(1) Objective of the project

We want to Recommended the product that user has is currently looking

↳ Why we care about search result.

→ estimated 35% increase in Revenue.

(1) Content based Recommendation.
(Text, Image description)

(11) Collborative Filtering

u1 → i1, i2, i3

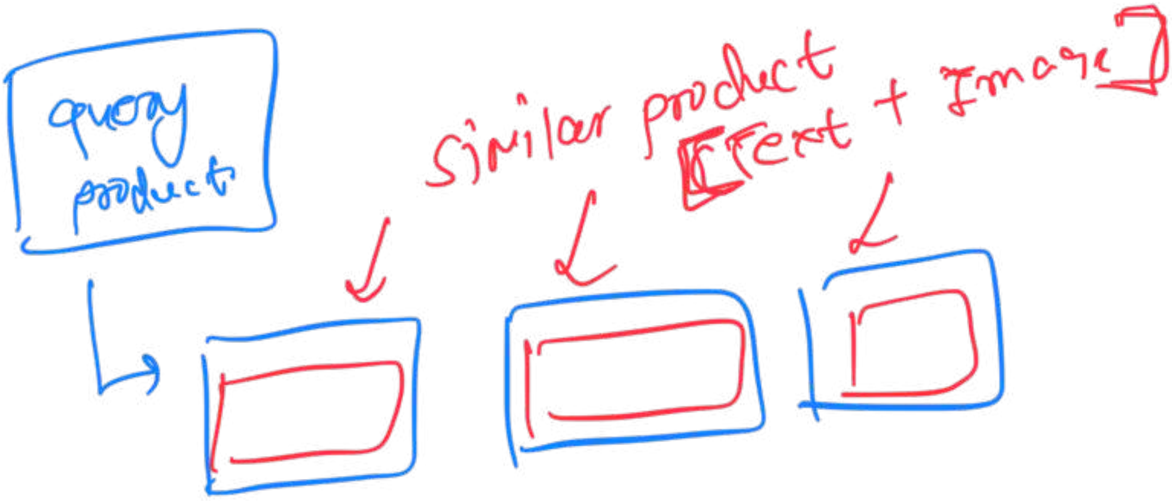
u2 → i1, i3, i4

↓ Recommended

u3 → i1, i3

this data is not easily available

For this project we will use Content based recommendation.



Plan of Attack

1. data acquisition
2. data cleaning
3. Text-processing
4. Linear Algebra.
5. Text based product Recommendation.
   - BOW
   - W2V
   - TF-IDF
6. Image based product recommendation
7. A/B Testing
8. Future work

(1) dataset Informati

1. Asin
2. brand
3. color
4. Image_url
5. title
6. price

(2) data cleaning

* Very important
overlooked

* Often

* $\boxed{\text{* basic EDA}}$

* Top products
* Top 10 Most frequent products

$\boxed{\text{you will get class imbalance information}}$

* Top 20 brands

Top 20 frequent brands

bad or messy

* Real world data is very

* Title
  - short
  - very informative
  - mostly present

best feature

* Remove duplicates.

* Find duplicates title

pandas dataframe. duplicated (Title)

