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

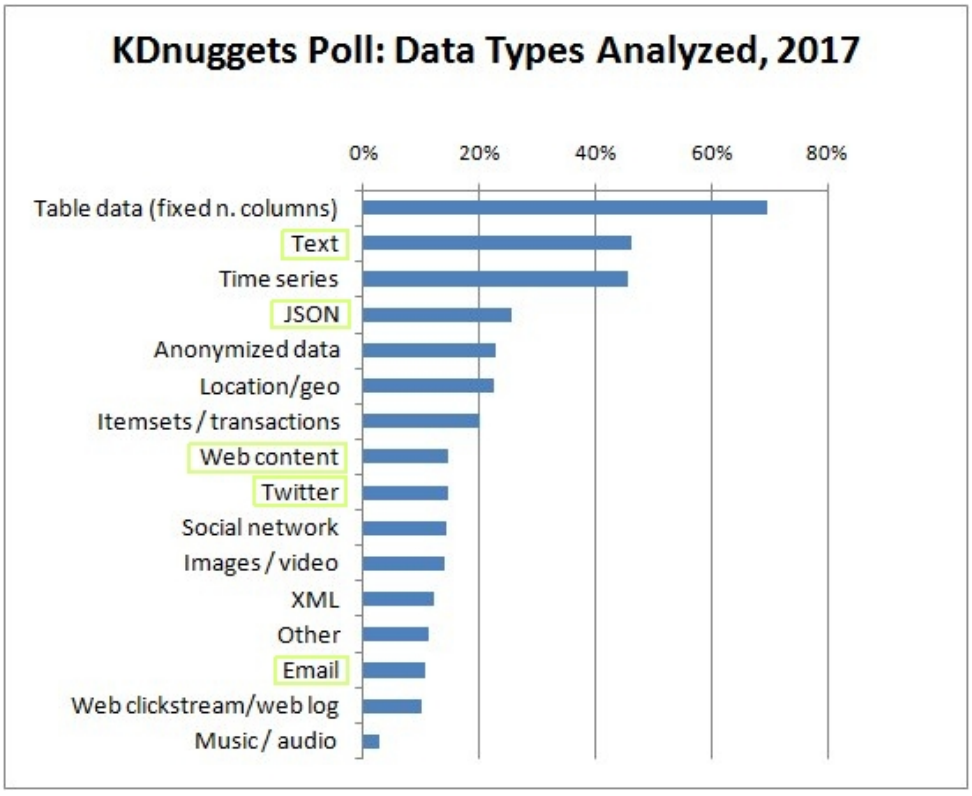
30 octobre 2018

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 entités nommées, 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)

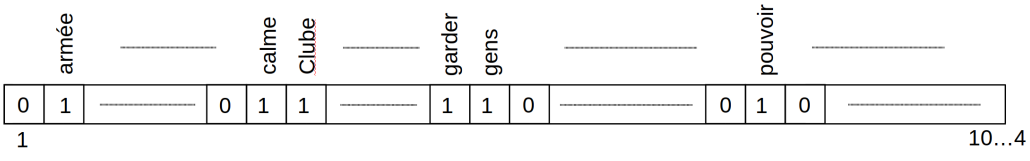
  → {Fait : *événement politique*}{{Où : *Portugal*}{Quand : *25/04/1974*}{Qui : *forces armées*}{Nature : *prise de pouvoir*}}
- Approche :
  - Définition du lexique et des entités nommées spécifiques, des schémas d'interprétation, des règles d'inférence spécifiques
  - Étiquetage grammatical, extraction 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

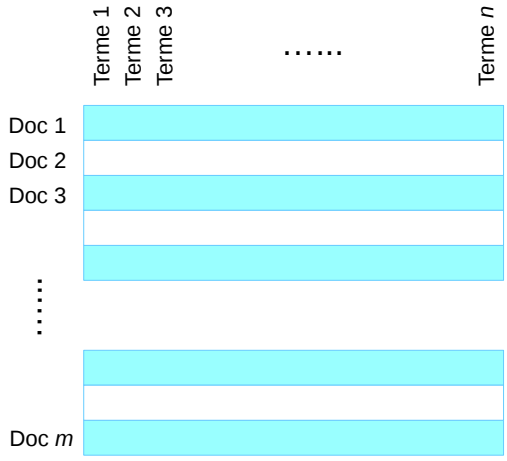
- Objectif : représentation unique pour les formes fléchies d'un même lemme
- Racinisation :
  - Remplacer un mot (par ex. « pensons ») par sa racine (« pense »)
  - En général basée sur des règles et un lexique
  - Peut engendrer des confusions, par ex. « mange » pour « mangeable » comme pour « immangeable », « 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

Représentation vectorielle des 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 »)
- Principe : affecter une dimension de l'espace à chaque lemme trouvé dans la base de documents → chaque texte (document, requête) est représenté par un vecteur de (très) grande dimension, (très) creux :



Représentation vectorielle des textes (2)



- 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 : pondération *TF-IDF*, sélection de mots (test du χ^2), analyse sémantique latente (*LSA*), analyse sémantique explicite (*ESA*), Word2Vec [6, 7]...

