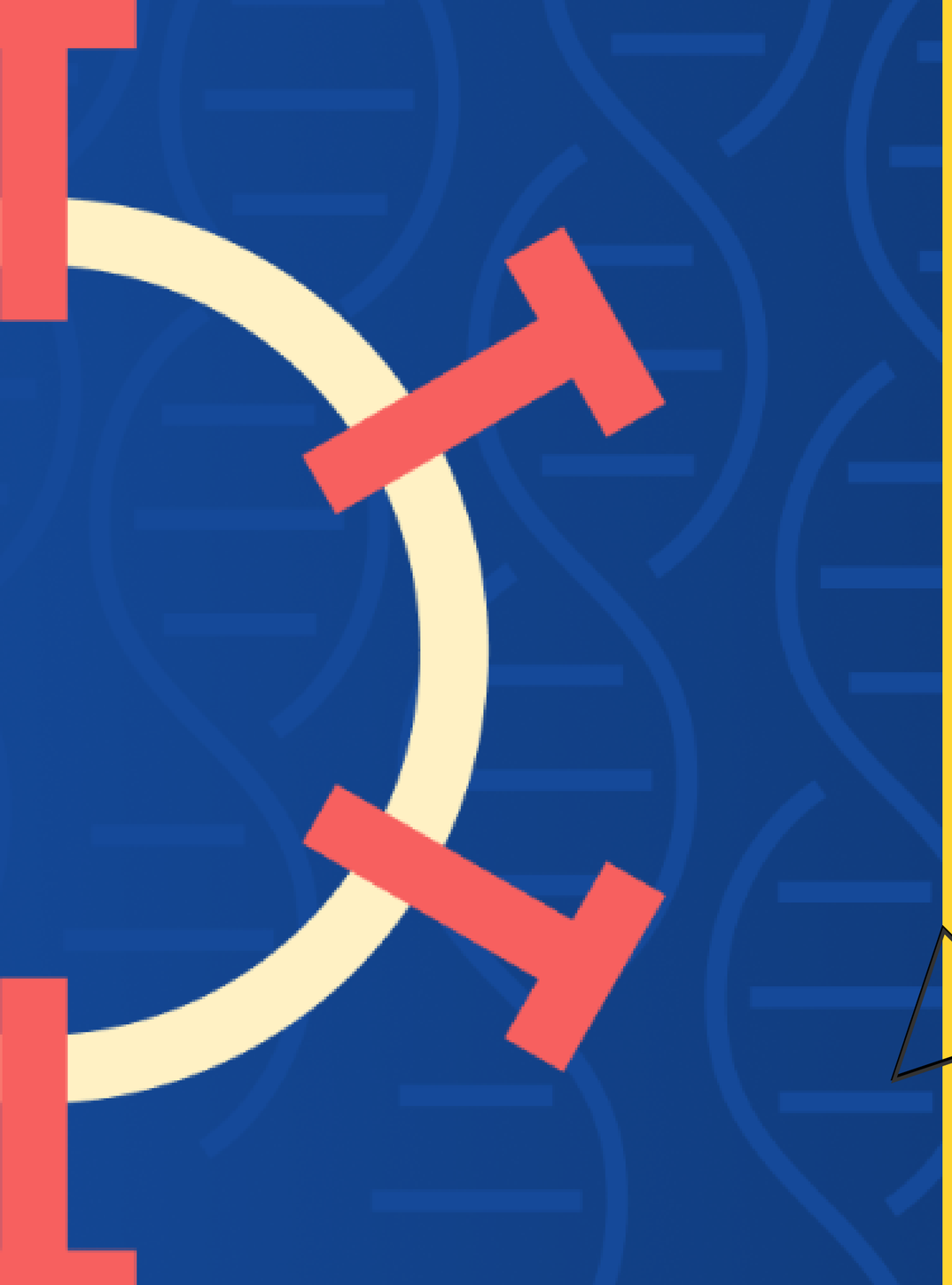
BIG DATA PROJECT

COVID SEARCH ENGINE

Luca Podo - Dario Aragona

Goal

REALIZE A SEARCH ENGINE SOLUTION FOR COVID-19 OPEN RESEARCH DATASET (CORD-19), A RESOURCE OF OVER 69,000 SCHOLARLY ARTICLES WITH FULL TEXT, ABOUT COVID-19, SARS-COV-2, AND RELATED CORONAVIRUSES USING TOPIC MODELING AND CLUSTERING TECHNIQUES



Before we begin

COVID-19 OPEN RESEARCH DATASET CHALLENGE

In response to the COVID-19 pandemic, the White House and a coalition of leading research groups have prepared the COVID-19 Open Research Dataset. A call to action to develop text and data mining tools that can help the medical community develop answers to high priority scientific questions.

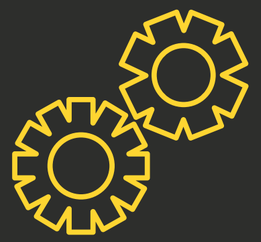
Based on this task we decided to use the dataset with full text to create a simple search engine that could help to get similar articles based on the keywords.

