

# Topic Modeling

NLP



# Le topic modeling, c'est quoi ?

Un topic model fait référence à un modèle probabiliste, définissant l'appartenance de documents à des thématiques précises : on identifie le ou les sujets d'un texte.

Par exemple, des livres sur Python ont une forte probabilité de contenir des termes en rapport avec le code et la data science, comme « statistiques », « package », « fonctions », etc à l'inverse d'un livre de Spinoza 🐱.



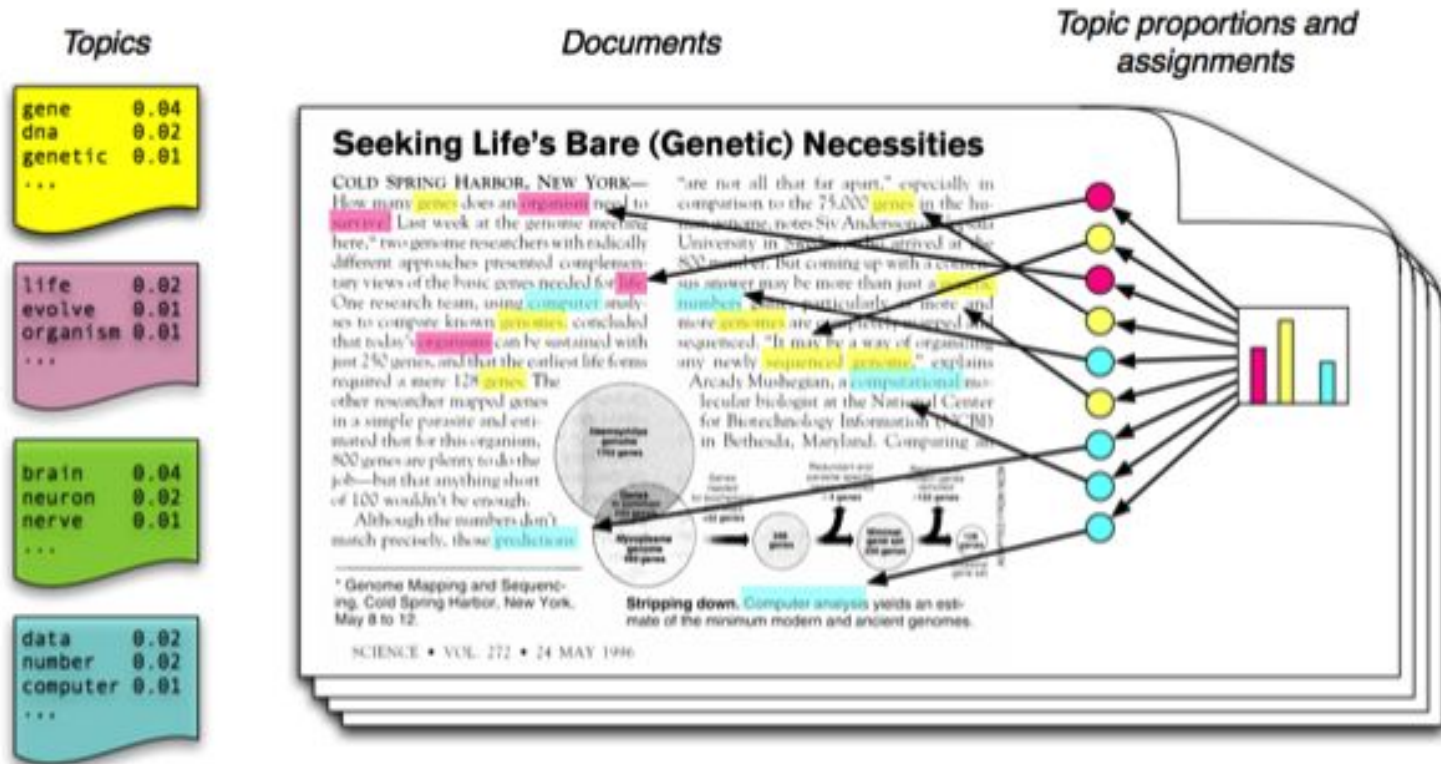
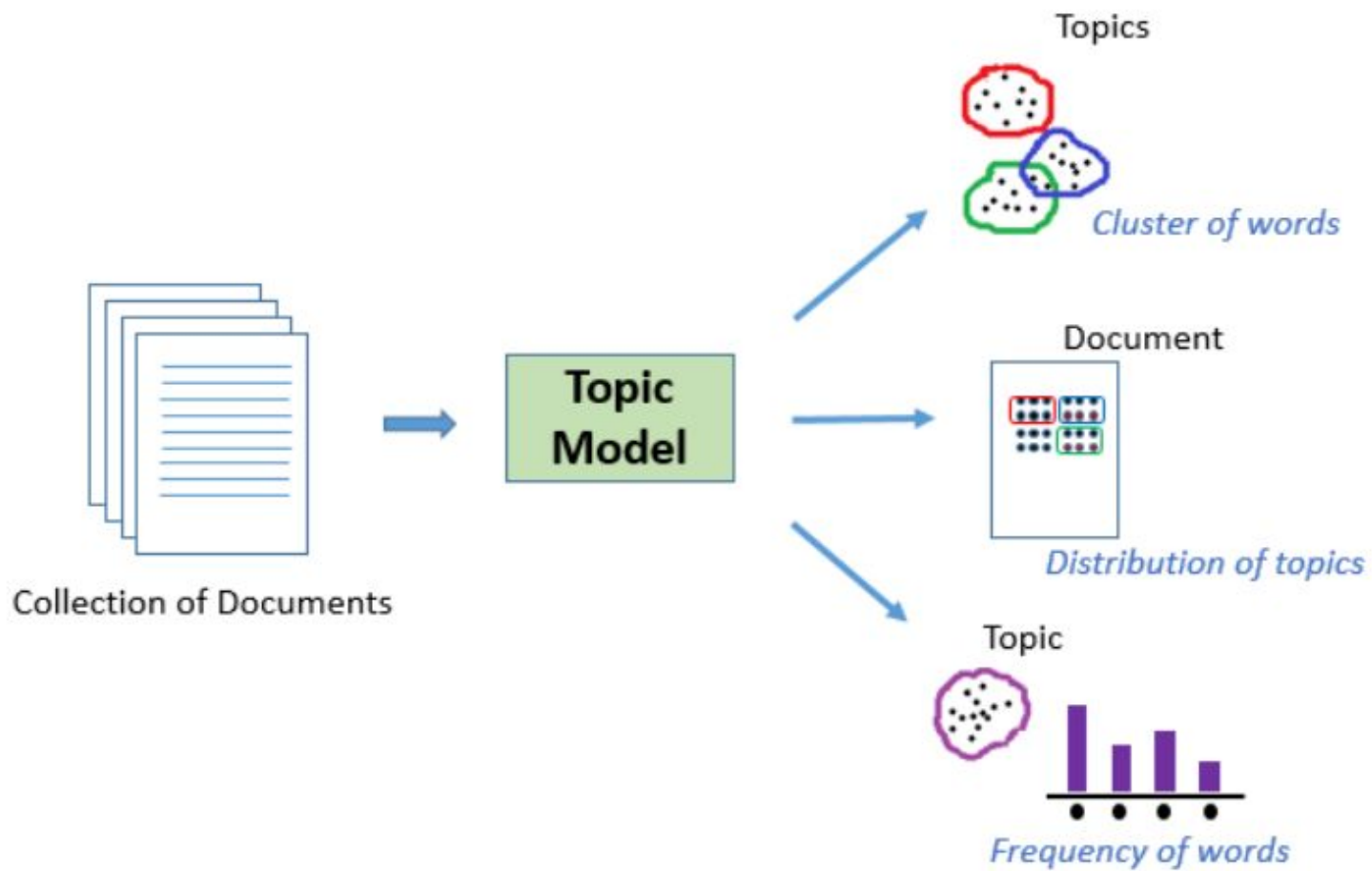


Figure source: Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM*, 55(4), 77-84.



