

# Similarity metrics in recommender systems



Kirill Bondarenko · Follow

5 min read · Feb 27, 2019



Deeper to evaluation process

## Introduction

This article is a short explanation of recommender system technique named KNN (k — nearest neighbors) and collaborative filtering. Here will not be long explanation what is KNN and CF. There are a lot of articles for these topics. Like these: [article 1](#), [article 2](#), [article 3](#).

Here will be an explanation **how to choose metrics**, used in models based on KNN principle.

## Similarities metrics

What is similarity between two objects ? In any case, **total similarity is an absence of differences**. For example: we are comparing two people, first one male, 25 y.o. , other male 25 y.o. . By two features (gender and age) we may state they are equal.

Next part requires understanding of next techniques: one-hot encoding (OHE), TFIDF , or other feature extraction methods.

Briefly: OHE: we have a user and three features , answering on a three questions: does he like item 1 ? item 2? item 3 ? If yes -1 , else — 0 . For example, user likes item 1 and 3, but dislikes item 2. In this case we got 3 dimensional vector, user = [1,0,1]. It might be float values too. User likes item 1 on 60% , item 2 on 20% and third 85 % . We got user = [0.6,0.2,0.85].

The same case with vectors. We have two 3 D vectors A and B.  $A = [0,1,0]$  and  $B = [0,1,0]$ . Their coordinates are pairwise equal ( $0=0,1=1,0=0$ ). But what we should do if

we have  $A = [0,1,1]$  and  $B = [1,1,0]$  ? Are they equal ? What percent of similarity between them ?

There are many techniques to compare similarity between two vectors. Let's define two the most popular of them in case of recommender systems.

### Euclidean distance similarity

It's very easy. Similarity is a distance between two vectors, calculated by formula:

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \dots + (p_i - q_i)^2 + \dots + (p_n - q_n)^2}.$$

[Wikipedia article](#)

So, if we have two vectors  $p = [0,0,1,1]$  and  $q = [1,0,1,0]$ , distance between them will be:  $d(p,q) = \sqrt{((0-1)^2 + (0-0)^2 + (1-1)^2 + (1-0)^2)} = \sqrt{(1+0+0+1)} \approx 1.4$ . And we have vector  $q1 = [0,1,1,1]$ . Is it closer to  $p$  than  $q$  ? Calculating distance:  $d(p,q1) = \sqrt{((0-0)^2 + (0-1)^2 + (1-1)^2 + (1-1)^2)} = \sqrt{(0+1+0+0)} = 1$ . We see that  $d(p,q1) < d(p,q) \Rightarrow q1$  is closer to  $p$  than  $q$ . If vectors were users marks, we would say that user  $q1$  is closer to user  $p$  than user  $q$ . So, we may say:

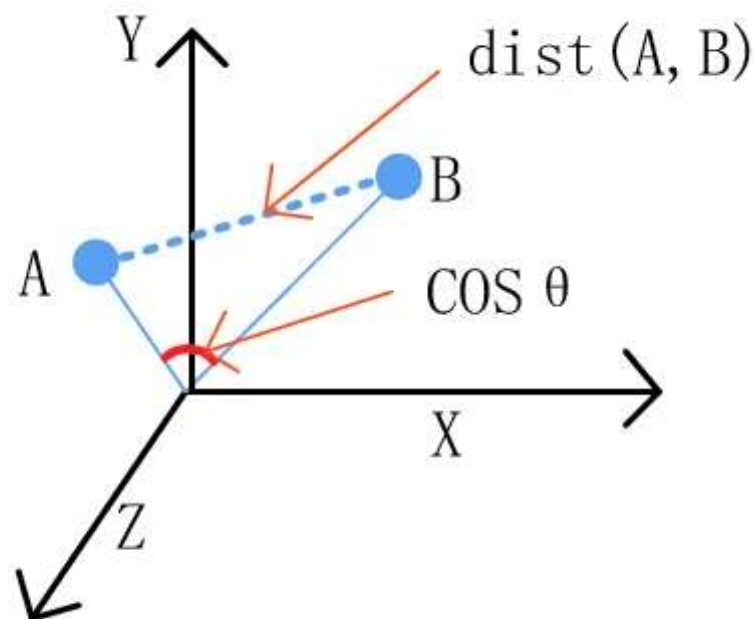
*Decreasing of Euclidean distance between two user-vectors has indirect influence on similarity between them.*

And we may interpret it like a coefficient of similarity  $E_{sim}$ :

$$E_{sim}(p,q) = 1/(1+d(p,q))$$

### Cosine similarity metrics

This method uses cosine of angle between two vectors  $A$  and  $B$  with start point in the zeros of coordinate axes.



Difference with Euclidean metrics

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}},$$

Formula and [Wikipedia article](#)

We have the same vectors  $p = [0,0,1,1]$  and  $q = [1,0,1,0]$  and  $q1 = [0,1,1,1]$ .

