TOPIC MODELING:

It is a process to automatically identify topics present in a text object and to derive hidden patterns exhibited by a text corpus. Thus, assisting better decision making.
It is an unsupervised approach used for finding and observing the bunch of words (called “topics”) in large clusters of texts.
Topics can be defined as “a repeating pattern of co-occurring terms in a corpus”. A good topic model should result in – “health”, “doctor”, “patient”, “hospital” for a topic – Healthcare, and “farm”, “crops”, “wheat” for a topic – “Farming”.

All topic models are based on the same basic assumption:

- each **document** consists of a mixture of ***topics***, and
- each ***topic*** consists of a collection of **words**.

topic models are built around the idea that the semantics of our document are actually being governed by some hidden, or “latent,” variables that we are not observing.

As a result, the goal of topic modeling is to uncover these latent variables — *topics* — that shape the meaning of our document and corpus.

Topic Models are very useful for the purpose for document clustering, organizing large blocks of textual data, information retrieval from unstructured text and feature selection

METHODS:

1.**LDA**: LATENT DRICHLET ALLOCATION

(LDA) is an example of topic model and is used to classify text in a document to a particular topic. It builds a topic per document model and words per topic model, modeled as Dirichlet distributions.

have a collection of documents which we want to cluster under different topics.

First document:

"France was the 2018 Football worldcup winners".

$\Downarrow$ After cleaning.

France, (2018), worldCup, football.

Define the no. of topics you want (n_topics)

n_topics = 3 (let). [3 topics {1, 2, 3}].

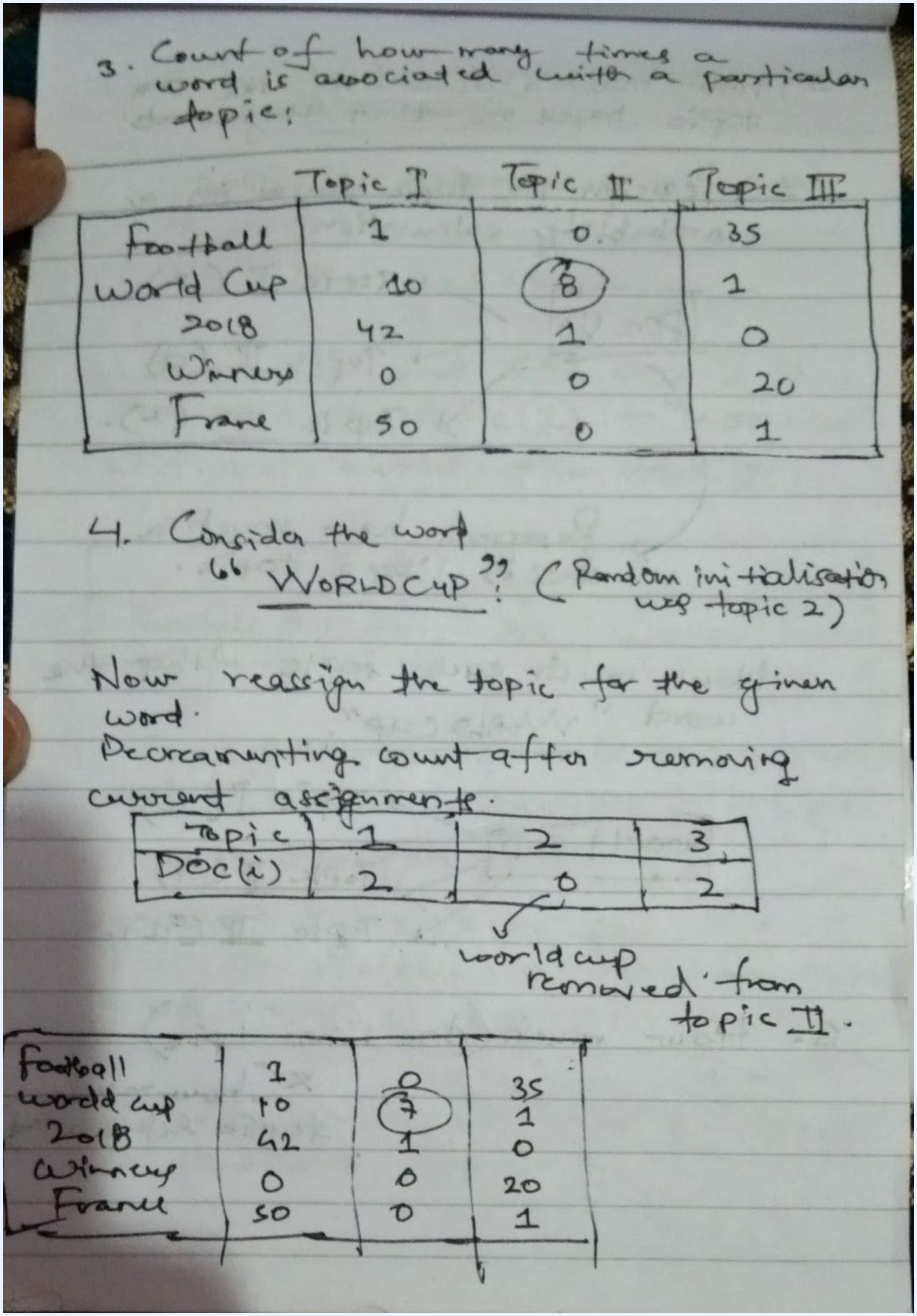
1. Now it assigns topic to each word in the corpus of documents provided randomly.