Link: https://www.kaggle.com/allen-institute-for-ai/CORD-19-research-challenge

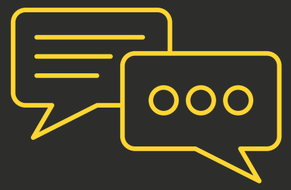
Solution proposed 1/2

DATASET SCHEMA

Dataset creation



Text filtering and processing



TF-IDF (CountVectorizer or Hashing)

Labeling (LDA or KMeans)

Classification (Naive or Logistic)

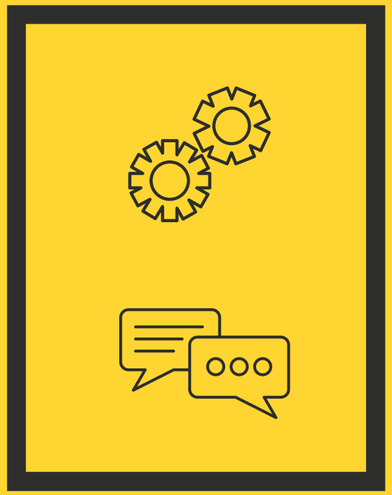
SearchBar

Solution proposed 2/2

SEARCHBAR SCHEMA

KEYWORDS

COVID EFFECTS ?



Words to Vector

Topic/Cluster Classification

Document filtering based on topic/cluster

GET BEST RESULTS USING COSINE SIMILARITY

DATASET

The dataset is structured in the following way:

PAPER_ID **TITLE** **ABSTRACT** **BODY_TEXT** **AUTHORS** **JOURNAL**

The analysis of each field told us that:

- **paper_id** is the only field to discriminate an article
- there are articles with same **title** but with different **body_text** (noisy titles)
- some **abstracts** are not provided, so we got null values (noisy body_text are few)
- **authors** and **journals** do not provide information we can use.

TEXT FILTERING / EN ARTICLES

Based on the previous analysis, we decided to select only **paper_id**, **title** and **body_text** fields, and delete noisy items (for example rows with duplicated body_text)

The next operation is to **detect language** of each document: this can be done thanks to the library ***langid*** applied on the body text.
Langid is an NPL library that performs language detection of a text, based on models of specific languages.

Once got languages for each article, we filtered them in order to keep only the **english** ones and saved them into a checkpoint csv, containing ca. 45000 documents.

Link: https://pypi.org/project/langid/1.1dev/

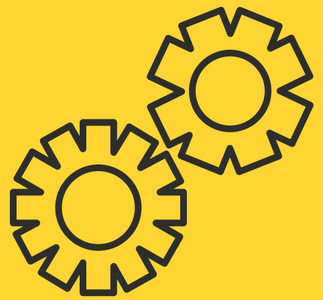
PROCESSING PIPELINE / TEXT

Text processing is performed only on the **body_text** field, which is the one containing more information.

We decided to adopt a standard text processing workflow due to the memory constraints of Google Colab.

The **pipeline** is the following:

- Case normalization
- Whitespaces filtering
- Punctuation filtering
- Tokenization
- Stopwords removal
- Stemming



TF-IDF

It is a method that allows us to evaluates how relevant a word is to a document in a document collection

It is calculated by two metrics:

- TF: **term frequency** of a word in a document
- IDF: **inverse document frequency**, how common or rare a word is in the entire document set

It is used for:

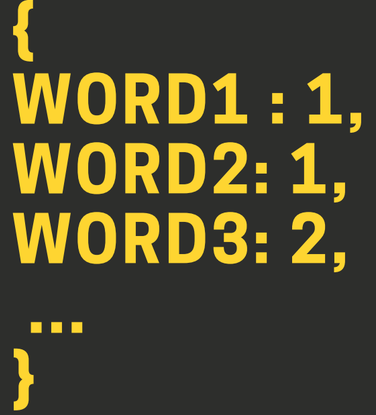
- **Information retrieval**
- **Keyword Extraction**

Link: https://monkeylearn.com/blog/what-is-tf-idf/

TF / COUNTVECTORIZER

It is a simple way to both tokenize a collection of text documents and build a vocabulary of known words

WORDS



```
{
WORD1 : 1,
WORD2: 1,
WORD3: 2,
...
}
```

WORD1 =
[1,0,0,0,0, ... , 0]