# Comment ça marche ?

**01**

## Data

Récupération par  
scraping ou dans  
des datasets (avec  
ou sans aide de Regex)

**02**

## Traitement & Analyse

Tokenisations, lemmatisations,  
stopwords, TD-IDF, n-gramms, bag  
of words, corpus, ponctuations,  
espace vides extra

**03**

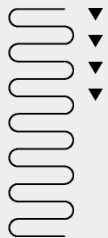
**04**

## Techniques

LDA  
Word embedding

# Notre modèle

LDA  
(non supervisé)



# Le dataset

Notre dataset contient des articles de recherche scientifique.

Nous devons utiliser l'ensemble de données pour prédire le sujet d'un article à l'aide de son résumé (colonne ABSTRACT).

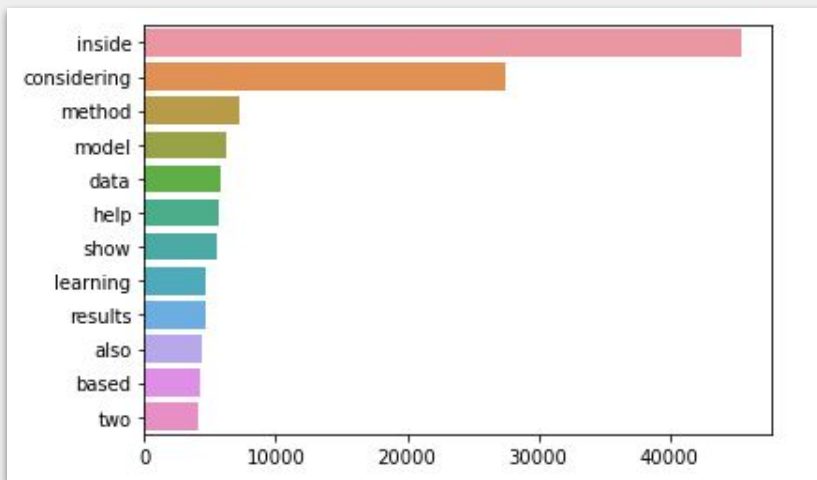
train.csv

	id	ABSTRACT	Computer Science	Mathematics	Physics	Statistics	Analysis of PDEs	Applications	Artificial Intelligence
0	1824	a ever-growing datasets inside observational a...	0	0	1	0	0	0	0
1	3094	we propose the framework considering optimal \$...	1	0	0	0	0	0	0
2	8463	nanostructures with open shell transition meta...	0	0	1	0	0	0	0

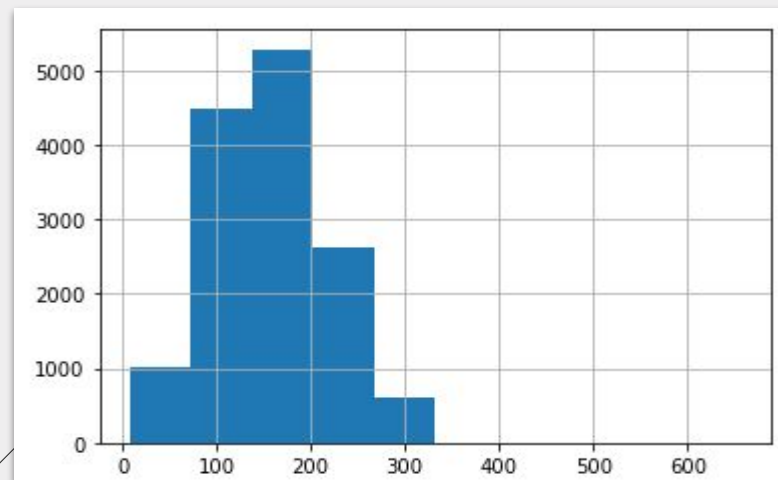


# Les étapes 1. Analyse

- **Exploration**, analyse et visualisation



Nombre de mots les plus fréquents  
hors stopwords



Nombre de mots par résumé  
(colonne ABSTRACT)