Doc(i):	Football	worldCup	france	2018	Winners
	3	2	1	1	3

$\hookrightarrow$ Random assignment of topics.

2. Calculate Document to topic count. In a given document how many times does.

Doc(i)	2	1	2
	Topic 1	Topic 2	Topic 3.



3. Count of how many times a word is associated with a particular topic:

	Topic I	Topic II	Topic III
Football	1	0	35
World Cup	10	$\boxed{8}$	1
2018	42	1	0
Winners	0	0	20
Frane	50	0	1

4. Consider the word "WORLDCUP"? (Random initialisation was topic 2)

Now reassign the topic for the given word.
Decreamenting count after removing current assignments.

Topic	1	2	3
Doc(i)	2	0	2

world cup removed from topic II.

football	1	0	35
world cup	10	$\boxed{7}$	1
2018	42	1	0
Winners	0	0	20
France	50	0	1

5. How much a document like one topic based on other assignments.

⟹ Reassign the topic based on a probability calculation.

$$\begin{tikzcd}
 & \text{Topic I (2)} \\
\boxed{\text{Doc (i)}} \arrow[ur] \arrow[r] \arrow[dr] \arrow[d, bend right] & \text{Topic II (0)} \\
\text{Represents how much a doc(i) likes a topic.} & \text{Topic III (2).}
\end{tikzcd}$$

How much each topic likes the word "WORLD CUP".

$$\begin{tikzcd}
 & \text{Topic I (10)} \\
\boxed{\text{world Cup}} \arrow[ur] \arrow[r] \arrow[dr] & \text{Topic II (7)} \\
 & \text{Topic III (1).}
\end{tikzcd}$$

6. How much (doc likes topic) × (how much topic likes word)

7. Calculate the Area enclosed for word "WORLDCUP":

Topic I : 2×10 = 20. (Max Area)

Topic II , 0×7 = 0

Topic III : 2×1 = 2

8. Next step what LDA does is it reassigns the topic (I) to "WORLD cup" which was initially given topic (II).

3	1	1	3	1
Football	World Cup	2018	Winners	France

9. This same procedure is done for each word in corpus of documents in one pass. This is one pass (Iteration).

Depending upon no. of pass it will keep on changing the topics for each word untill a convergence is acheived.

10. Finally when each word is assigned a particular topic.

We now look at a particular topic and see the probability of different words coming in that.

Topic I:

word I $\Rightarrow$ Prob. P_1
word II $\Rightarrow$ Prob. P_2
⋮
word n $\Rightarrow$ Prob P_n.

Representation of Topic I:

$(P_1)(\text{word I}) + (P_2)(\text{word II}) + (P_3)(\text{word III}) + \cdots$
$\cdots + (P_n)(\text{word } n)$.

$$\boxed{P_1 + P_2 + \cdots + P_n = 1}.$$

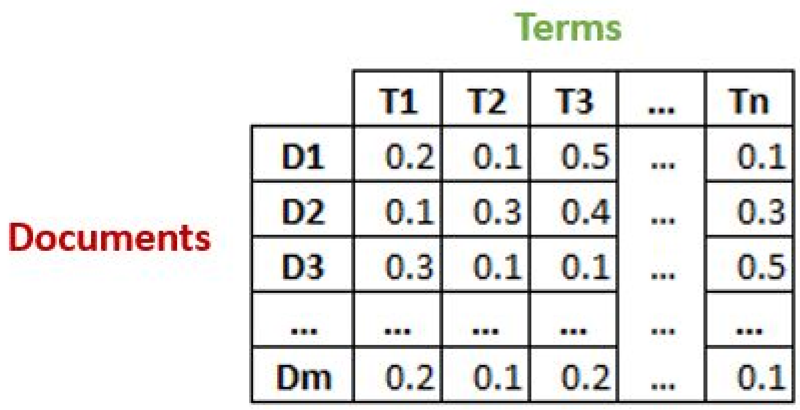
(*) The no. of word (n) that we need for each topic is also an input from our side.

