

Topic Modeling

Alexis Perrier

Lundi 22 mai 2017

Qui suis-je ?

- Telecom Paris 92
- These 95 sur gradient stochastique
- ...
- Quelques années passent
- ...
- 2011 Move to Boston, Berklee College of Music
- 2017 Lead Data Scientist, DocentHealth.com

Twitter: [@alexip](https://twitter.com/alexip)

[Linkedin.com/in/alexisperrier](https://www.linkedin.com/in/alexisperrier)

alexis.perrier@gmail.com



Plan

Github: <https://github.com/alexperrier/upem-topic-modeling/>

I: Topic Modeling

- Nature et applications
- Approche deterministe: LSA
- Approche probabiliste: LDA
- Librairies en R et python

II: LAB - R STM

- Corpus articles tech
- Package STM
- Nombre optimal de topics
- Interpretation des résultats

III: Projet: Forum Alt-right Pro Trump

- Le Corpus brut
- Premier résultats sont mauvais
- Quels sont les problèmes?
- Comment améliorer le corpus?
 - NER
 - POS
 - Stopwords
 - Et beaucoup de cuisine
 - Exemples en Python

IV: Application au français

- Lemmatization
- POS
- NER

Qu'est ce que le Topic Modeling



Quels sont les sujets abordés dans un ensemble (large) de documents?

- Extraction des sujets de façon **non-supervisée**
- Ce n'est pas de la classification de document
- Aucun a priori sur le contenu des documents

⇒ Croisement avec des **variables externes**

⇒ Evolution dans le temps

⇒ Auteur ou Locuteur, parti politique, age du capitaine, info démographique,

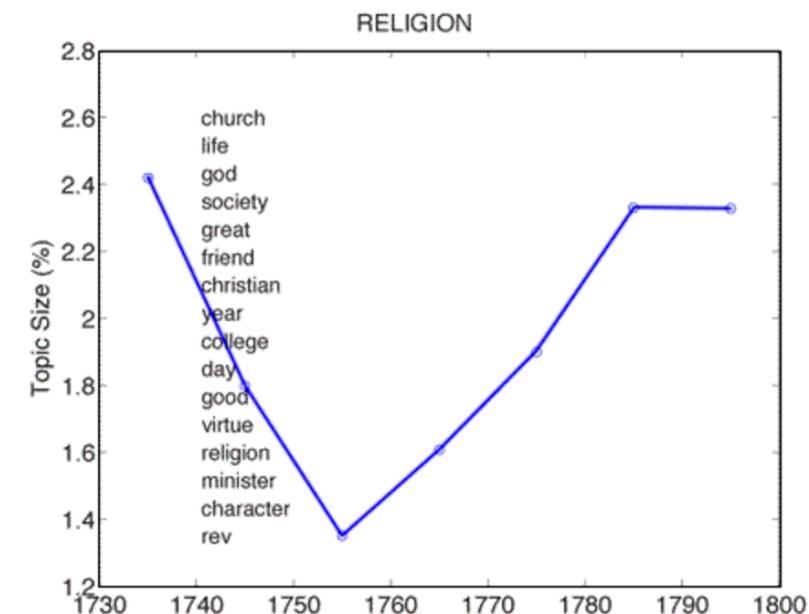
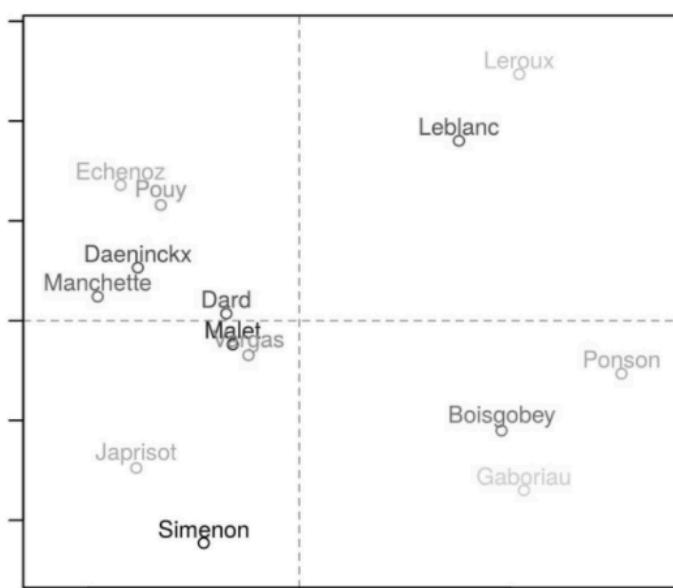
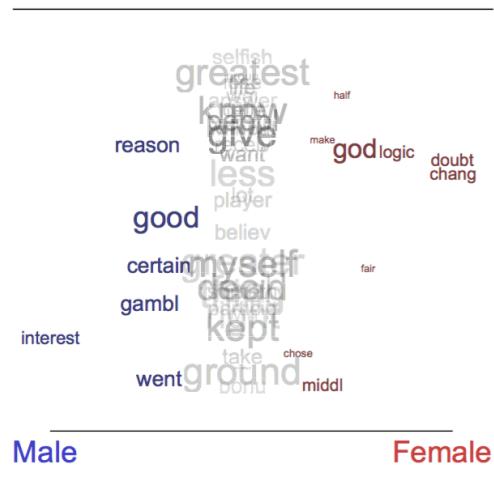
Exemples

- [Followers sur twitter](#) (gensim, LDA, LSA, python)
 - [Débats présidentiels américains](#) (stm, LDA, R)
 - Analyse des notes en milieu hospitalier pour prédire *patient satisfaction* et identifier les problèmes

Autres travaux:

- Gazette de Pensylvanie <http://www.common-place-archives.org/vol-06/no-02/tales/>
 - 3,346 works of 19th-century British, Irish, and American fiction evolution dans le temps
 - Analyse graph / réseau des topics dans Proust (Stanford, Mallet, Java)
 - Structural Topic Models for Open-Ended Survey Responses (Molly Roberts, STM)
 - Topic modeling of French crime fiction novels

FIGURE 15 Intuitive Topic Allowing for Different Vocabularies Based on Gender



Approche

1. Construction et définition du Corpus
 2. Définir l'unité d'analyse: tweet, commentaire, paragraphe, article, blog, livre, ...
 3. Technique utilisée: LSA, LDA, pLSA, HDL
 4. Variable externes

Etapes:

1. Pre-processing:
 1. Enlever le bruit, garbage-in garbage-out
 2. Adapter, transformer le contenu
 2. Topic Modeling: faire tourner le modèle
 3. Interprétation des résultats
 1. Nommer les topics
 2. Mesurer leur qualité
 4. Visualisation
 5. Wordcloud!

Post-Discharge Call



Corpus

Tous les corpus ne sont pas équivalents

- **Courts vs long:** litterature, documents officiels et articles de journaux vs tweets / FB
- **Langue:** anglais, francais, franglais
- **Niveau de langue:** classique, speech to text, argot, smiley
- **Contenu:** images, urls, sms, phrase, paragraphe, article, livre

« Nous avons des devoirs envers notre pays.

Nous sommes les héritiers d'une grande histoire et du grand message humaniste adressé au monde.

Nous devons les transmettre d'abord à nos enfants, mais plus important encore, il faut les porter vers l'avenir et leur donner une sève nouvelle. »

Emmanuel Macron, 08 mai 2017



s u h o 🎂
@kaidyoxe

Follow

Replying to @lchexo

j'avais mm pas vu j'sais pas ou g foutu mes écouteurs mais wsh crache tte ta haine du monde j'tecouterai tjrs

Translate from French

Bag of Words

L'information prise en compte est intégralement contenue dans la liste de mots issue du texte, indépendamment de leur position ou de leur fonction dans la phrase

Le texte original:

« Il sera une page dans un livre de dix mille pages
que l'on mettra dans une bibliothèque qui aura un million de livres,
une bibliothèque parmi un million de bibliothèques. »
Ionesco, Le roi se meurt 1962

devient:

aura bibliothèque bibliothèque bibliothèques dans dans de de de
dix il on I livre livres mettra mille million million
page pages parmi que qui sera un un un une une une

Vectoriser le texte avec Tf-Idf

Comment évaluer la fréquence des mots dans un ensemble de documents?

TF-IDF: fréquence du mot dans un document, normalisée par sa présence dans l'ensemble du corpus

Pour un document donné:

- **TF: Term Frequency:** combien de fois le terme est dans le document
- **IDF: Inverse document term frequency:**
de documents / # documents contenant le terme

On obtient une matrice terme – document comme illustrée =>

Voir:

- <https://fr.wikipedia.org/wiki/TF-IDF>
- [TfidfVectorizer de scikit-learn](#)

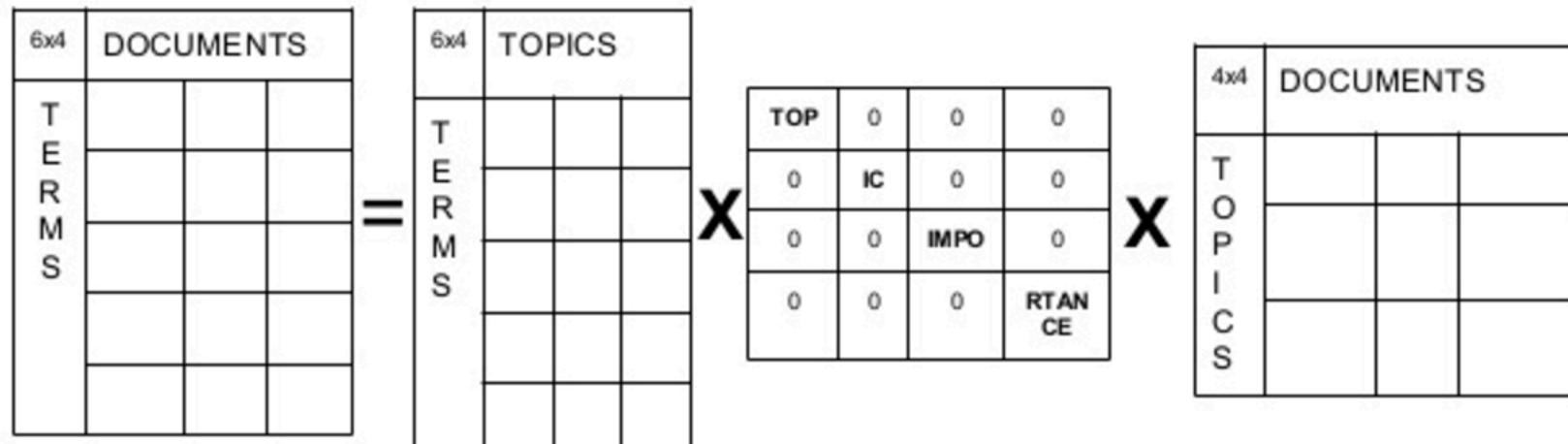
Term	Document					
	$d1$	$d2$	$d3$	$d4$	$d5$	$d6$
course	1	0	0	0	0	0
integral	1	0	0	0	0	0
equations	1	1	0	1	0	1
tractors	0	1	0	0	0	0
semi-groups	0	1	0	0	0	0
evolution	0	1	0	0	0	0
automatic	0	0	1	0	0	0
different	0	0	1	1	0	1
algorithms	0	0	1	0	1	0
theory	0	0	1	0	0	1
implementation	0	0	1	0	0	0
application	0	0	1	0	0	0
geometric	0	0	0	1	1	0
aspects	0	0	0	1	0	0
partial	0	0	0	1	0	0
ideals	0	0	0	0	1	0
varieties	0	0	0	0	1	0
introduction	0	0	0	0	1	0
computational	0	0	0	0	1	0
algebra	0	0	0	0	2	0
commutative	0	0	0	0	1	0
oscillation	0	0	0	0	0	1
neutral	0	0	0	0	0	1
delay	0	0	0	0	0	1