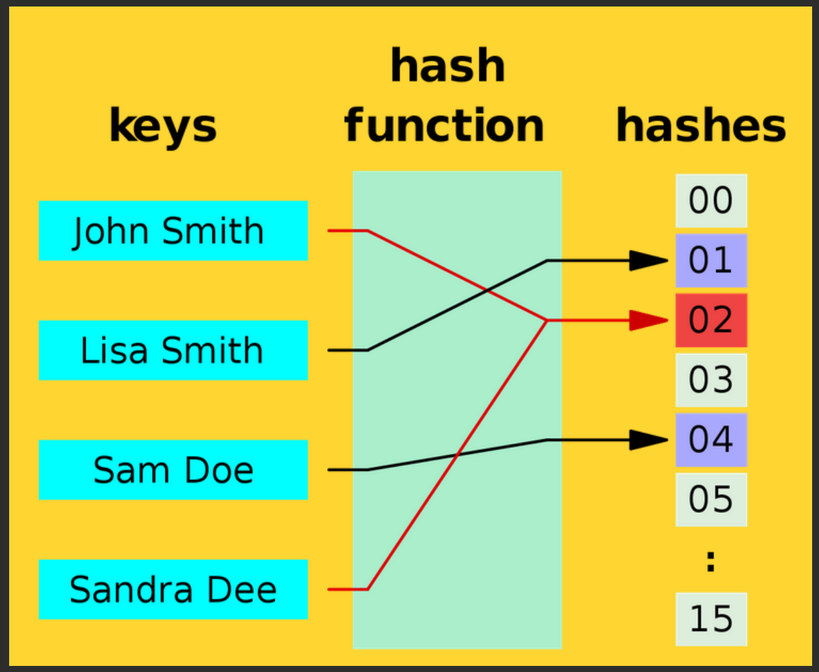
1. **Documents**

2. **Vocabulary and occurces**

3. **One-hot representation of a word**

TF / HASHING

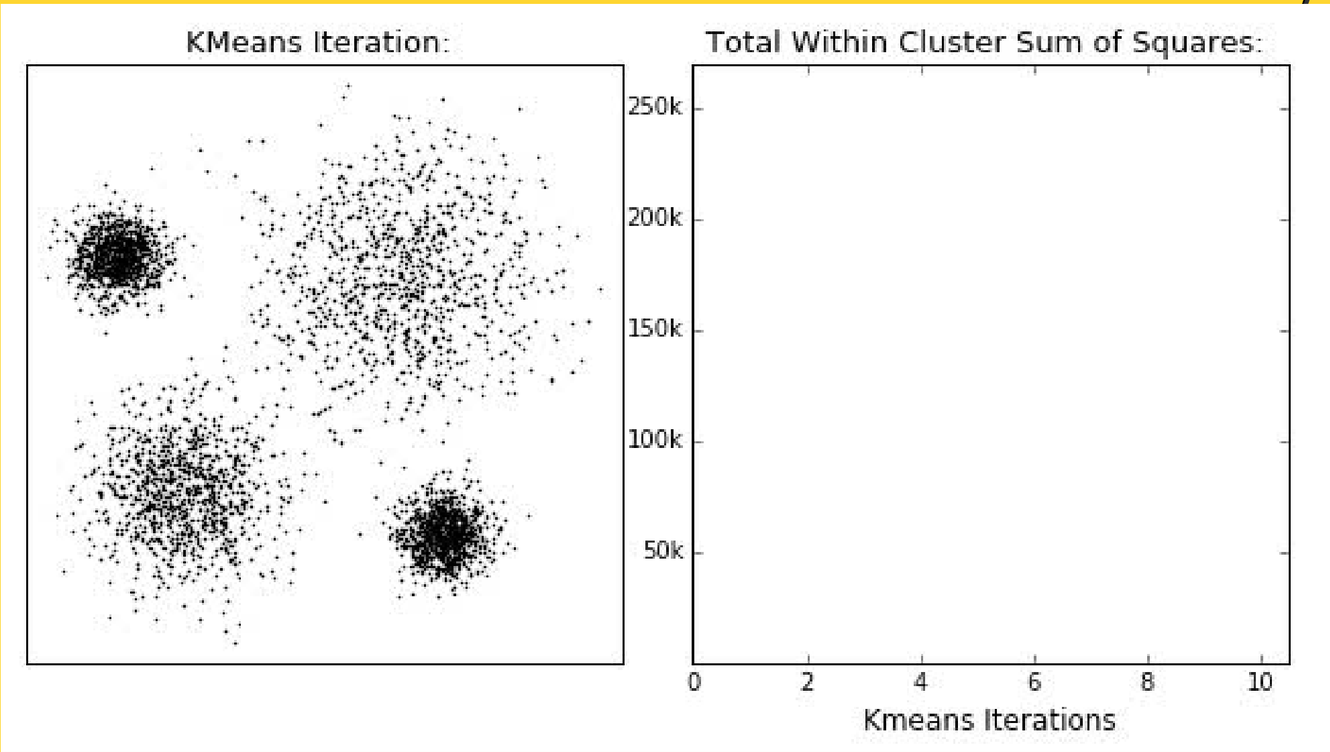
The hash function **maps** a raw feature into an **index** (term) in a **vector** or matrix. Then term frequencies are calculated based on the mapped indices. This approach avoids the need to compute a global term-to-index map, which can be expensive for a large corpus, but it suffers from potential hash collisions.



