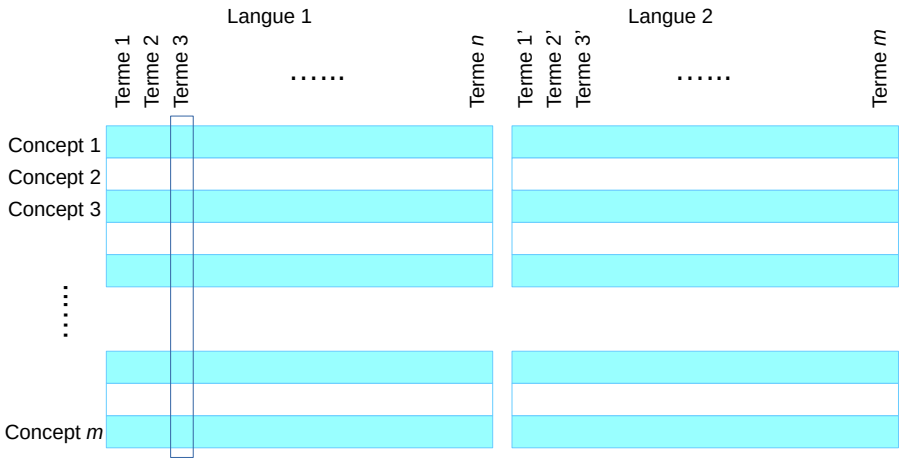
  ⇒ représentations générales et riches des termes, indépendantes d'ensembles spécifiques de documents applicatifs
  - Tout autre document est représenté par le centre de gravité de l'ensemble des mots qu'il contient (pondérations TF-IDF prises en compte) :



  - Contrairement aux résultats de LSA, avec ESA les dimensions des vecteurs sont *interprétables* (dans Wikipedia chaque document décrit un concept bien déterminé)

Représentation vectorielle : analyse sémantique explicite [8] (2)

- Possibilité d'utiliser un corpus multi-langues (par ex. Wikipedia) pour définir des représentations de documents indépendantes de la langue (exprimées comme des vecteurs dans l'espace des concepts Wikipedia) → *Cross-language* ESA



- Difficultés : biais présents dans Wikipedia (domaines, terminologie), disponibilité d'un corpus suffisant (et peu biaisé) dans une langue particulière

Représentation vectorielle des *mots* : Word2Vec [11, 12]

- Objectif : grand corpus de textes ⇒ représentations vectorielles des mots qui incorporent des caractéristiques sémantiques et syntaxiques
- Deux méthodes différentes utilisant des réseaux de neurones à une couche cachée :
  1. Méthode *continuous bag of words* (CBOW) : prédire chaque mot à partir de son contexte
  2. Méthode Skip-gram : à partir de chaque mot prédire son contexte
- Constats :
  - CBOW est moins coûteuse mais Skip-gram donne en général de meilleurs résultats (surtout lorsque le corpus de textes utilisé n'est pas très grand)
  - Avec cette représentation, les mots se regroupent par similarité de contexte
  - Une forme de « additivité » : la représentation la plus proche du résultat du calcul `vec(Madrid)` - `vec(Spain)` + `vec(France)` est `vec(Paris)`

→ Les bons résultats obtenus pour des problèmes très divers (désambiguïsation, traduction de textes, classification de textes, etc.) ont suscité de nombreux travaux qui ont mis au point d'autres méthodes

Autres représentations vectorielles de mots

Méthodes qui construisent, comme Word2Vec, une représentation unique pour chaque mot (plongement lexical ou *word embedding*), quel que soit son contexte et son sens dans une phrase (liste non exhaustive) :

- GloVe [14] :
  - Traitement direct de la matrice (très creuse) de cooccurrence des mots dans des fenêtres de contexte
  - Fonction à minimiser : écarts entre le produit scalaire des *embeddings* et le $\log$ de la probabilité de cooccurrence
  - Comme Word2Vec, permet d'obtenir des *word embeddings* seulement pour les mots rencontrés dans les textes du corpus traité (mots du vocabulaire)
- FastText [2, 10] : vise à résoudre le problème des mots hors vocabulaire
  - Méthode similaire à Word2Vec, sauf que l'unité de base n'est pas le mot mais le *n*-gramme de caractères
  - Hypothèses : la combinaison des *embeddings* des *n*-grammes est un bon *embedding* d'un mot, les mots hors vocabulaire sont composés de *n*-grammes rencontrés dans les mots du corpus
  - Le *embedding* d'un mot est obtenu à partir des *embeddings* de ses *n*-grammes (par ex. pour le mot `vecteur` les trigrammes sont {vec,ect,cte,teu,eur})

Autres représentations vectorielles de mots (2)