Approche deterministe: Latent Semantic Analysis

LSA

Nothing more than a **singular value decomposition (SVD)** of document-term matrix:

Find three matrices U , Σ and V so that: $X = U\Sigma V^t$



For example with 5 topics, 1000 documents and 1000 word vocabulary:

Original matrix: $1000 \times 1000 = 10^6$

LSA representation: $5 \times 1000 + 5 + 5 \times 1000 \sim 10^4$

-> 100 times less space!

[TruncatedSVD](#) de scikit-learn

Approche Probabiliste: Latent Dirichlet Allocation

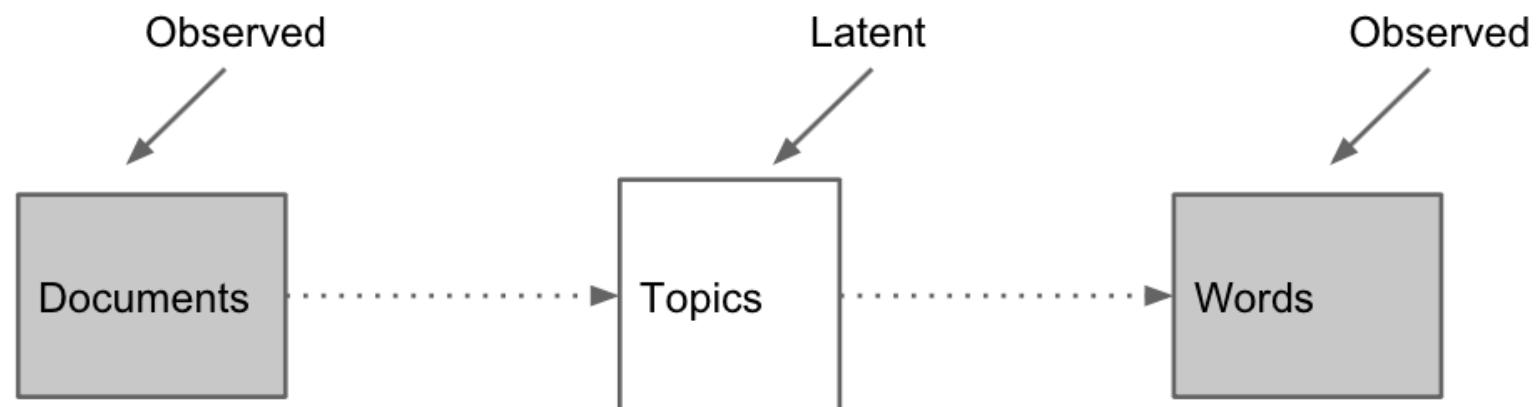
Blei 2002:

- Les topics sont distribués aléatoirement dans les documents
- Les mots sont distribués aléatoirement dans les topics
- Les topics et les mots sont répartis suivant une loi de Dirichlet

K: Nombre de topics

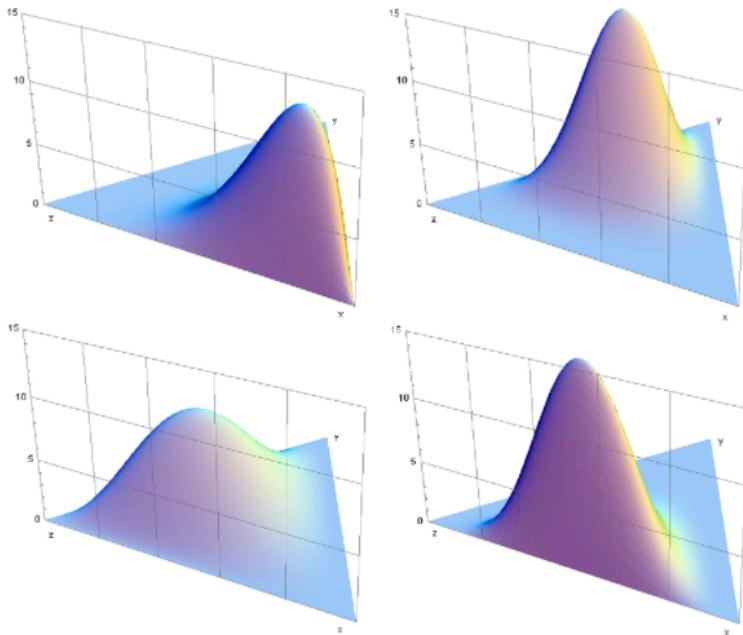
α : Nombre de topics par document

β : Nombre de mots par topic



Distribution de Dirichlet

Sorte de gaussienne généralisée



The Dirichlet Distribution

- ▶ Let $\Theta = \{\theta_1, \theta_2, \dots, \theta_m\}$
- ▶ We write:

$$\Theta \sim \text{Dirichlet}(\alpha_1, \alpha_2, \dots, \alpha_m)$$

$$P(\theta_1, \theta_2, \dots, \theta_m) = \frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \prod_{k=1}^m \theta_k^{\alpha_k - 1}$$