Link : https://spark.apache.org/docs/latest/mllib-feature-extraction.html#tf-idf

K-MEANS

K-Means is one of the most important **centroid**-based **clustering** algorithm.
To find the clusters, it solves an optimization problem:



"***find the k cluster centers and assign the objects to the nearest cluster center, such that the squared distances from the cluster are minimized***"

The optimization problem is NP-hard, so it looks for **approximate** solution using **Lloyd's algorithm**.

Link: https://dashee87.github.io/data%20science/general/Clustering-with-Scikit-with-GIFs/

K-MEANS/LLOYD'S ALGORITHM

Given an initial random set of k centroids, the algorithm proceeds by iterating two steps:

1. **Assignment step**: Assign each observation to the cluster with the nearest centroid (nearest wrt the used metric, for example Euclidean distance or cosine).

2. **Update step**: Recalculate centroids for observations assigned to each cluster.

The algorithm converges when the assignments no longer change, but it is not guaranteed to find the optimum because it may depend on the initial random centroids, and in some cases results can be bad compared to optimal clustering.

How to do better?

K-MEANS++

K-Means++ is an algorithm for choosing the initial values for the k-centroids in a smart way. The intuition behind this approach is that spreading out the k initial cluster centroids is a good thing.

1. Randomly select the first centroid from the data points.
2. For each data point compute its distance from the nearest, previously chosen centroid.
3. Select the next centroid from the data points such that the probability of choosing a point as centroid is directly proportional to its distance from the nearest, previously chosen centroid. (i.e. the point having maximum distance from the nearest centroid is most likely to be selected next as a centroid)
4. Repeat steps 2 and 3 untill k centroids have been sampled

K-MEANS MODEL

Train

TF-IDF VECTORS → KMEANS FITTING → KMEANS MODEL

Clusters assigment

TF-IDF VECTORS → KMEANS MODEL → CLUSTERS

Example

[0, 2.1, ... , 3.292] → KMEANS MODEL → CLUSTER 4

TOPIC MODELING

Topic modeling is an **unsupervised** ML technique for uncovering **hidden structure** in a collection of texts. It doesn't require a predefined list of tags or training data so it's a quick and easy way to analyze documents. This is also the main difference from Topic Classification.