Représentation vectorielle : *TF-IDF*

- Objectif : pondérer les termes suivant leur « importance » ; surtout pour la recherche d'informations
- Fréquence des termes (*term frequency*) : 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
- Inverse de la fréquence dans les documents (*inverse document frequency*) : 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 et n_i le nombre de documents 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
  - De nombreuses variations existent, notamment pour *idf*
- Au total, la pondération du terme i dans le document j sera $tf_{ij} \cdot idf_i$

Représentation vectorielle : analyse sémantique latente [2]

- Objectif : recherche de « concepts », correspondant à des corrélations entre termes, pour représenter les documents d'une collection
  - Suppression de « bruit » présent dans les données, comme l'utilisation accidentelle de mots inappropriés (ou dont un homonyme est beaucoup plus fréquent)
  - Remplacer les lemmes individuels par des concepts correspondant à des usages similaires (identifiés par LSA et non à travers une ontologie)
- 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 $\mathbf{M}_k = \mathbf{U}_k \cdot \mathbf{S}_k \cdot \mathbf{V}_k^T$
- Évolutions de l'idée : analyse sémantique latente probabiliste (*PLSA*, [4]), allocation de Dirichlet latente (*LDA*, [1])

Représentation vectorielle : analyse sémantique explicite [3]

- Représentations basées sur la matrice documents-termes $\rightarrow$ relations entre documents (et entre termes) très dépendantes de l'ensemble de documents dont la matrice est issue $\rightarrow$ nombreux biais de modélisation
- Analyse sémantique explicite (*Explicit Semantic Analysis*, ESA [3]) :
  - Utiliser un corpus très grand (Wikipedia, en général) pour construire une matrice documents-termes

  $\Rightarrow$ 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)
- 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) $\rightarrow$ *Cross-language* ESA
- Difficultés : biais présents dans Wikipedia (domaines, terminologie), disponibilité d'un corpus suffisant (et peu biaisé) dans une langue particulière

Représentation vectorielle : Word2Vec [6, 7]

- Objectif : obtenir, à partir d'un grand corpus de textes, des représentations vectorielles des mots qui incorporent des caractéristiques sémantiques et syntaxiques
- Modèle Skip-gram [6] : trouver des représentations permettant de prédire le mieux possible le contexte des mots
  - Étant donnée une séquence de mots $w_1, w_2, \ldots, w_T$, maximiser

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{j=-k}^{j=k}\log p(w_{t+j}|w_t)$$

    k étant la largeur du contexte autour de chaque mot w_t
  - Plusieurs solutions permettent de réduire le coût de calcul
- Constats :
  - 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)`
- Ces représentations sont utilisées avec de bons résultats pour résoudre des problèmes très divers : désambiguïsation, traduction de textes, classification de textes, illustration de textes, etc.

Word2Vec pour des textes ?

Représentation d'un texte :

1. Texte court ou petit ensemble de mots clés (< 10) : centre de gravité des vecteurs Word2Vec représentant ses mots, en général en supprimant les *stop words* et parfois en pondérant les mots (par ex. avec TF-IDF)
2. Texte long : si le nombre de mots est assez élevé alors le centre de gravité devient peu spécifique, d'autres méthodes sont préférables
   - Sent2Vec [8] : extension du contexte d'un mot à une phrase entière (ou un paragraphe, voire un document), vecteur phrase = centre de gravité des vecteurs représentant les mots *et* n-grammes de la phrase
   - Vecteurs de Fisher construits à partir des représentations Word2Vec des mots [5] :
     - À partir des vecteurs (ici Word2Vec) issus des textes de la base, un modèle de mélange (en général gaussien) est construit
     - Le vecteur de Fisher d'un ensemble de vecteurs (ici Word2Vec) est la concaténation des gradients de la log-vraisemblance de cet ensemble par rapport aux paramètres du modèle ($\rightarrow$ très grande dimension)

Environnements logiciels

- Apache OpenNLP `https://opennlp.apache.org`
  - Suite intégrée en java, au départ pour l'anglais
  - Segmentation (*tokenizer*), étiquetage (*tagger*), lemmatisation, extraction d'entités nommées, catégorisation de documents, *parsing*, co-référence
  - Pour le français : segmentation, étiquetage, extraction d'entités nommées (`https://sites.google.com/site/nicolashernandez/resources/opennlp`, `http://www.nuxeo.com/blog/mining-wikipedia-with-hadoop-and-pig-for-natural-language-processing/`)
  - Intégrée à *Unstructured Information Management applications* (UIMA), `https://uima.apache.org`
- Stanford CoreNLP `http://nlp.stanford.edu/software/`
  - Suite intégrée en java pour l'anglais, l'espagnol, le chinois
  - Segmentation (*tokenizer*), étiquetage (*tagger*), lemmatisation, extraction d'entités nommées, *parsing*, co-référence
- SparkNLP https://nlp.johnsnowlabs.com/
  - Intégration plus efficace avec Spark ML (API *DataFrame*)
  - Segmentation (*tokenizer*), étiquetage (*tagger*), lemmatisation, extraction d'entités nommées, analyse sentiments ; extensible

Opérations liées à la fouille de textes dans Spark

- Représentation vectorielle TF-IDF : `HashingTF, IDF`
- Sélection de mots 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

D. M. Blei, A. Y. Ng, and M. I. Jordan.

Latent dirichlet allocation.

J. Mach. Learn. Res., 3 :993–1022, Mar. 2003.

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.

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.

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.

B. Klein, G. Lev, G. Sadeh, and L. Wolf.

Associating neural word embeddings with deep image representations using fisher vectors.

In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 4437–4446, 2015.

Références II

T. Mikolov, K. Chen, G. Corrado, and J. Dean.

Efficient estimation of word representations in vector space.

CoRR, abs/1301.3781, 2013.

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.

M. Pagliardini, P. Gupta, and M. Jaggi.

Unsupervised learning of sentence embeddings using compositional n-gram features.

CoRR, abs/1703.02507, 2017.