- ▶ Samples from the distribution lie in the $m-1$ dimensional probability simplex

Librairies

Python

- Gensim <https://radimrehurek.com/gensim/>
- LDA Python library: <https://pypi.python.org/pypi/lda>
- Scikit-learn: <http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.LatentDirichletAllocation.html>

R

- LDA, LSA
- Topicmodels
- STM package <http://structuraltopicmodel.com/>

Visualisation

- Gensim: LDAviz
- R STM: stmBrowser

Premier exemple de topic modeling

Le Corpus: 935 résumés d'articles provenant de IEEE et ArsTechnica

On connaît les rubriques, les sujets abordés, le dataset est consistant mais a besoin d'un peu de nettoyage

Constitution du corpus: Liste de 20 fils RSS des sites webs en question correspondant à 20 rubriques différentes

Par exemple:

- Energy: <http://spectrum.ieee.org/rss/blog/energywise/fulltext>
- Gaming: <http://feeds.arstechnica.com/arstechnica/gaming>

Ce Notebook jupyter python permet de constituer le corpus:

<https://github.com/alexperrier/upem-topic-modeling/blob/master/py/Creation%20du%20corpus%20articles%20tech.ipynb>

Corpus obtenu : <https://github.com/alexperrier/upem-topic-modeling/blob/master/data/techno.csv>

Le Script R suivant utilise le package [STM](#):

- https://github.com/alexperrier/upem-topic-modeling/blob/master/R/stm_techno.R
- www.structuraltopicmodel.com

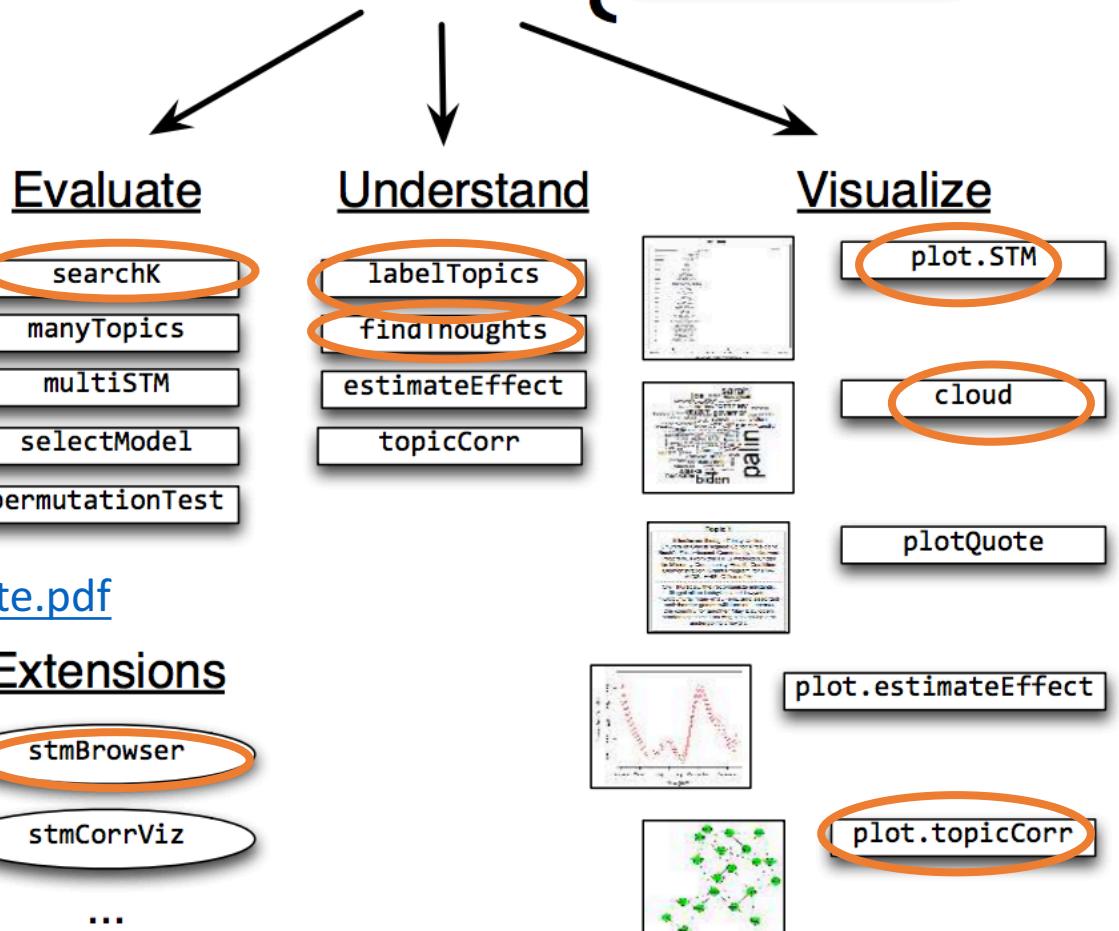
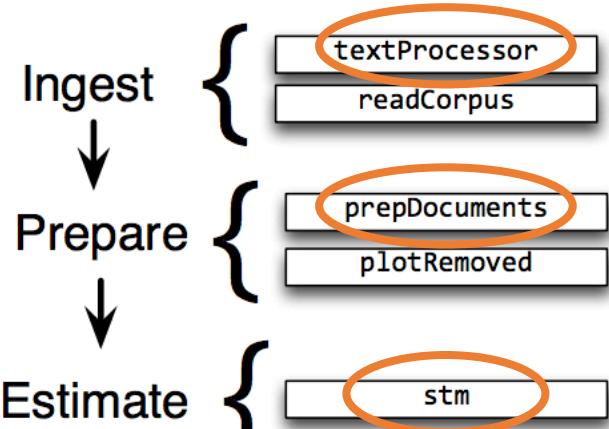
STM R package

Pourquoi ce package et pas un autre?