ASIN:#1

title:- Red top

ASIN:#2

title:- Red top

only color difference

ASIN = #3

title - Blue top

* Remove products with short title

Text pre processing : Tokenization and STOP word Removal

Text preprocessing

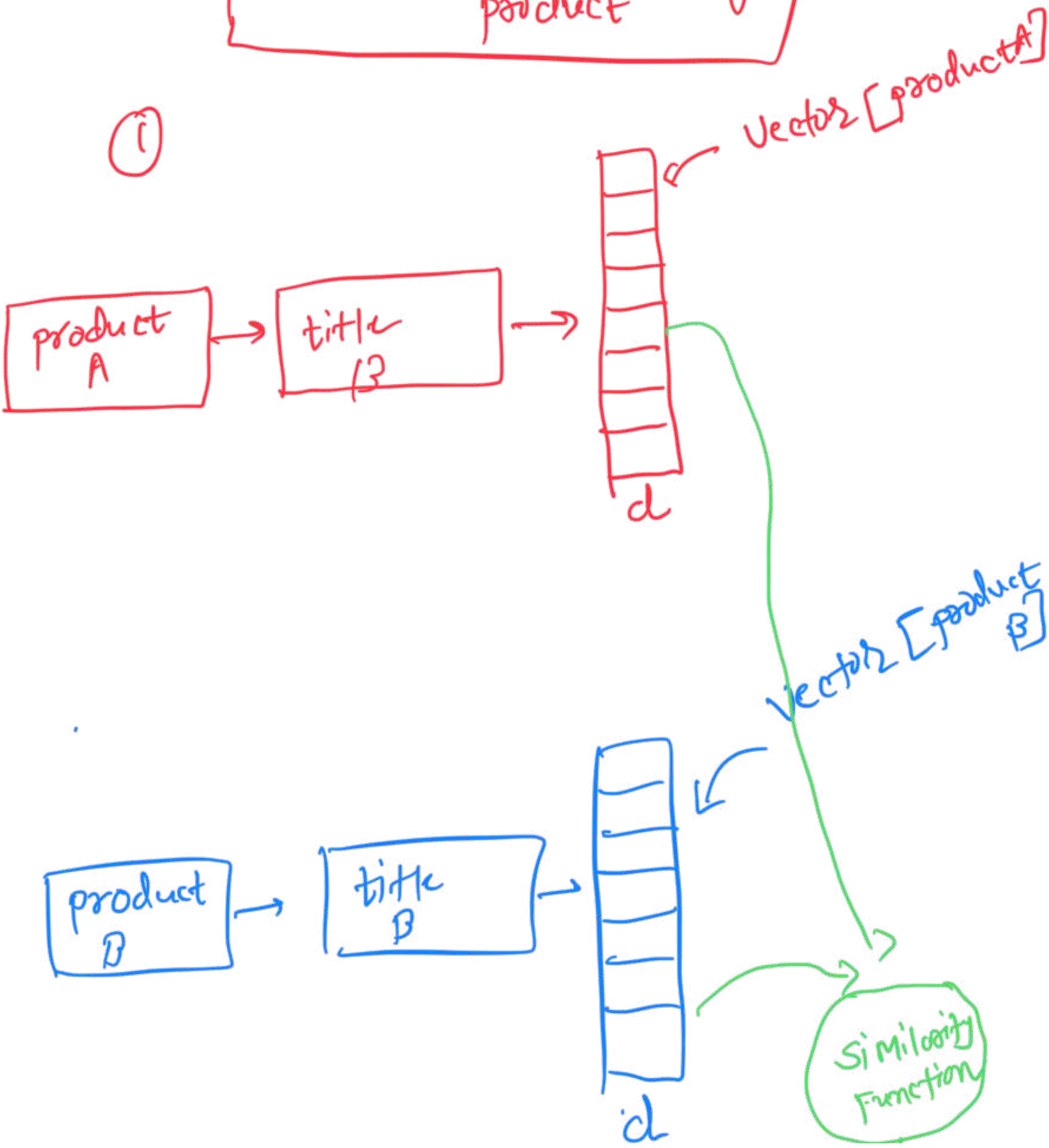
(i) STop word Removal

(ii) Stemming [expieriment and check if it works]

First cut approach

(i) Text based similarity product

(i)



similarity score

① Simple BOW Vectors.

T1, T2, T3 ..... Tn (Title)

① build vocabulary set [S]

S = all the words in all the title.

$S \in \mathbb{R}^d$

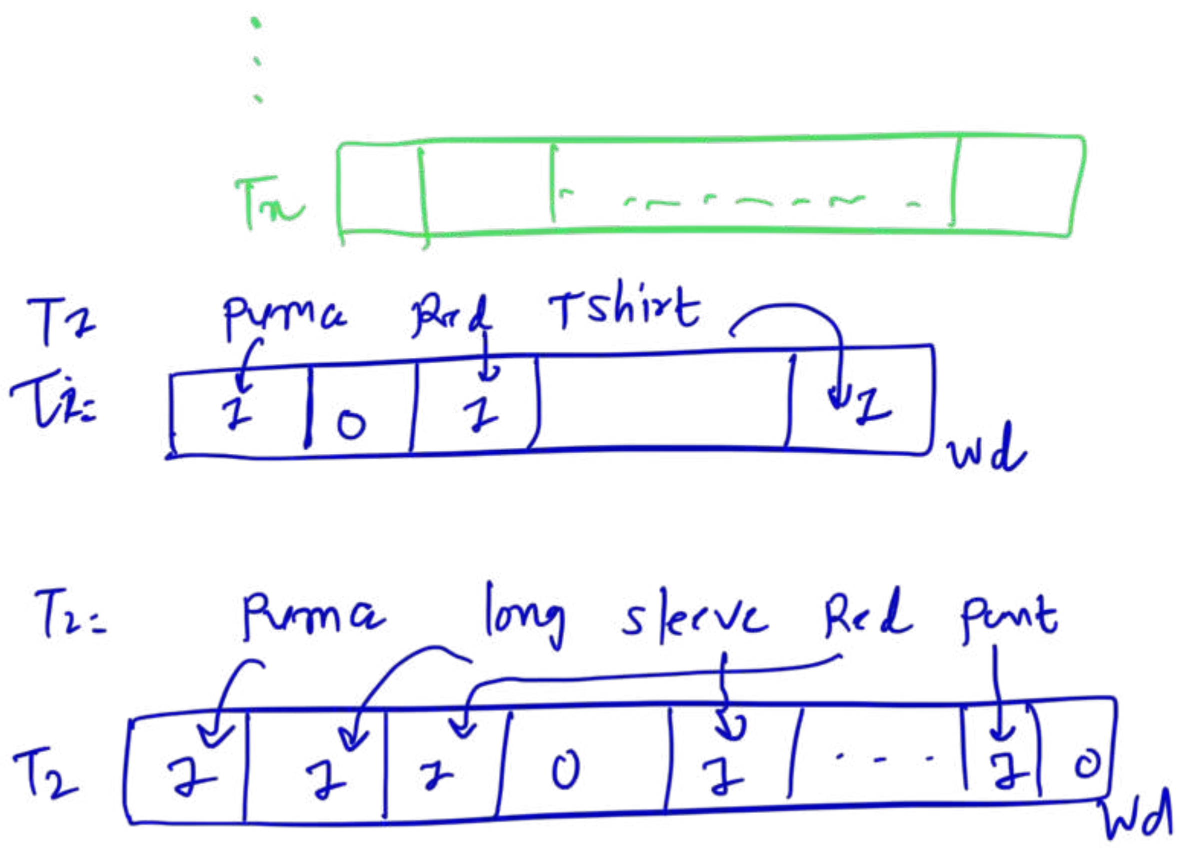
W1	W2	....	Wd

d is very large

② Generate vectors using BOW.