## Les étapes 2. Preprocessing



### Tokenisation, lemmatisation et suppression des stopwords

```
# tokenize / remove stopwords / lemmatize

def preprocess_news(df):
    corpus=[]
    stem=PorterStemmer()
    lem=WordNetLemmatizer()
    for news in df_train['ABSTRACT']:
        words=[w for w in word_tokenize(news) if (w not in stop)]

        words=[lem.lemmatize(w) for w in words if len(w)>2]

        corpus.append(words)
    return corpus

corpus=preprocess_news(df_train)
```

### Conversion du corpus en dictionnaire puis bag of words

```
# bag of words" avec gensim

dic=gensim.corpora.Dictionary(corpus)
bow_corpus = [dic.doc2bow(doc) for doc in corpus]
```

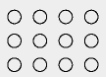
### Preprocessing avec les librairies

**NLTK et Gensim** > Identification des stopwords, création du corpus, tokenisation, lemmatisation et suppression des stopwords, conversion du corpus en dictionnaire (mappage des mots avec leur ID) puis bag of words (nombre de fois où le mot apparaît).



## Les étapes 3. Le modèle LDA/Multicore

Le modèle LDAMulticore utilise tous les cœurs de processeur pour accélérer la formation du modèle.



```
lda_model = gensim.models.LdaMulticore(bow_corpus,  
                                         num_topics = 25,  
                                         id2word = dic,  
                                         passes = 10,  
                                         workers = 2)
```

**bow\_corpus** : notre corpus transformé en bag of words

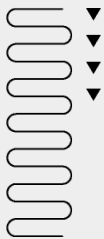
**num\_topics = 25** : nombre de sujets que l'on souhaite trouver

**id2word = dic** : déterminer la taille du vocabulaire, ainsi que pour le débogage

**passes = 10** : nombre de passages dans le corpus pendant l'entraînement.

**workers = 2** : propre au LDAMulticore pour optimiser le temps/performance





# Métrique

Afin de mesurer et analyser notre modèle, nous avons utilisé un métrique :

**Coherence score / Topic coherence** : Le score de cohérence thématique évaluent le degré de similitude sémantique entre les mots les mieux 'notés' dans le sujet.

```
coherence_model_lda = CoherenceModel(model=lda_model, texts=corpus, dictionary=dic, coherence='c_v')
```



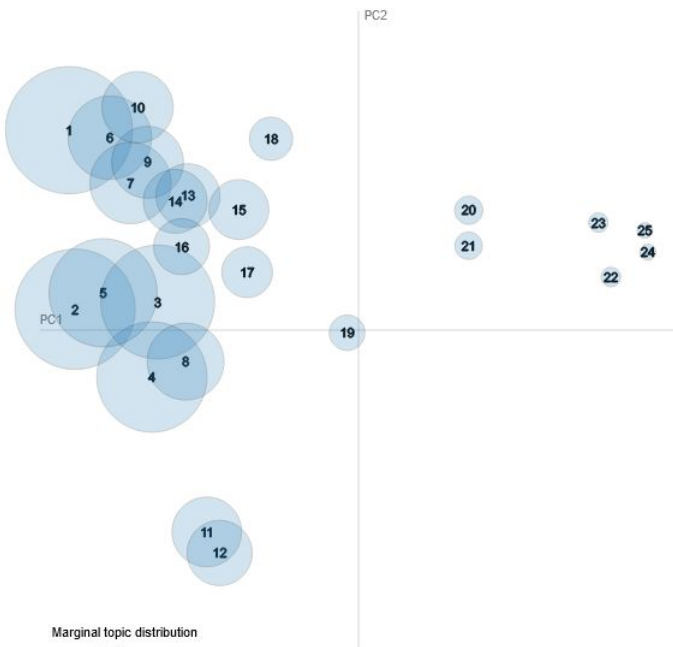
**Coherence score de 0.47 avec notre modèle non supervisé avec LDA**