Il permet de:

- Déterminer simplement le nombre optimal de topics
- Analyser l'impact de variables externes sur les topics
- Préparer les textes (stem, token, ...)

Il possède de nombreuses méthodes d'exploration des résultats



- <http://www.margaretroberts.net/> et al
- <http://www.structuraltopicmodel.com/>
- <https://cran.r-project.org/web/packages/stm/vignettes/stmVignette.pdf>

Parametres de LDA

Le plus difficile est de trouver K le nombre de topics optimum

De nombreuses techniques et reponses

- Hierarchical Dirichlet process a la place de LDA https://en.wikipedia.org/wiki/Hierarchical_Dirichlet_process
- Clustering metrics
- 4 methodes: <https://cran.r-project.org/web/packages/ldatuning/vignettes/topics.html>

STM definit 2 métriques parlantes:

Semantic Coherence : *is maximized when the most probable words in a given topic frequently co-occur together attaining high semantic coherence was relatively easy by having a few topics dominated by very common words.*
A metric to summarize topics that combines term frequency and exclusivity to that topic into a univariate summary statistic referred to as FREX

FREX exclusivity: *The harmonic mean ensures that chosen terms are both frequent and exclusive, rather than simply an extreme on a single dimension.*

(vignette STM)

Rstudio

- [Rstudio](#)
- Run script ou Load environment
- Plot
- Visualize
- Explore
- Find stuffs!

Les packages

- `install.packages()`

Working directory

- `setwd()`
- `getwd()`



Lab 1: Topic Modeling sur IFFF - ArsTechnica

Le script suivant applique STM sur le corpus précédemment créé:

https://github.com/alexperrier/upem-topic-modeling/blob/master/R/stm_techno.R

- Lire et stocker les données dans une data frame: **read.csv**
- **textProcessor**: preprocessing du texte, transformations
 - removestopwords : filtrer les stopwords
 - removenumbers , removepunctuation : enlever les chiffres et signes de ponctuation
 - wordLengths : ne garder que les mots de n caractères ou plus
 - striptags : enlever les tags HTML,
 - stem : ne garder que la racine des mots (non utilisé)
- **prepDocuments**:
 - Création des structures requises pour STM,
 - Filtrage des mots trop ou peu fréquents
- Définir les variables externes: *rubrique et journal*
- **stm**: trouver les topics
- Analyser les résultats avec:
 - labelTopics
 - plot.STM
 - topicCorr
 - topicQuality
 - Cloud
 - stmBrowser
 - findThoughts
 - findTopic

Lab 1: Topic Modeling sur IEEE - ArsTechnica

STM trouve K par lui-même (mettre k =0 dans stm)

=> On obtient 63 topics

- Les topics sont de qualité, ils sont cohérents et compréhensibles
 - Mais peu de documents par topic, les topics sont maigres
 - Illustré par le wordcloud. Quelques mots très importants.

Une autre méthode existe pour trouver le nombre optimal de topics:

Grid search:

=> On fait varier K suivant une sequence (ex: 20 à 50)

⇒ On regarde Semantic coherence vs Exclusivity

⇒ Sur ce corpus, quand le nombre de topics croît:

⇒ Semantic coherence croit

⇒ Exclusivité décroît

⇒ Essayons aussi avec K = 20 topics



Lab 1: Interprétation des Résultats

Un Gridsearch donne 20 pour un nombre de topics favorisant la coherence semantique
Et ≈ 60 pour l'exclusivité

Pour le modèle K=20, on identifie bien les 20 topics suivants

La préparation avec textProcessor est efficace malgré quelques mots bizarres due au traitement de la ponctuation

Le corpus est: propre, sans faute d'orthographe, de longs paragraphes équilibrés, des sujets bien établis...

C'est trop beau pour durer

ANY
QUESTIONS



Partie III: L'alt right US sur un forum facebook



This is not Pepe, this is Alt-pepe

Corpus God Emperor Trump

Attention: le contenu de ce forum est souvent ouvertement raciste, anti semite, révisioniste, ...

<https://www.facebook.com/GodEmperorTrump/>

418790 posts et commentaires,

100+ Mb de donnees en csv

Tres bruité: orthographe défaillant, images, liens, abréviations, argot et expressions, ...

Example:

<https://www.facebook.com/GodEmperorTrump/photos/a.792821487511350.1073741828.791373644322801/1208866165906>

Dataset dispo sur: https://github.com/alexperrier/upem-topic-modeling/blob/master/data/main_fbget.csv.zip

Topic Most Frequent Words

STM sur échantillon

- 50.000 posts et commentaires,
- Entre 10 et 100 tokens
- meme pre-processing que précédemment

-> 26 topics apres grid search

Quelques topics sont cohérents:

- 14: mexico wall
- 8: fake news
- 24: bernie
- 18: islam

Mais les autres sont peu interprétables

Topic 6: get, need, head, job, away
Topic 11: look, looks, like, feel, looking
Topic 20: picture, read, comment, comments, ago
Topic 14: wall, pay, mexico, build, great
Topic 13: year, old, say, saying, said
Topic 3: donald, president, train, stop, maga
Topic 22: many, lives, wonder, matter, goes
Topic 23: fucking, stupid, every, dumb, piece
Topic 19: better, pretty, much, bit, knew
Topic 16: believe, change, fact, kind, mind
Topic 15: wait, thought, dead, sick, jeb
Topic 12: left, right, alt, alright, side
Topic 8: video, news, new, fake, times
Topic 24: vote, supporters, voted, bernie, sanders
Topic 1: bill, haha, lol, johnson, question
Topic 21: white, black, house, obama, racist
Topic 25: one, know, least, sorry, maybe
Topic 17: meme, real, magic, internet, memes
Topic 9: russia, putin, russian, live, million
Topic 10: god, emperor, thank, please, die
Topic 7: best, sent, supporter, folks, sad
Topic 18: islam, muslims, middle, religion, muslim
Topic 2: cuck, hell, jones, lmao, young
Topic 4: police, states, united, family, close
Topic 5: pence, rich, ben, mike, weak

STM sur échantillon: résultats

Les topics sont difficilement interprétables



Le nettoyage intense du corpus s'avère nécessaire

- Faire le tri entre le bruit et l'information
- Enlever les noms des copains taggés
- Enlever les 'stopwords': or, can, I, will, he, but, ...
- Booster le signifiant
- Réduire le bruit

Faire le tri entre le bruit et l'information

Noms de personnes:

- Info: Trump, Clinton, Bernie, Zuckerberg
- Bruit: tags de copains (britney smith, john doe, ...)

Argot

- Info: murica, dindu nuffin, guac bowl, cuck, (urbandictionnary.com tres utile)
- Bruit: lmfaooooo, lololololol, xdd, ...

Abreviations:

- Info: fb (facebook), swj (social justice warrior) , blm (black lives matter)
- Bruit: thatll (that will), shed (she would), I'm (I am)...

Stopwords: trouver le juste équilibre: enlever les mots trop fréquents sans toucher aux mots signifiants

Arsenal NLP de réduction de bruit en python



- La base: distribution anaconda, jupyter notebooks, pandas
- NLTK: tokenization et bien d'autres choses
- [Spacy.io](#): Part-of-speech tagging, Named entity recognition, tokenization, ...
- [Inflect](#): pluriels, singuliers, string transformations
- [Enchant](#): spellchecking library for Python
- Regex: pour réduire certaines strings *lmaooooo* => *lmao*
- textBlob, vader pour sentiment analysis
- Wordnet et Sentiwordnet: synonyme, proximité, signification

Pipeline pour FB-GET: les noms

Comment séparer les noms de personnes utiles (Trump, Clinton) des noms inutiles (Bob Smith, Joe Dalton, ...)?

Named Entity Recognition (NER) avec Spacy

Pour chaque post ou commentaire du forum, le NER va identifier:

- PERSON: nom de personnes
- GPE: pays, villes, états
- ORG: entreprises, marques
- NORP: Nationalités, religions, ...
- ...
- <https://spacy.io/docs/usage/entity-recognition#entity-types>

La première étape consiste donc à lister toutes les entités trouvées dans le corpus pour faire un tri:

Code: https://github.com/alexperrier/upem-topic-modeling/blob/master/py/ner_spacy_on_fbget.py

Résultats: 203708 persons, 41381 orgs, 17688 norp et 29410 GPE!

Le notebook est sur : https://github.com/alexperrier/upem-topic-modeling/blob/master/py/FB_GET_NER.ipynb

Pipeline pour FB-GET: les noms: creative stuff!

Voici toutes les variantes des noms pour Trump, Clinton et Zuckerberg trouvés en utilisant le NER de spacy.io

Donald Trump, Trump, God King, God Trump, God-Emperor, **EMPEROR TRUMP**, Emperor Palpatine, Emperor Trump,
Donald J. Trump, Donald J Trump, Drumpf, Baron Trump, The Emperor, Der Donald, Lord Emperor, **Führer Trump**,
#Trump, Adolf Trump, **Adolf Trumper**, Donaldus Magnus, **Trumpachu**, Trumpaloompas, Trumpamaniacs, Trumpanzee,
Trumpasaurus Rex, Trumpboner, Trumpborne, Trumpbot, Trumpcloaks, Trumpen Reich, Trumpenreich, Trumpenverse,
Trumpepe, Trumper, Trumpeteer, Trumpeter, **Trumpettes**, Trumpf, Trumpfucker, Triumph, **Trumpfalla**, Trumpidari,
Trumpinator, Trumpinreich, Trumpire, Trumpis, Trumpivia, Trumpkin, Trumpkins, **Trumplon**, Trumpman, Trumpmanía,
Trumpmas, **Trumpmeister**, Trumpmoji, Trumo, Trumpophiles, Trumpover, Trumpp, Trumps, Trumpsherd, Trumpster,
Trumpstika, Trumptrain, **Trumpulus**, Caesar, Trumpumvirate, Trumpus, DJT, Drumft, Drumpf

Shillary, Hillary Clinton, Crooked Hillary, Killary Clinton, Hillary, Clinton, Hilary, Killary, HRC, Killiry

Mark Zuckerberg, Zuckerberg, zucc, Mark Cuckerberg, Mark Fuckerberg, Cuckerberg, Cuckerbrg,
Suckerberg, Fuckerberg, Mark Zuck, Zuck

Pipeline pour FB-GET: Nettoyage

Etape 1: Commentaires et Entités

- Traduire les *abbréviations* qui font sens : *fb* => *facebook*, *ppl* => *people*, *blm* => *black lives matter*
- Résoudre les *abbréviations*: 'em => them, cant => cannot, hed => he would, i'm => I am, ...
- Unifier les *onomatopées*: lololol => lol, lmaooooo => lmao, yuuuuuge => yuge, hahahaha => haha, ahahahaha => haha
- Traiter les *apostrophes*: possessif et windows (Trump's => Trump, Trump's => Trump)

Puis:

- Etendre la liste des entités avec leur version plurielle: Cuban => Cubans,, Meme => Memes, Clinton => Clintons
 - Remplacer les variantes d'une entité (Trump, Trumper, DJT) par une racine commune (DonaldTrump)
 - Pour chaque thème: flagger les commentaires où le thème est mentionné
- => Pour chaque commentaire, remplacer les noms des personnes qui ne sont pas des entités par un token arbitraire: 'prsn'.