Topic modeling example

topics of a set of customer reviews by detecting patterns and recurring words

TOPIC FOUND: FOOD

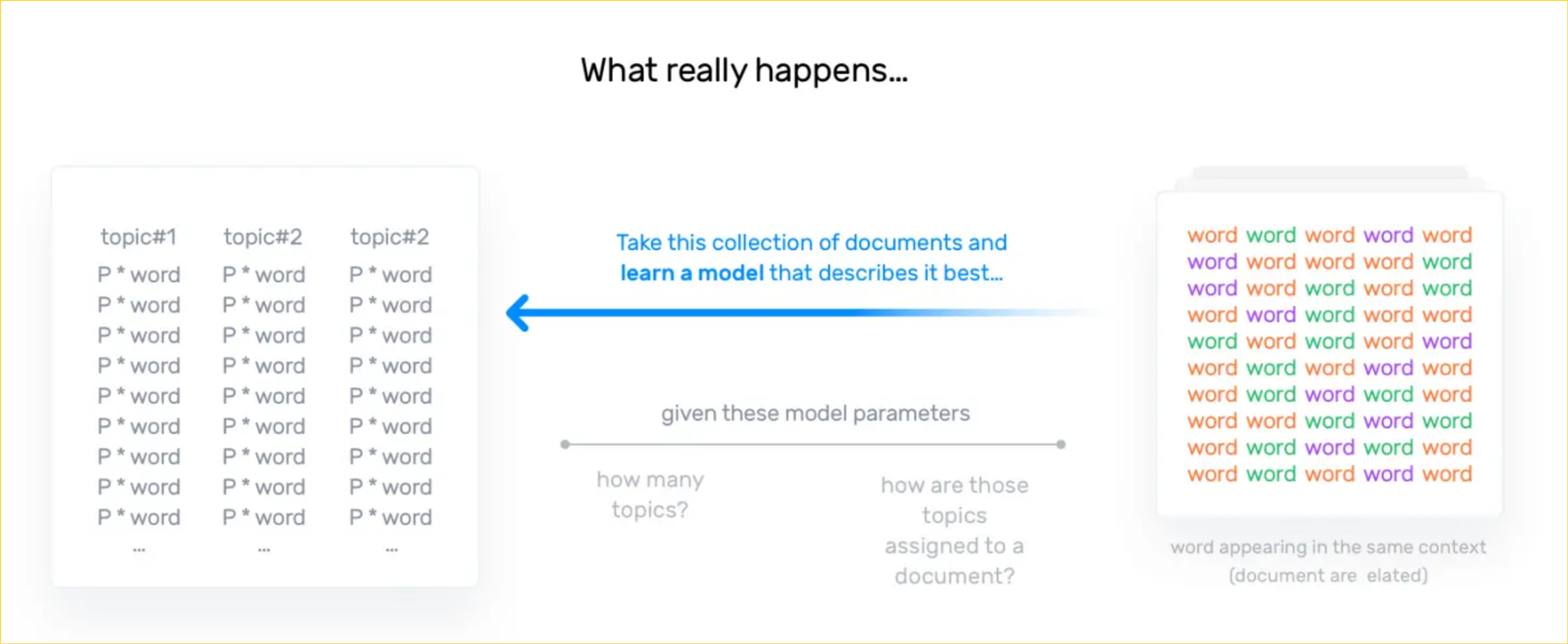
Topic classification example

topics of a set of customer reviews by predefined topic tags

TAG: FOOD

TOPIC MODELING / LDA

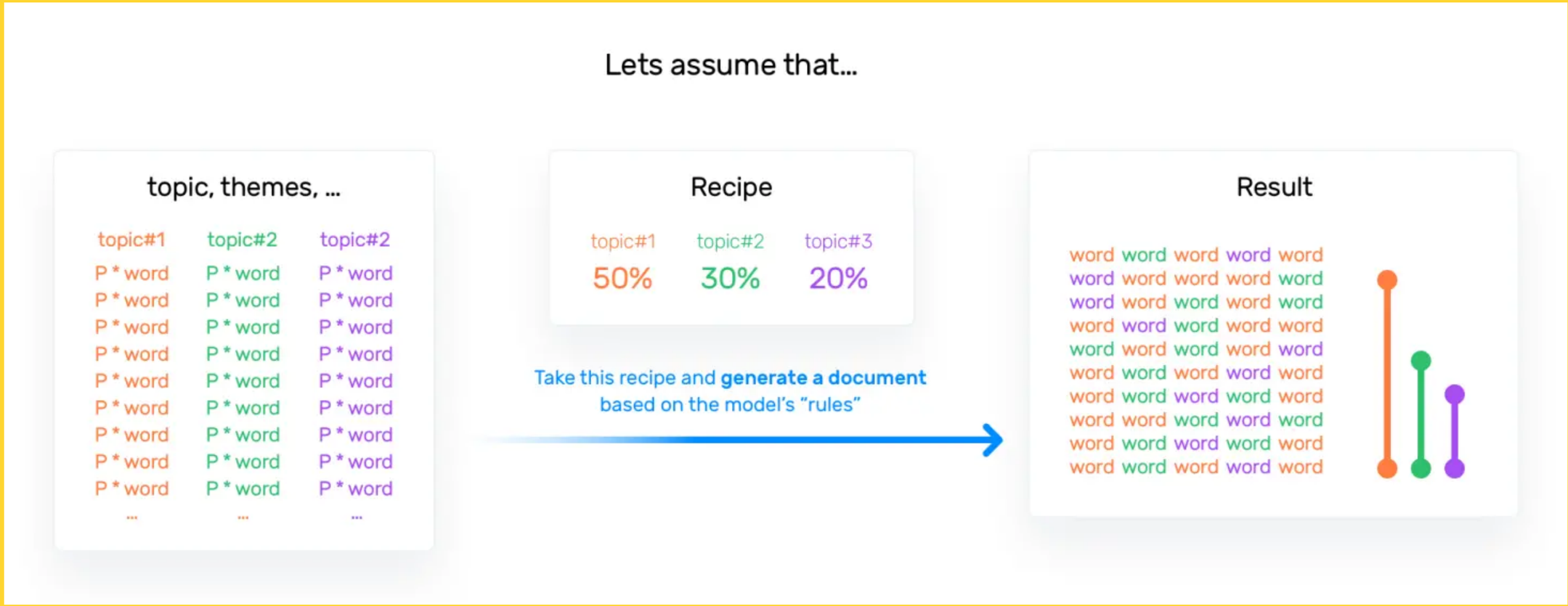
Latent Dirichlet Allocation (LDA) is one of the most frequent topic modeling methods. It is based on the distributional hypothesis: the **semantics** of two words will be similar if they tend to occur in similar **contexts**. LDA assumes that each **topic** is a mixture over an underlying set of words, and each **document** is a mixture over a set of topic probabilities.



Link: https://monkeylearn.com/blog/introduction-to-topic-modeling/#what-is-topic-modeling

TOPIC MODELING / LDA

Then it map back documents based on the topic distribution learned



TOPIC MODELING / LDA MODEL

Train

TF-IDF VECTORS → LDA FITTING → LDA MODEL

Topics assigment

TF-IDF VECTORS → LDA MODEL → TOPICS PROBABILITY DISTRIBUTION

Example

DOCUMENT → LDA MODEL → TOPIC_1: 50%
TOPIC_2: 10%
TOPIC_3: 40%

CLASSIFICATION / NAIVE BAYES

A Naive Bayes classifier is a **probabilistic**, **multiclass** classifier based on applying **Bayes' theorem** with strong (*naive*) independence assumptions between every pair of features.