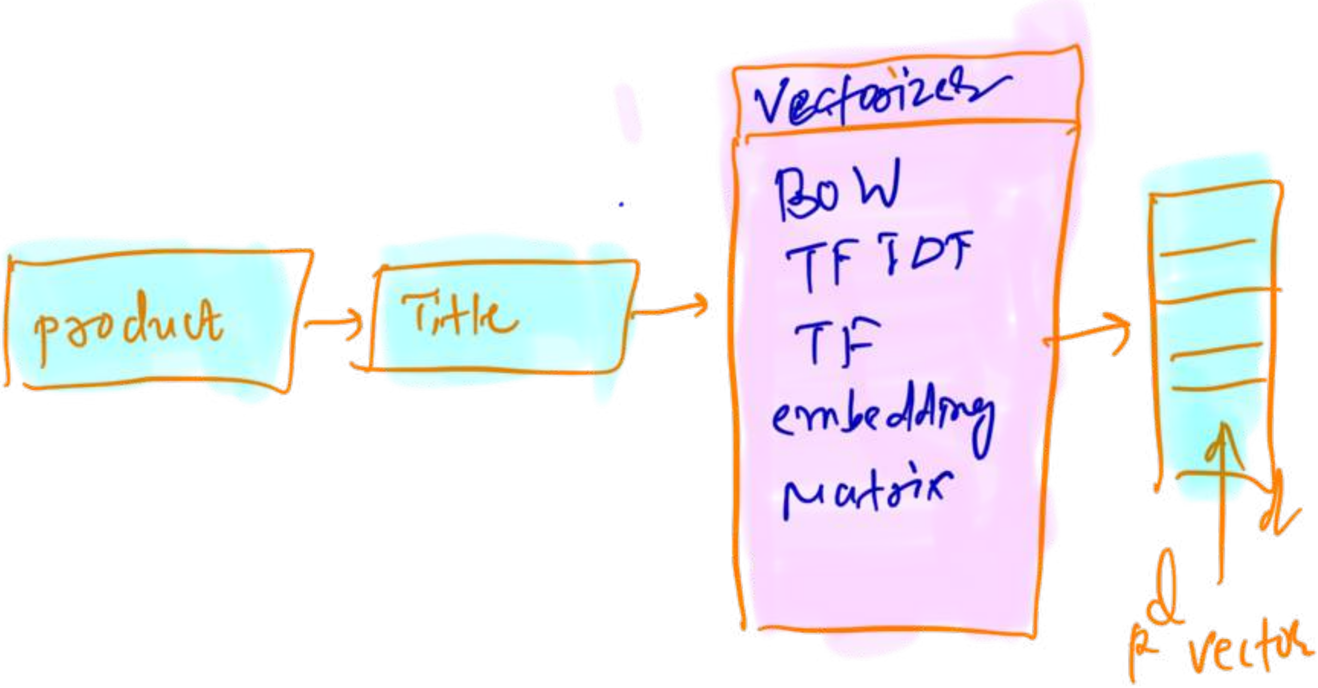
T_1	W1	W2	W3	Wd
T_2	W2	W1	W4	Wd

⋮



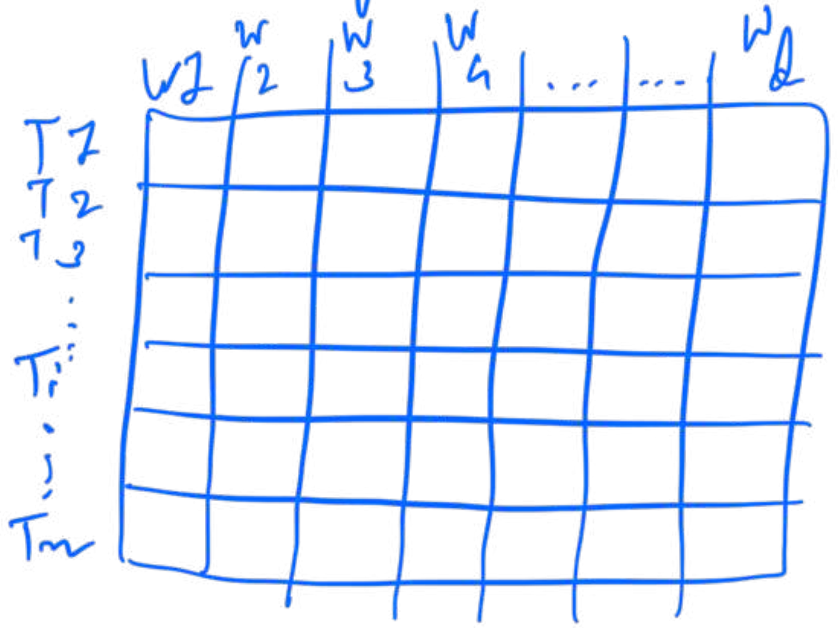
Problem

1. sequence information
2. counter



1. BoW → vectors

↳ very sparse vector

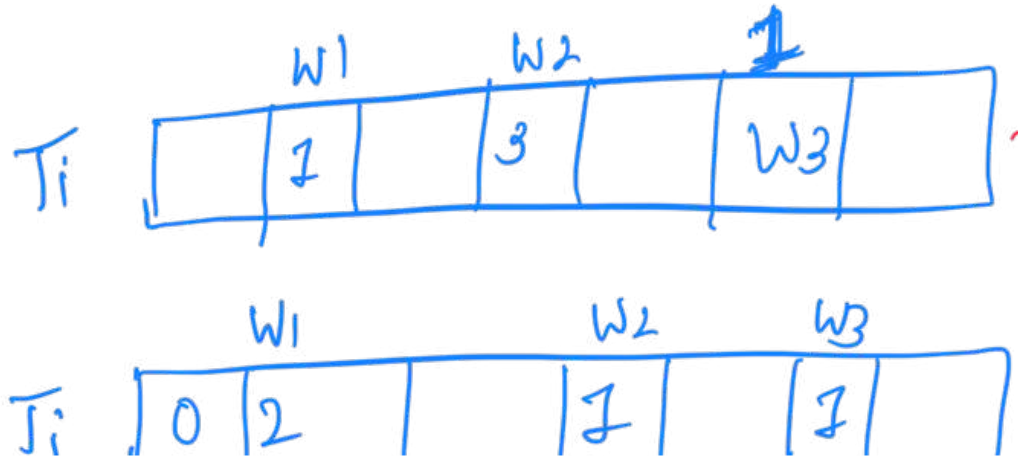


sparse Matrix

① This very big matrix store in sparsed form

① get vector → BOW

② use Euclidean distance for pair wise distance



if most of the words commonly occurring then they are ...

simi

$$dist[t_i, t_j] = \sum_{k=1}^{n} [T_{ik} - T_{jk}]^2$$

deduplication can be improved

2. TF IDF based approach

Term Frequency TF

Inverse Document Frequency IDF

$$TF[W_j, T_i] = \frac{\#\text{ times } W_j \text{ occurs in } T_i}{\#\text{ Words in } T_i}$$

$$IDF(W_i, D) = \log\left[\frac{\#\text{ Document in corpus}}{\#\text{ document in } D \text{ having word } W_i}\right]$$