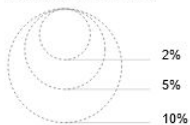
# Visualisation des termes fréquents

```
pyLDAvis.enable_notebook()
vis = gensimvis.prepare(lda_model, bow_corpus, dic)
vis
```

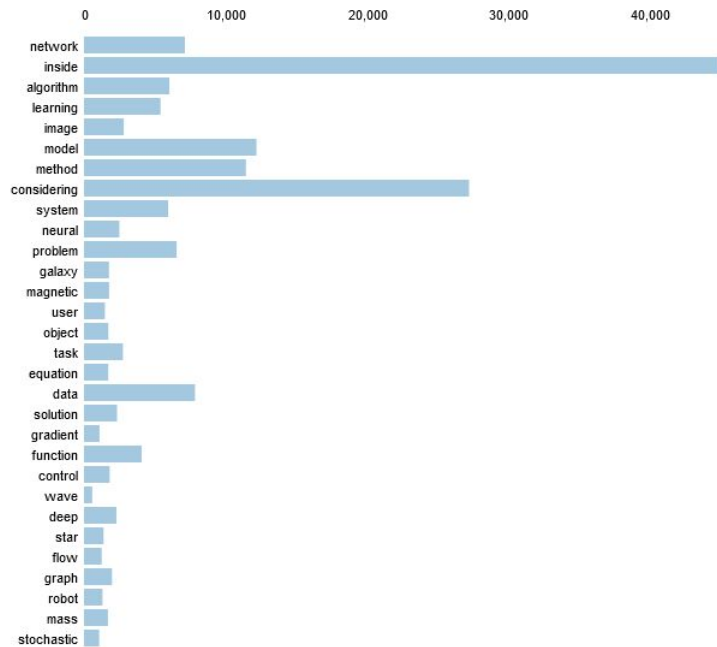
Intertopic Distance Map (via multidimensional scaling)



Marginal topic distribution



Top-30 Most Salient Terms<sup>1</sup>



Overall term frequency

Estimated term frequency within the selected topic

<sup>1</sup>  $salency(term, w) = frequency(w) * (\sum_{t \in topics} \log(p(t|w)/p(t)))$  for topics  $t$ ; see Chuang et al. (2012)

<sup>2</sup>  $relevance(term, w, topic, t) = \lambda * p(w|t) + (1 - \lambda) * p(w|t)/p(w)$ ; see Sievert & Shirley (2014)



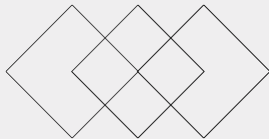
## Wordcloud





# Amélioration possible de notre modèle

Nous vous proposons de tester d'autres modèles grâce à des techniques et/ou algorithmes comme LDA2vec en deep learning.





## LDA2vec

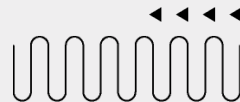
Algorithme “hybride” de deep learning créé, en 2016, par Christopher Moody. Il associe ainsi les algorithmes de LDA et word2vec, dans le but d’optimiser le topic modeling.

[Slideshow de Christopher Moody](#)

[Publication de Christopher Moody](#)

*Mixing Dirichlet Topic Models and Word Embeddings to Make lda2vec*

[Github de LDA2vec](#)





# Liens

[LDA avec Gensim](#)

[Modèle LSTM et Bert avec Pytorch](#)

[Word Embedding avec Gensim](#)

[Liste de 25 algorithmes de Word embedding](#)