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

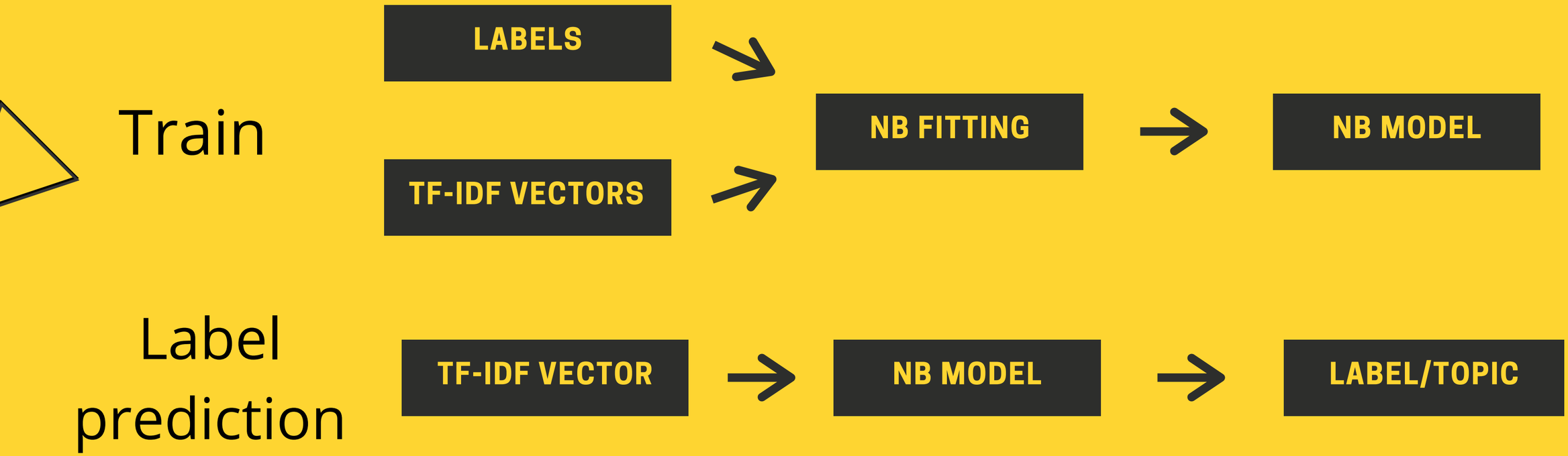
*"probability of observation belonging to class **y**, given **X** as its features"*

This model is typically used for **document classification**: within that context, each **observation** is a **document** and each **feature** represents a **term**.

A feature's value is the **frequency** of the term or a **zero** or **one** indicating whether the term was found in the document or not.

Link: https://spark.apache.org/docs/latest/ml-classification-regression.html#naive-bayes

CLASSIFICATION / NAIVE BAYES MODEL



We use this model to classify searched keywords among the clusters/topics and filter out documents in the corpus that do not belong to the predicted cluster

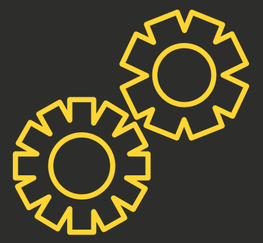
SEARCHBAR / PIPELINE - 1

To develop our search engine we start from a **form** in the notebook, used as search bar.

COVID EFFECTS ?

Text inserted is **filtered** and **processed** in the same way of articles of COVID19 corpus.

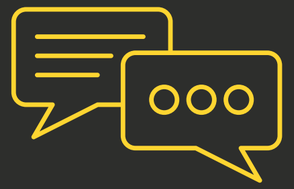
At this point we have tokens to pass to the TF-IDF function.



SEARCHBAR / PIPELINE - 2

Now we split the pipeline in 2 branches: one using **KMeans-labeled** corpus and the other using **LDA-labeled** one, in order to compare different solutions.

We load the corresponding **TF-IDF** models (with CountVectorizer for LDA, with Hashing for KMeans), and compute the TF-IDF vector representing the **searched** text.



The next step is to load the **classification models** (Naive-Bayes), already trained on the document corpus (one model for LDA-labels, one for KMeans), in order to **predict** the label/topic of the searched text.