Méthodes qui construisent une représentation différente pour chaque mot dans chaque contexte (*language model* ; liste non exhaustive) :

- ELMo (*Embedding from Language Models*) [15] :
  - Entrées : *embeddings* des caractères successifs, suivis d'une couche convolutionnelle ($\rightarrow$ *n*-grammes de caractères) et 2 couches de *highway network*
  - Suivent plusieurs niveaux de LSTM (réseaux de neurones récurrents) vers la gauche et vers la droite, les représentations étant concaténées à chaque niveau
  - Les représentations de mots résultantes sont obtenues comme des combinaisons linéaires des entrées et des représentations issues des différents niveaux
- **BERT** [7] (*Bidirectional Encoder Representations from Transformers [20]*) :
  - Apprentissage sur : prédiction de mots masqués, phrase suivante ou non
  - Après, possible apprentissage supervisé sur différentes tâches cibles (classement de phrase, réponse aux questions, étiquetage de phrase, coréférence…)
  - Pour le français : CamemBERT, FlauBERT
- GPT [16] (*Generative Pre-Trained transformer*), **GPT-2** [17], **GPT-3** [3] :
  - Apprentissage sur la prédiction du mot suivant et plusieurs tâches supervisées
  - GPT-3 appris sur CommonCrawl filtré ($\sim$ 570 GB) $\rightarrow$ jusqu'à $175 \cdot 10^9$ paramètres
  - Grand potentiel de mauvais usage [3] $\Rightarrow$ GPT-3 175 n'est pas *open source*

Quelles représentations pour les textes ?

1. Représentations exploitant les représentations vectorielles des mots individuels (indépendantes ou dépendantes du contexte) – liste non exhaustive :
   1. Centre de gravité des vecteurs représentant ses mots, en général en supprimant les *stop words* et parfois en pondérant les mots (par ex. pondérations TF-IDF)
   2. Sent2Vec [13] : extension du contexte d'un mot à une phrase entière (ou un paragraphe court), vecteur phrase = centre de gravité des vecteurs représentant les mots individuels *et* n-grammes de la phrase
   3. InferSent [5] : réseaux LSTM bidirectionnels, apprentissage de discrimination phrase suivante (*entailment/contradiction/neutral*) sur la base Stanford NLI
   4. *Universal Sentence Encoder* (USE) [4] avec deux variantes :
      1. *Deep Averaging Network* (réseau de neurones à 4 couches), prend en entrée les *embeddings* des mots et des bi-grammes de la phrase
      2. *Transformer encoder* (6 niveaux de *transformers*), prend en entrée les *embeddings* des mots ; meilleurs résultats mais coût plus élevé (complexité quadratique dans la longueur de la phrase)
   5. *Sentence-BERT* (SBERT, voir aussi `https://www.sbert.net/`) [18] :
      - Utilise deux réseaux BERT siamois pré-entraînés et fait du *fine-tuning* ⇒ coût limité
      - Trois fonctions objectif (suivant information de supervision disponible) : discrimination phrase suivante (*entailment/contradiction/neutral*), régression similarité cosinus, similarité triplet

Quelles représentations pour les textes ? (2)

2. Représentations issues du modèle vectoriel de texte
   - 1 lemme = 1 variable (1 colonne de la matrice documents-mots), aucun lien entre deux colonnes même si les lemmes sont liés (par ex. synonymes)
   - Textes longs mieux représentés que textes courts qui n'ont pas assez de lemmes

   1. Modèle vectoriel de base : représentation simpliste qui n'est plus utilisée
   2. Modèle vectoriel avec pondérations TF-IDF : plus fidèle grâce aux pondérations mais ignore les liens entre mots différents
   3. Analyse sémantique latente : réduction du « bruit », prise en compte des similarités entre mots (par ex. synonymes) à travers leurs corrélations avec d'autres mots
   4. Analyse sémantique explicite (aussi basée sur *word embeddings* !) : représentations obtenues sur grand corpus générique, possibilité de comparer des documents multi-langues

Quelles représentations pour quels textes ?

1. Texte court ou petit ensemble de mots clés ($< 10 - 15$ termes) :
   - Le centre de gravité des vecteurs représentant les mots constitue une solution facile, dans ce cas préférer les représentations non-contextuelles comme GloVe (vocabulaire prédéfini) ou FastText (traite les termes hors vocabulaire)
   - USE ou SBERT sont des solutions plus complexes à mettre en œuvre et un peu plus coûteuses mais plus performantes, si disponibles pour la langue cible
   - Les représentation issues du modèle vectoriel de texte sont à éviter car trop pauvres. Exception : l'analyse sémantique explicite surtout pour documents multi-langue
2. Texte plus long :
   - Centre de gravité des vecteurs représentant les mots : centre de gravité devient peu spécifique (nombre élevé de mots) donc peu discriminant $\Rightarrow$ approche à éviter
   - USE et SBERT : méthodes de choix pour textes de longueur moyenne (~ 1 paragraphe)
   - Les représentation issues du modèle vectoriel de texte (surtout LSA et ESA), plus faciles à mettre en œuvre, peuvent donner des résultats satisfaisants pour textes plus longs

Quelques environnements logiciels

Évolution rapide. API en général utilisables à partir de plusieurs langages.