D: entire corpus

How rare a word is in corpus

TF IDF vector

$T_i \rightarrow W_1, W_2, W_3 \ldots W_d$

$W_1 \rightarrow TF(W_1, T_i) * IDF(W_1, D)$

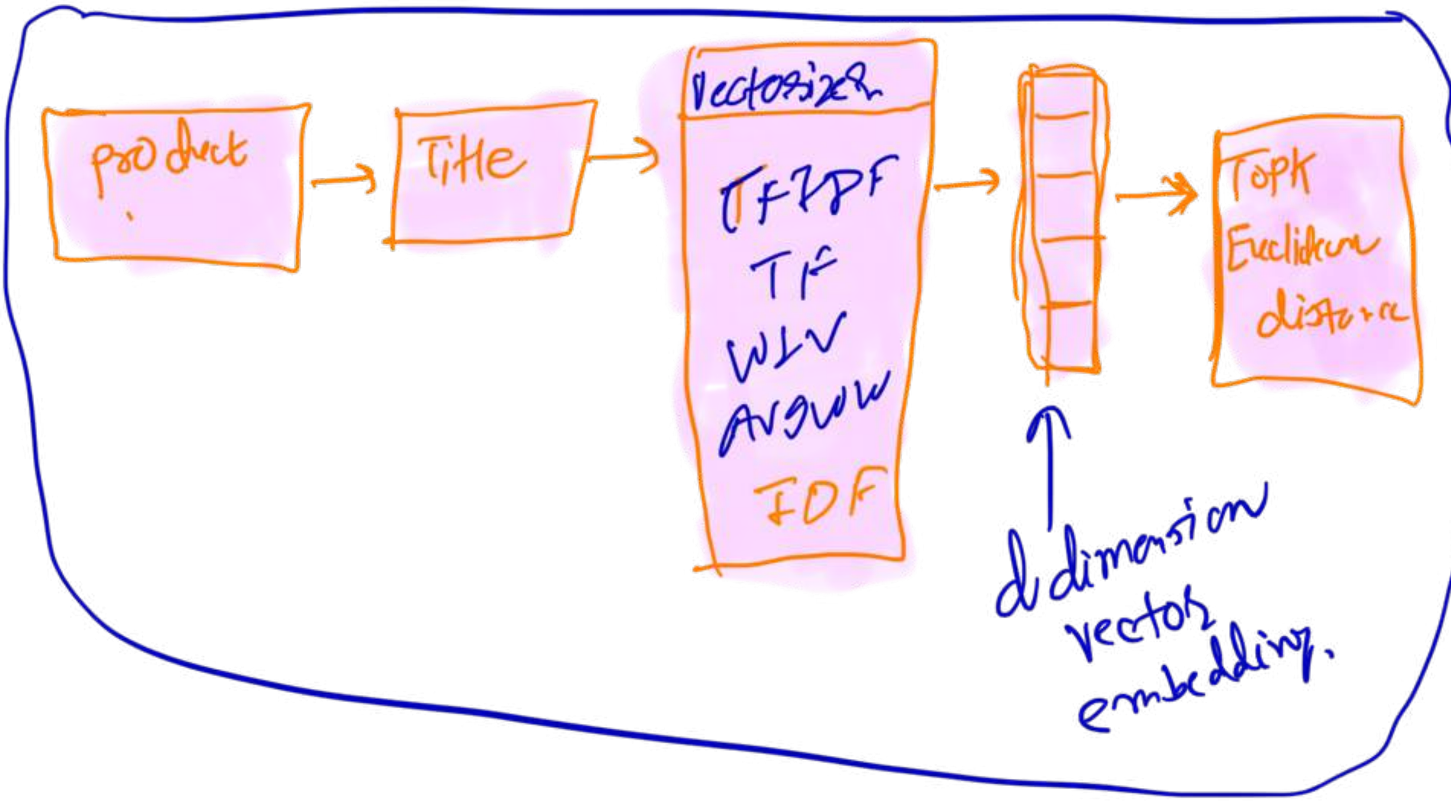
BOW vector

W_1	W_2				W_d
3	2	. .		. .	. .

TF IDF vector

W_1	W_2				W_d
0.2	0.02				0.05

$TF\,IDF(W_i) \uparrow \rightarrow TF \uparrow$



* TF IDF results were better than IDF.

$\boxed{\text{problem with TF IDF}}$ *

Observation (1) words don't Repeat Mostly in our dataset

$$TF(W_i, T_j) = \frac{\# \, W_i \text{ in } T_j}{\# \text{ words in}}$$

words...

T_j

$T_1 \rightarrow$ w1, w2, w3.. w4, w5, w6... w10

$T_2 \rightarrow$ w1, w2, w3, w4

* if all word frequency is 1 then

For T_1 $\boxed{TF(w_i, T_1) = \frac{1}{10} = 0.1}$

For T_2 $\boxed{TF(w_j, T_2) = \frac{1}{4} = 0.25}$

this algorithm will favour the short title

* We can use directly IDF

This experiment we should try, it may work or it may not.

semantic similarity and W2V

zebra
tiger

stripes
leopard

semantically
similar

TF-IDF
BOW
IDF
Fails

Word2vec

	1	2	3	4	d
wi				.....	

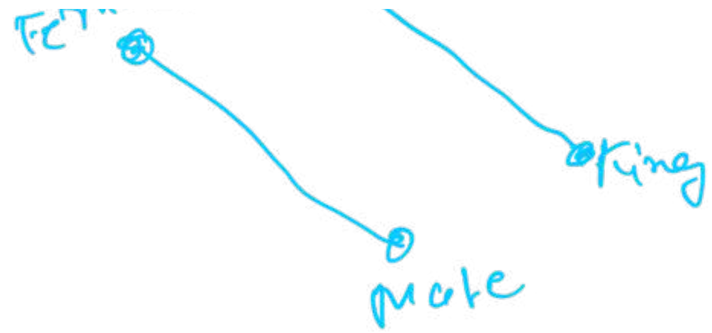
→ dense vector

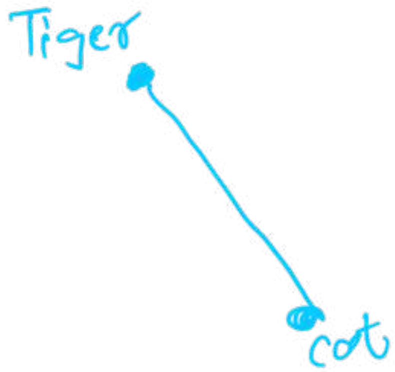
D → Very large corpus required

geometric intuition of W2V

queen

male





$\boxed{\text{google News}} \rightarrow \boxed{\text{W2V}} \rightarrow \boxed{\text{300 dim}}$