SEARCHBAR / PIPELINE - 3

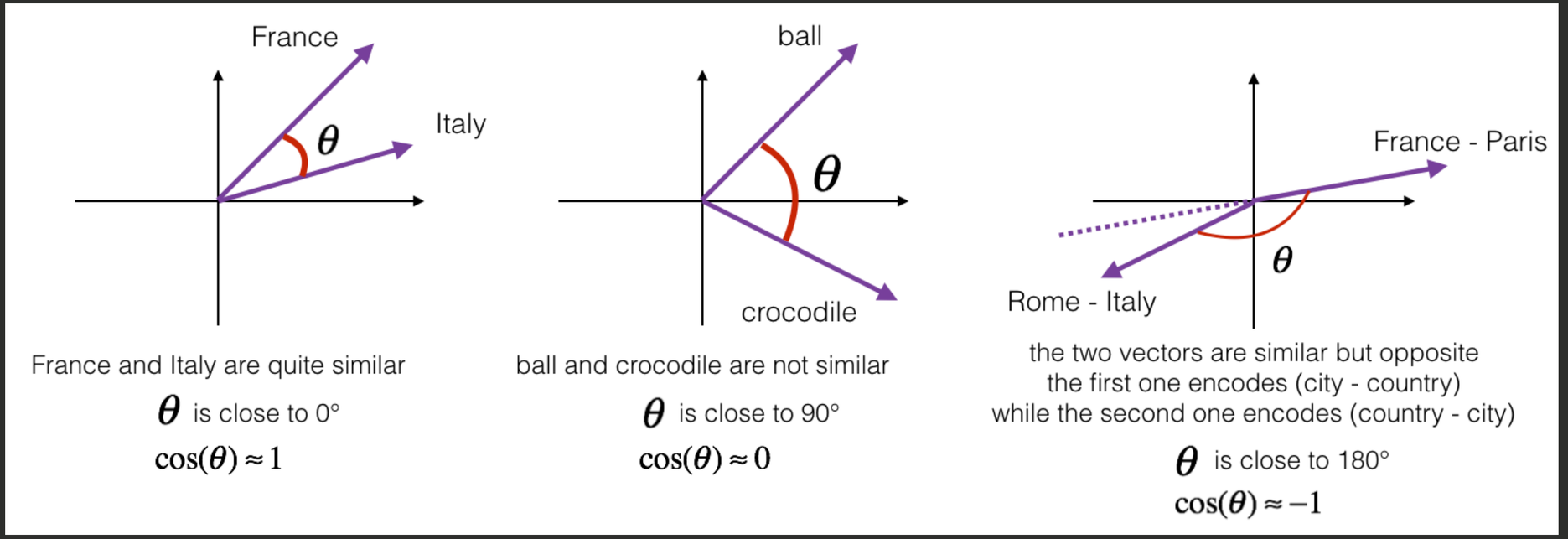
This operation returns the topic/cluster of the search, and we use this information to **filter** the document corpus belonging to that topic/cluster.

The last step is to compute the **cosine similarity** between the normalized TF-IDF vector of the research and the normalized TF-IDF vectors of all the filtered articles: this lets us get the most relevant papers.

SEARCHBAR / COSINE SIMILARITY

The similarity metric used is the **cosine similarity**, the most recommended metric when we talk about documents.

It is a measure of similarity between two non-zero vectors of an inner product space, and it is represented as the cosine of the **angle** between the vectors.



Link: https://en.wikipedia.org/wiki/Cosine_similarity

CONCLUSIONS / ISSUES

The main problems we faced during this project are:

DOCUMENTATION

Lack of documentation in some context. Spark provides two main libraries for machine learning methods: **mllib** and **ml**. The first one showed us some limitations in terms of methods and documentation. Based on this we decided to swap to ml. This library extends mllib and has more online support in terms of documentation and community resources.

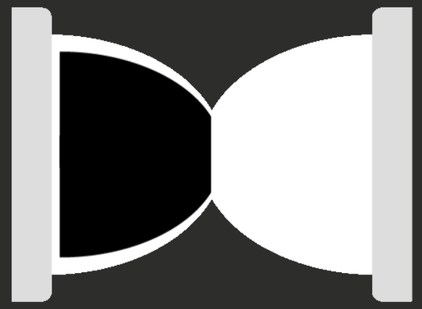
COLAB

Google Colab offers 12Gb of RAM, and sometimes it was not enough for the project. We upgraded it to get up to 25Gb, but also in this case sometimes the programs takes long time to execute and the RAM ended to be full. We noticed that by selecting the **TPU** accelerator we can get up to 35Gb. Now the RAM was enough, but due to the long time executions, sometimes Google blocked us from using it. We find out that sharing the file with a new Google account solves this problem.

CONCLUSIONS / ISSUES

TIME OF EXECUTION

During the project we faced very long time executions that blocked us to do further actions, and with RAM saturation we lost result of previous long calculations. To solve this problem we exported csv files at every step, so that we could restart from checkpoints instead of executing all previous code.