So, it represents top (n) words with high Probability.

2.LSA:

The core idea is to take a matrix of what we have — documents and terms — and decompose it into a separate document-topic matrix and a topic-term matrix.
we can start thinking about our latent *topics*. Here's the thing: in all likelihood, *A* is very sparse, very noisy, and very redundant across its many dimensions. As a result, to find the few latent topics that capture the relationships among the words and documents, we want to perform dimensionality reduction on *A*.
This dimensionality reduction can be performed using **truncated SVD**. SVD, or singular value decomposition, is a technique in linear algebra that factorizes any matrix *M* into the product of 3 separate matrices: $M=U*S*V$, where *S* is a diagonal matrix of the singular values of *M*.
Let's say we have **m** number of text documents with **n** number of total unique terms (words). We wish to extract **k** topics from all the text data in the documents. The number of topics, k, has to be specified by the user.

Generate a document-term matrix of shape **m x n** having TF-IDF scores.



Terms

Documents

	T1	T2	T3	...	Tn
D1	0.2	0.1	0.5	...	0.1
D2	0.1	0.3	0.4	...	0.3
D3	0.3	0.1	0.1	...	0.5
...	...	...	...	...	...
Dm	0.2	0.1	0.2	...	0.1

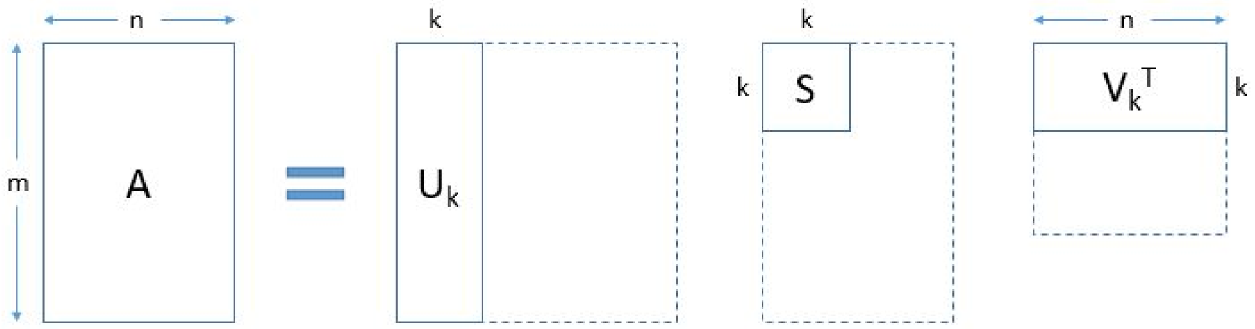
Then, we will reduce the dimensions of the above matrix to **k** (no. of desired topics) dimensions, using singular-value decomposition (SVD).

SVD decomposes a matrix into three other matrices. Suppose we want to decompose a matrix A using SVD. It will be decomposed into matrix U, matrix S, and VT (transpose of matrix V).

$$A = USV^T$$

Each row of the matrix **Uk (document-term matrix)** is the vector representation of the corresponding document. The length of these vectors is k, which is the number of desired topics. Vector representation for the terms in our data can be found in the matrix **Vk (term-topic matrix)**.

So, SVD gives us vectors for every document and term in our data. The length of each vector would be **k**. We can then use these vectors to find similar words and similar documents using the cosine similarity method.



With these document vectors and term vectors, we can now easily apply measures such as cosine similarity to evaluate:

- the similarity of different documents
- the similarity of different words

LSA is quick and efficient to use, but it does have a few primary drawbacks:

- lack of interpretable embeddings (we don't know what the topics are, and the components may be arbitrarily positive/negative)
- need for *really* large set of documents and vocabulary to get accurate results
- less efficient representation

- LSA involves SVD, which is computationally intensive and hard to update as new data comes up
- Since it is a linear model, it might not do well on datasets with non-linear dependencies.