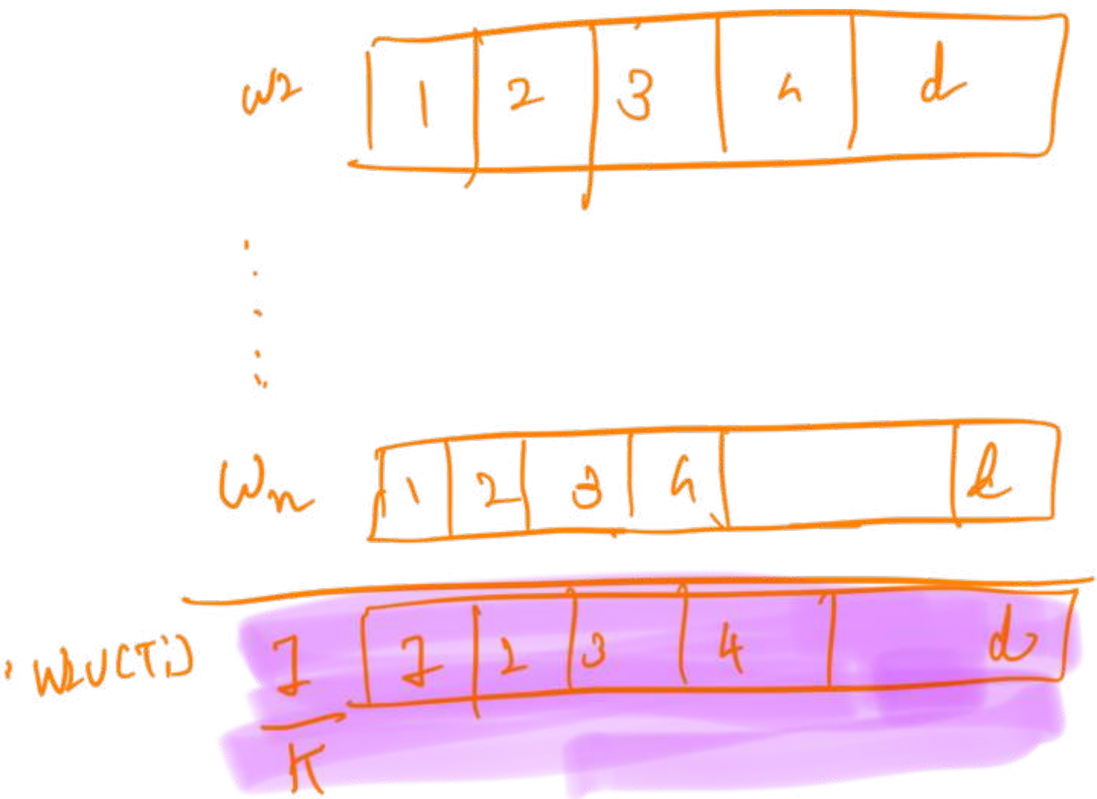
vector provided by google.

For T_1 $\rightarrow$ $w_1, w_2, w_3, \ldots\ldots, w_d$

$\downarrow$ $\boxed{\mathbb{R}^{300}}$ $\boxed{\mathbb{R}^{300}}$ $\boxed{\mathbb{R}^{300}}$ $\boxed{\mathbb{R}^{300}}$

W1

1	2	3	4	0



AVG W2V

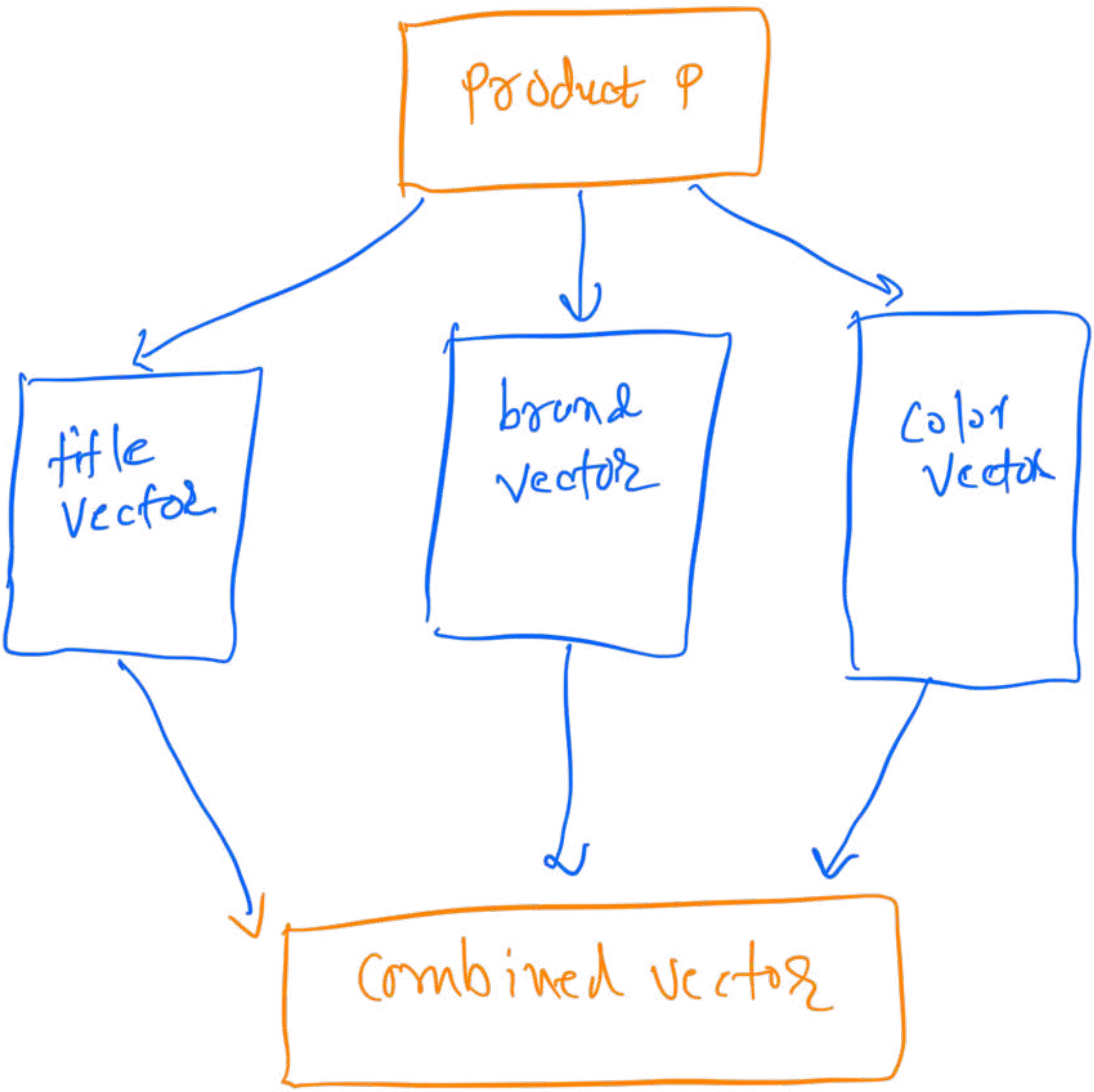
→ We get better result with W2V → this take cares about ↑ similarity semantic