Résultat: des commentaires plus propres, avec un vocabulaire uniifié, plus simple et moins de variance

Pipeline pour FB-GET: bag of words, lemma, tokens

Etape 2:

Plus facile de travailler sur des paragraphes long que sur des phrases courtes

=> **Je groupe tous les commentaires par post**

- Enlever la ponctuation
- Transformer les mots en lemma (et tokeniser en meme temps)
- Enlever les stopwords (liste étendue)
- Enlever les mots qui ne sont pas de l'anglais (drastique mais la aussi liste etendue)

Si besoin

- Bigrams et skipgrams
- Limiter au verbes, noms et adjectifs

On obtient une liste de mots par post assez propre

Pipeline pour FB-GET: feature engineering et sentence

Etape 3:

On a maintenant une liste de tokens par post

STM attends des phrases

=> `' '.join(tokens)` pour retrouver une phrase que l'on puisse donner à STM

- **Feature extraction**, variable externes:
- **Popularité du post**: nombre de share, reaction, commentaires => klout
- **Duree de vie du post**: nombre de jours entre premier et dernier commentaire
- **Volume du post**: nombre de tokens
- **Themes**: grouper les entités par theme (america, Islam, Mexico, Europe, Clinton, Left wing media, ...)

Les scripts python associés à ce workflow sont:

- Dictionnaire de conversions des abbreviations, groupement par thème, regex et argot
 - <https://github.com/alexperrier/upem-topic-modeling/blob/master/py/invariants.py>
- Tagging des commentaires
 - https://github.com/alexperrier/upem-topic-modeling/blob/master/py/tag_entities.py
- Nettoyage, lemmatization et tokenization du corpus
 - https://github.com/alexperrier/upem-topic-modeling/blob/master/py/stm_prepare.py

Pipeline pour FB-GET: Themes vs Topics

1. Themes transversaux
 1. NER
 2. Filtrage manuel des noms de personnes et d'entités
 3. Definition de thèmes comme groupement de mots clés:
 - Trump, Clinton, Obama, Racism, main stream media, Islam, Judaism, France, europe, brexit,
 4. Grouper les entités par thème:
'Brexit' :['Brexit', 'Nigel Farage', 'Nigel Farrage', 'Farage', 'Farrage','Brexpats','UKIP'],
'left_orgs' :['ADL','ACLU','Panned Parenthood','Clinton Foundation','LGBT','LGBTQ'],
2. Tagger chaque commentaire avec les thèmes identifiés

On obtient une matrice *commentaires x themes*

Question sous jacente: est-ce que les topics varient en fonction d'un thème donné?

- Quels sont les topics quand Hillary, Brexit, Islam, ... sont mentionnés

Ce que l'on obtient en sortie du pipeline

- Pid
- Tags par theme
- Klout, days, log_klout, log_days, token_count
- Sent_tokens

Log_days = int(log(days + 1))

Prendre le log des variables externes permet de réduire
le nombre de valeurs prise par chaque variable
et ainsi de rendre plus rapide chaque itération

In [9]: df.head()

Out[9]:

	pid	alt_media	america	barackobama	brexit	christian	\		
0	1016462821813881	0	8	1	0	4			
1	1001240910002739	0	8	0	0	0			
2	1015848171875346	0	4	1	0	4			
3	990764697717027	1	6	0	0	1			
4	954656761327821	4	6	2	0	2			
	communism	democrats	donaldtrump	europe	\	war	klout	log_klout	\
0	0	0	5	0		0	8210.0	9.0	
1	0	4	27	0		0	6207.0	9.0	
2	0	7	14	0		0	9754.0	9.0	
3	0	1	12	1		1	2785.0	8.0	
4	1	0	23	1		0	3177.0	8.0	
	days	log_days	token_count	started_at	created_at	finished_at	sent_tokens		
0	23	3.0	900.0	2016-11-11 05:27:45	2016-11-11	2016-12-04 10:07:10	muslim exclude criticize woman bitch seem crit...		
1	2	1.0	900.0	2016-10-28 01:50:12	2016-10-28	2016-10-30 01:37:31	throne kid president king sign picture reckles...		
2	4	2.0	900.0	2016-11-10 16:50:04	2016-11-10	2016-11-14 10:23:30	ashleigh smith max helpful straight thank hear...		
3	2	1.0	900.0	2016-10-16 18:37:58	2016-10-16	2016-10-18 07:48:49	well shell run stand blubber contribute downfa...		
4	7	2.0	900.0	2016-09-04 00:09:22	2016-09-04	2016-09-11 16:19:25	lame suck glad sort pathetic respect creation ...		