1. SparkNLP https ://nlp.johnsnowlabs.com/
   - Écrite en scala, intégration plus efficace avec Spark ML (API *DataFrame*)
   - Segmentation (*tokenizer*), étiquetage (*tagger*), lemmatisation, extraction d'entités nommées, *entity linking*, différents outils de construction de représentations (*embedding*), analyse de sentiments, etc.
2. spaCy https ://spacy.io/ (voir aussi https ://course.spacy.io/en/)
   - Écrite en Python (et Cython pour meilleure efficacité)
   - Segmentation (*tokenizer*), étiquetage (*tagger*), lemmatisation, extraction d'entités nommées, *entity linking*, différents outils de construction de représentations (*embedding*), etc.
3. Stanford CoreNLP http ://nlp.stanford.edu/software/
   - Écrite en Java mais utilisable à partir d'autres langages
   - Segmentation (*tokenizer*), étiquetage (*tagger*), lemmatisation, extraction d'entités nommées, *entity linking*, *parsing*, co-référence, etc.

- Exemple de comparaison concrète entre SparkNLP et SpaCy : https ://medium.com/spark-nlp/spacy-or-spark-nlp-a-benchmarking-comparison-23202f12a65c

Opérations liées à la fouille de textes dans Spark

- Modèle vectoriel de texte avec pondérations TF-IDF : `HashingTF, IDF`
- Sélection de termes par test du χ^2 : `ChiSqSelector`
- Analyse Sémantique Latente : implémentation externe utilisant `SVD` (voir travaux pratiques)
- Allocation de Dirichlet latente (LDA) : `LDA, DistributedLDAModel`
- Représentation vectorielle Word2Vec : `Word2Vec, Word2VecModel`

Références I

[1] D. M. Blei, A. Y. Ng, and M. I. Jordan.
Latent dirichlet allocation.
J. Mach. Learn. Res., 3 :993–1022, Mar. 2003.

[2] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov.
Enriching word vectors with subword information.
Transactions of the Association for Computational Linguistics, 5 :135–146, 2017.

[3] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei.
Language models are few-shot learners, 2020.

[4] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. St. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, B. Strope, and R. Kurzweil.
Universal sentence encoder for English.
In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing : System Demonstrations*, pages 169–174, Brussels, Belgium, Nov. 2018. Association for Computational Linguistics.

Références II

[5] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes.
Supervised learning of universal sentence representations from natural language inference data.
CoRR, abs/1705.02364, 2017.

[6] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman.
Indexing by latent semantic analysis.
JASIS, 41(6) :391–407, 1990.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova.
Bert : Pre-training of deep bidirectional transformers for language understanding, 2018.

[8] E. Gabrilovich and S. Markovitch.
Computing semantic relatedness using wikipedia-based explicit semantic analysis.
In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 1606–1611, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[9] T. Hofmann.
Probabilistic latent semantic indexing.
In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 50–57, New York, NY, USA, 1999. ACM.

Références III

[10] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov.
Bag of tricks for efficient text classification.
In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics : Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics, April 2017.

[11] T. Mikolov, K. Chen, G. Corrado, and J. Dean.
Efficient estimation of word representations in vector space.
CoRR, abs/1301.3781, 2013.

[12] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean.
Distributed representations of words and phrases and their compositionality.
In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[13] M. Pagliardini, P. Gupta, and M. Jaggi.
Unsupervised learning of sentence embeddings using compositional n-gram features.
CoRR, abs/1703.02507, 2017.

[14] J. Pennington, R. Socher, and C. D. Manning.
Glove : Global vectors for word representation.
In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

Références IV

[15] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer.
Deep contextualized word representations.
In *Proc. of NAACL*, 2018.

[16] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever.
Improving language understanding by generative pre-training, 2018.

[17] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever.
Language models are unsupervised multitask learners.
OpenAI blog, 1(8) :9, 2019.

[18] N. Reimers and I. Gurevych.
Sentence-BERT : Sentence embeddings using siamese BERT-Networks.
CoRR, abs/1908.10084, 2019.

[19] G. Salton, A. Wong, and C. S. Yang.
A vector space model for automatic indexing.
Commun. ACM, 18(11) :613–620, Nov. 1975.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin.
Attention is all you need, 2017.