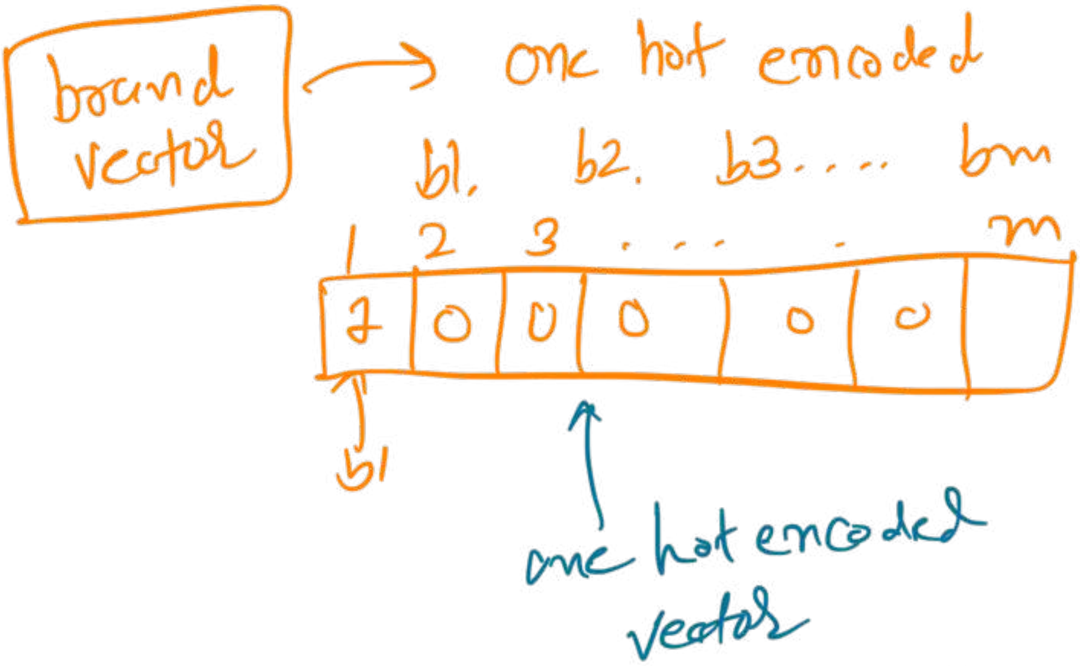
TF IDF Weighted W2V

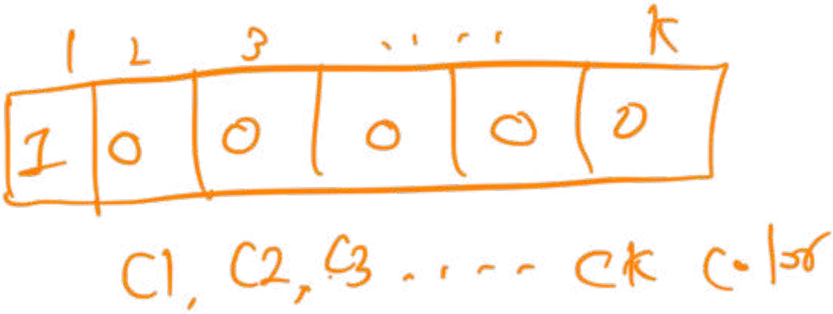
→ word importance is useful

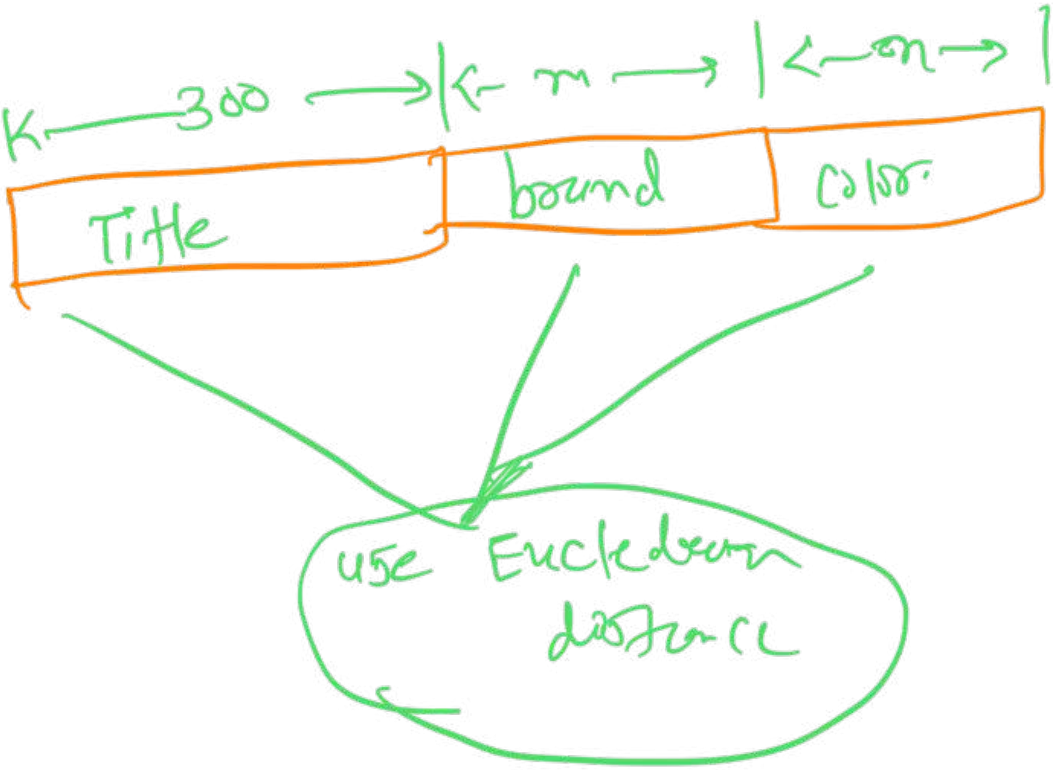
semantic similarity.