CONCLUSIONS / FINAL CONSIDERATIONS

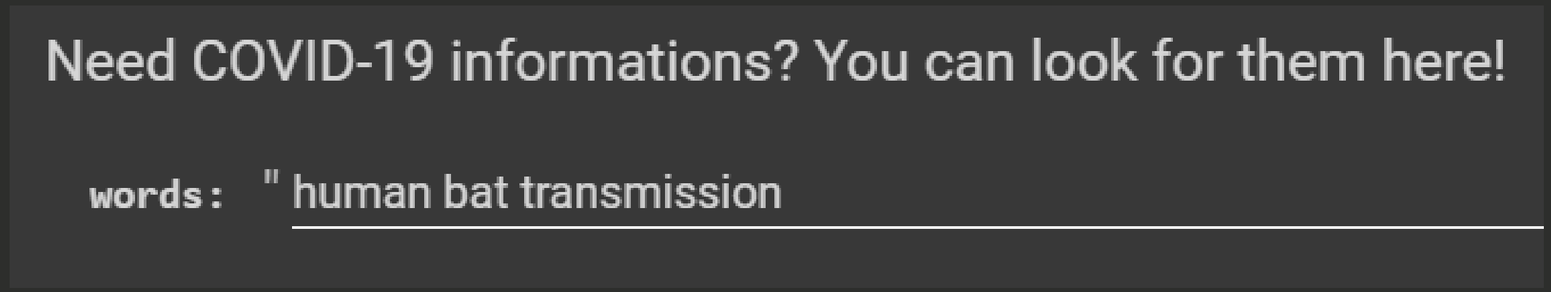
The results returned by the search bar using the two different labeling methods do not show relevant differences (example later).

Documents returned by the search engine are quite similar in both scenarios, and the only difference is in the order of the articles: in fact the cosine similarity of an article with respect to the same search text **changes** between KMeans and LDA labeled corpus.

With LDA we can access to the **vocabulary** of CountVectorizer model, extracted from the corpus; instead, with Hashing method we cannot (due to the not reversible hash function) , but the computations are **faster**.

CONCLUSIONS / EXAMPLE

Here there is an example of a research and the corresponding results:



CONCLUSIONS / EXAMPLE

LDA labeled results

title	dot
Education to Action: Improving Public Perception of Bats	0.8986313889698669
viruses Editorial Viruses and Bats	0.892625521547304
Emerging horizon for bat borne viral zoonoses	0.892326894497412
40 Bats	0.8863304816219524
Bat-man disease transmission: zoonotic pathogens from wildlife reservoirs to human populations	0.8826205441540544
Bat-Related Zoonoses	0.877095926652204
Diseases and Causes of Death in European Bats: Dynamics in Disease Susceptibility and Infection Rates	0.8738801131090413
Bats and zoonotic viruses: can we confidently link bats with emerging deadly viruses?	0.8705262109974642
PERSPECTIVE Bat and virus	0.8652800634695612
Bat pathogens hit the road: But which one?	0.86322661493389
Bats as a continuing source of emerging infections in humans	0.8626476697540687
Going to Bat(s) for Studies of Disease Tolerance	0.8622173210661633
Chiroptera (Bats)	0.8605533413482195
Mass extinctions, biodiversity and mitochondrial function: are bats 'special' as reservoirs for emerging viruses?	0.8603107114701792
viruses Can Bats Serve as Reservoirs for Arboviruses?	0.8527239517004146
Novel Insights Into Immune Systems of Bats	0.8507059608332418
Immunology of Bats and Their Viruses: Challenges and Opportunities	0.8425786714272065
viruses Bats and Coronaviruses	0.8420606271706514
viruses Immune System Modulation and Viral Persistence in Bats: Understanding Viral Spillover	0.839579276532594
Bats and Viruses: Friend or Foe?	0.8365561786351853

CONCLUSIONS / EXAMPLE

K-Means labeled results

title	dot
Education to Action: Improving Public Perception of Bats	0.8886982225203328
Emerging horizon for bat borne viral zoonoses	0.8802992504712933
Bats and Academics: How Do Scientists Perceive Their Object of Study? a11111 OPEN ACCESS	0.8715036412654669
Mass extinctions, biodiversity and mitochondrial function: are bats 'special' as reservoirs for emerging viruses?	0.858902664298934
viruses Bats and Coronaviruses	0.8566317155852312
Bat-man disease transmission: zoonotic pathogens from wildlife reservoirs to human populations	0.8551620000311129
Chiroptera (Bats)	0.8542021620090652
Bats and zoonotic viruses: can we confidently link bats with emerging deadly viruses?	0.8446692564603485
viruses Can Bats Serve as Reservoirs for Arboviruses?	0.8429247510405036
Bat pathogens hit the road: But which one?	0.839939941182609
Bat-Related Zoonoses	0.8372892929646382
Going to Bat(s) for Studies of Disease Tolerance	0.836903643893131
PERSPECTIVE Bat and virus	0.8330783694181746
Emerging Viruses: Coming in on a Wrinkled Wing and a Prayer	0.8323069799901054
Novel Insights Into Immune Systems of Bats	0.8311080644375766
viruses Immune System Modulation and Viral Persistence in Bats: Understanding Viral Spillover	0.8300012418505032
Bats and Viruses: Friend or Foe?	0.8297761187194955
viruses Editorial Viruses and Bats	0.829394237592699
40 Bats	0.8291445677661271
Bat consumption in Thailand	0.8282482274375553