Ce que l'on obtient en sortie du pipeline

Liste des tokens aggregés en phrase

```
'lame suck glad sort pathetic respect creation fun fact people behind centipede amazing video disturbed check folk
praise kek praise kek pepe donaldtrump supporter easily amuse yarn child proof bottle crayon busy hour heres story far rem
ember send notification facebook unpublished thankfully night reason add admin without fight issue send story follower bal
l roll message tweet send story breitbart tech reach contact lone conservative result massive attention issue markzuckerbe
rg praise kek image markzuckerberg news donaldtrump welcome alt right fight triggered safe space shame warn people overly
sensitive emotion possibly trigger bend white male racist sexist homophobic muslim hater white person include apparently b
elieve donaldtrump supporter blind people hillaryclinton supporter believe god donaldtrump spew baby birdie unreal gobble
suspension post video muslims attack caption religion peace month ago nobody respect usa barackobama hillaryclinton run ev
idence chinas airport welcome joke president dumb dumb markzuckerberg cuck eliza egyptians nazis quick turnaround free mil
oyiannopoulos lack try try page lmfao backfire three page bear lords power blessing upon dank steal flat flat donaldtrump
supporter markzuckerberg cuck shut page charge god donaldtrump feel charge donaldtrump super human post accident mysteriou
sly disappear evil emperor hillaryclinton execute order son law handle money hillaryclinton enjoy rare donaldtrump thank p
ower number talk google stuff rant people follow page god emperor crack milk water cause salty group people history god tu
rn nazi minion dang alexjones crew love smash liberal meme page bro dude right away nice matter content page matter event
transpire reasoning behind page political concept political preference biased corrupt bro stump donaldtrump praise kek ide
```

Retour dans Rstudio

- **Fichier:** fbget_stm_ready_full_01.csv
 - Fichier plus petit si probleme de mémoire: fbget_stm_ready_sample_01.csv
 - Entre 100 et 900 tokens par post
 - **Features:** themes, date debut et fin, days, reactions, shares, commentaires, klout
 - **Textes:** original *message*, *sent_tokens*, *display_message* (pour la visualisation)
 - Sans **pre-processing:** FALSE dans textProcessor
 - Filtrer par theme
-
- Gridsearch donne: 20 a 26 pour semantic coherence et > 55 pour exclusivité
 - K=0: donne 53 topics
 - On choisit 25 topics

En français

Beaucoup moins de choix dans les librairies

Lemmatizer:

- Lemmatiser est encore plus important qu'en anglais à cause entre autres des conjugaisons
- FrenchLefffLemmatizer <https://github.com/ClaudeCoulombe/FrenchLefffLemmatizer>

POS:

- Spacy a un nouveau modèle en français <https://github.com/explosion/spaCy/releases/tag/v1.8.2>
- Pattern , polyglot sont aussi des libraries python intéressantes

References

- STM: www.structuraltopicmodel.com , <http://www.margaretroberts.net/about/research/>
- [Followers sur twitter](#) (gensim, LDA, LSA, python)
- [Debats presidentiels américains](#) (stm, LDA, R)
- Tutorial sur Topic Modeling: <https://github.com/alexperrier/memewar/blob/master/docs/lTutorial-workshop.pdf>
- Topic modeling made just simple enough <https://tedunderwood.com/2012/04/07/topic-modeling-made-just-simple-enough/>
- A Review of Topic Modeling Projects <https://medium.com/@neala/a-review-of-topic-modeling-projects-941922a1216c>
- What can topic models of PMLA teach us about the history of literary scholarship?
<https://tedunderwood.com/category/methodology/topic-modeling/bayesian-topic-modeling/>
- Topic Modeling in the Humanities: An Overview <http://mith.umd.edu/topic-modeling-in-the-humanities-an-overview/>
- Topic Modeling for Humanists: A Guided Tour <http://www.scottbot.net/HIAL/index.html@p=19113.html>
- APPLYING DATA SCIENCE TO THE SUPREME COURT: TOPIC MODELING OVER TIME WITH NMF (AND A D3.JS BONUS)
<http://www.emilyinamillion.me/blog/2016/7/13/visualizing-supreme-court-topics-over-time>
- <http://scholar.princeton.edu/sites/default/files/bstewart/files/stm.pdf>
- A model of text for experimentation in the socialsciences
- “On Estimation and Selection for Topic Models.” In Proceedings of the15th International Conference on Artificial Intelligence and Statistics. <http://proceedings.mlr.press/v22/taddy12/taddy12.pdf>
- “Structural Topic Models for Open-Ended Survey Responses.” AmericanJournal of Political Science, 58(4), 1064–1082.Roberts ME, Stewart BM, Aioldi E (2016b).
- “A model of text for experimentation in thesocial sciences.” Journal of the American Statistical Association, 111(515), 988–1003.Roberts ME, Stewart BM, Tingley D, Aioldi EM (2013).
- “The Structural Topic Model andApplied Social Science.” Advances in Neural Information Processing Systems Workshop onTopic Models: Computation, Application, and Evaluation.
- LDA par Blei <http://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf>
- Video Topic Models par Blei: <https://www.youtube.com/watch?v=DDq3OVp9dNA>

Conclusion

- Peu de chance que le topic modeling marche du premier coup. Il faut s'immerger dans le corpus.
- Le pipeline de nettoyage est l'étape qui prends le plus de temps
- Mais c'est l'étape incontournable pour obtenir des résultats interprétables
- Le nettoyage dépend totalement du corpus initial
 - Sur les articles tech: peu de travail, bons résultats
 - Sur le forum FB, le nettoyage est incontournable

Ce que j'ai appris sur le corpus FB:

- Bien s'approprier le contenu: connaître le texte et le vocabulaire, NER, argot, ...
- Ne pas hésiter à réduire le corpus drastiquement
- Il n'y a pas de paramètre optimal pour obtenir tous les topics

Code, data et slides sur <https://github.com/alexperrier/upem-topic-modeling/>

Twitter: [@alexip](https://twitter.com/alexip)

alexis.perrier@gmail.com