using brand and color and other features









(1) prefer showing customers to

specific $\boxed{\text{brand or color}}$

use Weighted Euclidean distance

$W_{Title} = 1$, $W_{brand} = 3$, $W_{color} = 7$

distance = W_{title} * title Feature

+ W_{brand} * brand Feature

+ W_{color} * Color Feature

building Real world solution.

Product query

BoW → 1 2 3 4

TF IDF → 1 2 5 1

W2V → 3 1 5.

↑

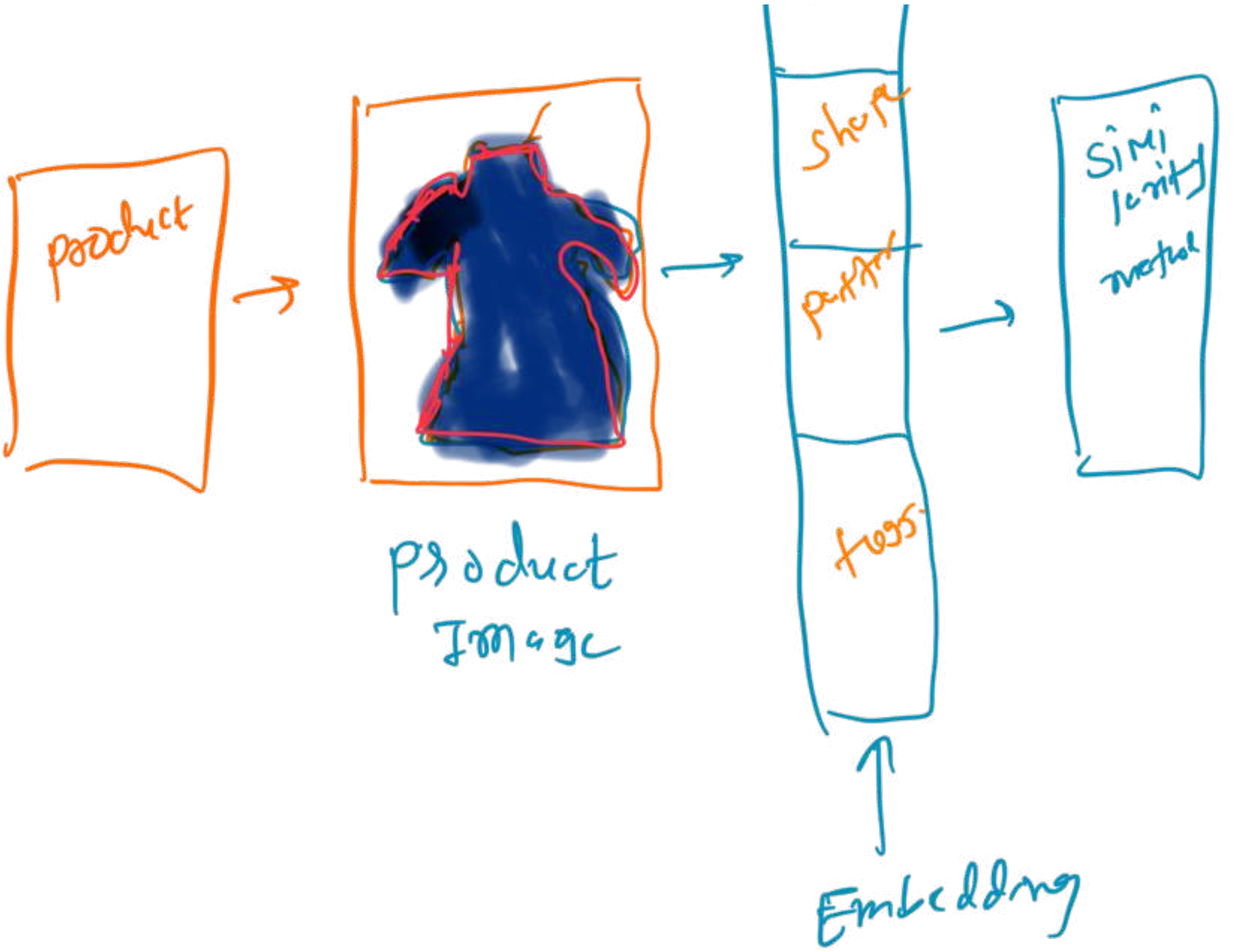
Apply business Rule

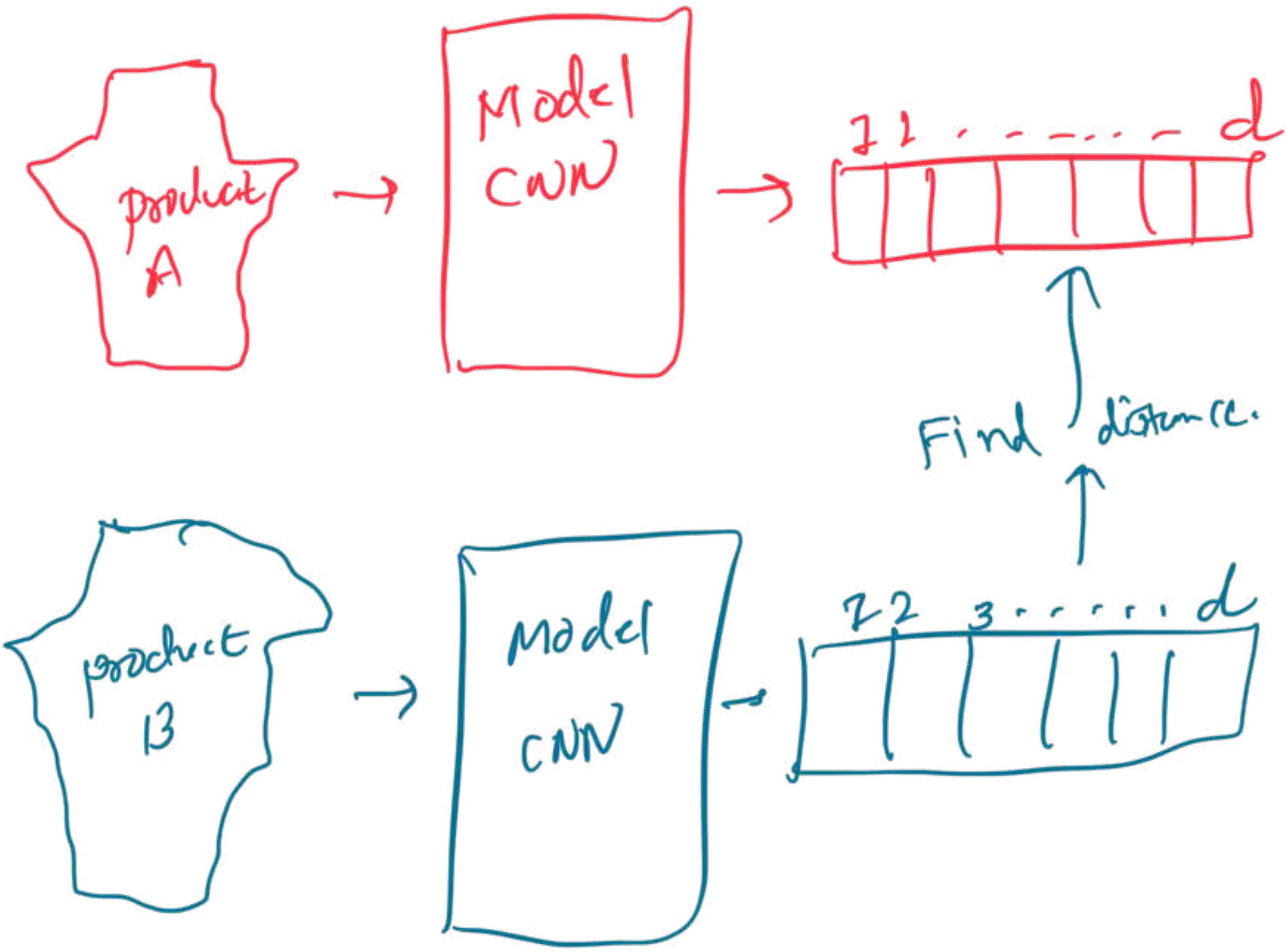
business is most important

Deep Learning based visual similarity.

Algorithm Remains same

Color


product
product Image
shape
pattern
tags
Similarity method
Embedding


product A
Model CNN
1 2 . . . . . . d
Find distance
product B
Model CNN
1 2 3 . . . . . d

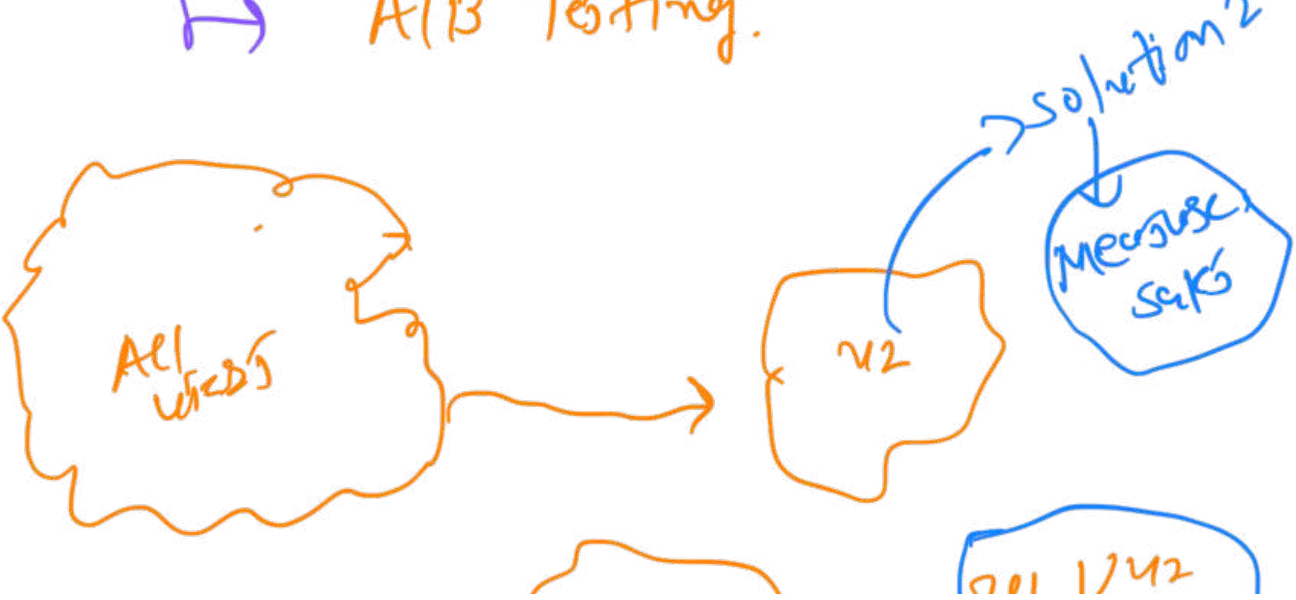
A/B Testing

P_q → Solution 1 → P_1 P_3 P_5 P_7

P_q → Solution 2 → P_3 P_5 P_7 P_8.

How should we evaluate solution 1 and solution 2 quantitatively?

→ A/B Testing.



u_1

$= users$

Solution 1

Measure sales.

$Sales[u_1] > Sales[u_2]$

solution 1 is good