$\text{similarity}(p,q) = \text{cosine}(p,q) = \text{dot product of } p \text{ and } q / \text{square root of dot product } p \text{ and } p * \text{square root of dot product } q \text{ and } q = 0.9 / 1.36 = 0.66$ . We interpret it like 66% of similarity. A similarity  $(p,q1) = 1.042 / 1.371 = 0.76$ . It means 76% of similarity. The same results as Euclidean approach.

### What is the difference and how to choose ?

Sure I'm here not to rewrite already made articles. The aim is how to analyze difference between approaches and choose the best one for your task. So first is the difference.

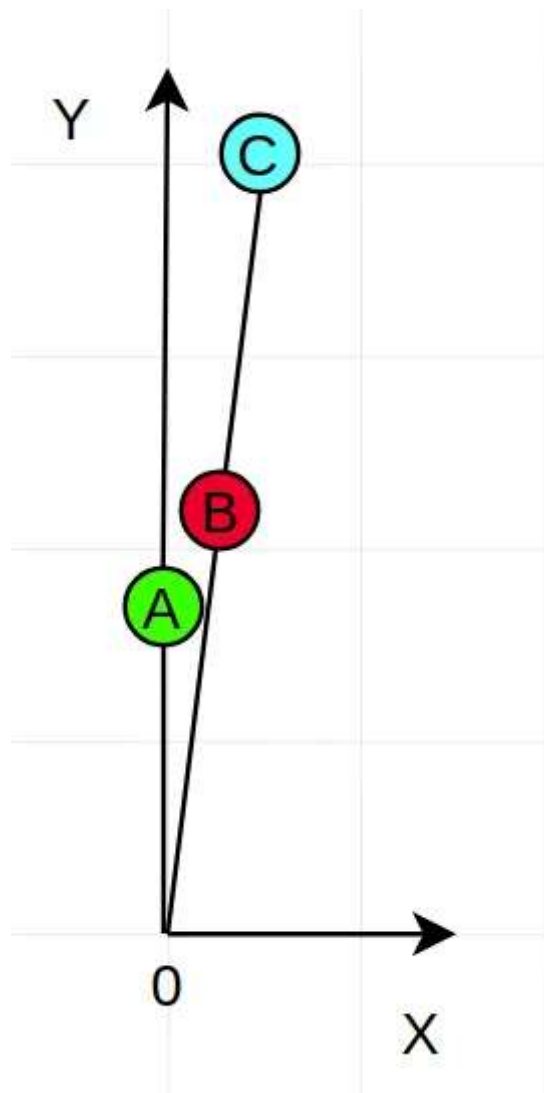
*Euclidean metrics compare absolute values and says that best variant of similar vectors are two absolute identical vectors with zero distance between them.*

But what if we are looking for similar direction vectors ? In case of recommender systems we are matching similar users on the basis of their marks, preferences e.t.c. Mark 0.1 and mark 0.49 are both negative (if we took threshold = 0.5) and both have negative direction in our vision, but absolute distance between them is 0.39. **In this case we want a metric, that may takes to account directions of vectors.**

The key idea was taken from text classification methods. Using TFIDF vectorizing we transform texts of any size to fixed dimensional vector of frequency of appear each word in a text. We don't care about frequency of words 'war' and 'suffering' in two different texts, if they appeared, we understand that both text are about a war conflict (just for example). We may interpret it for out task with user marks. If we understand that both people like the same product, we suppose that they have the same interests.

On this point we may do a little philosophy pause. What is direction of vector of user marks ? User marks is a set of values, describes user's unique preferences, mood, lifestyle e.t.c. You may call it as you like. But the meaning should be common: **we are looking for same "mood" users.**

Let's look on a one example. We have to items X and Y. And users A,B and C. User A has only good rating Y and X=0, user B has good rating for Y and bad for X, user C has rating Y better than B and almost same bad rating for X.



Example of difference

We see, that  $AB < AC$ , and if we take huge threshold in distances we may loose data of C user. If we use cosine similarity, users B and C has the same similarity score with A, ( $A \cdot B = A \cdot C$ ) because they are on the one direction. In this case we can get wider data explanation for future analysis and recommendations.

There are a lot of other metrics like Pearson coefficient, Mean Squared Difference similarity, e.t.c. There is a good library Surprise with well done explained [documentation](#).

### Sources

1. Good [article](#). Thanks to Chris Emmery.
2. [Stackoverflow answer](#).

### From author

Topic of selection the best metrics are the one of the hardest tasks in machine learning. Try to see deeper to see the sense what are you doing.

Many thanks for reading this article! Hope, it helps you to solve your problem.

Bondarenko K. , machine learning engineer.

- Data Science
- Recommendation System
- Similarity
- Cosine Similarity
- Machine Learning

Open in app ↗



Search



Follow



## Written by Kirill Bondarenko

98 Followers

Young and positive thinking machine learning engineer, living in Ukraine.

### More from Kirill Bondarenko





Kirill Bondarenko

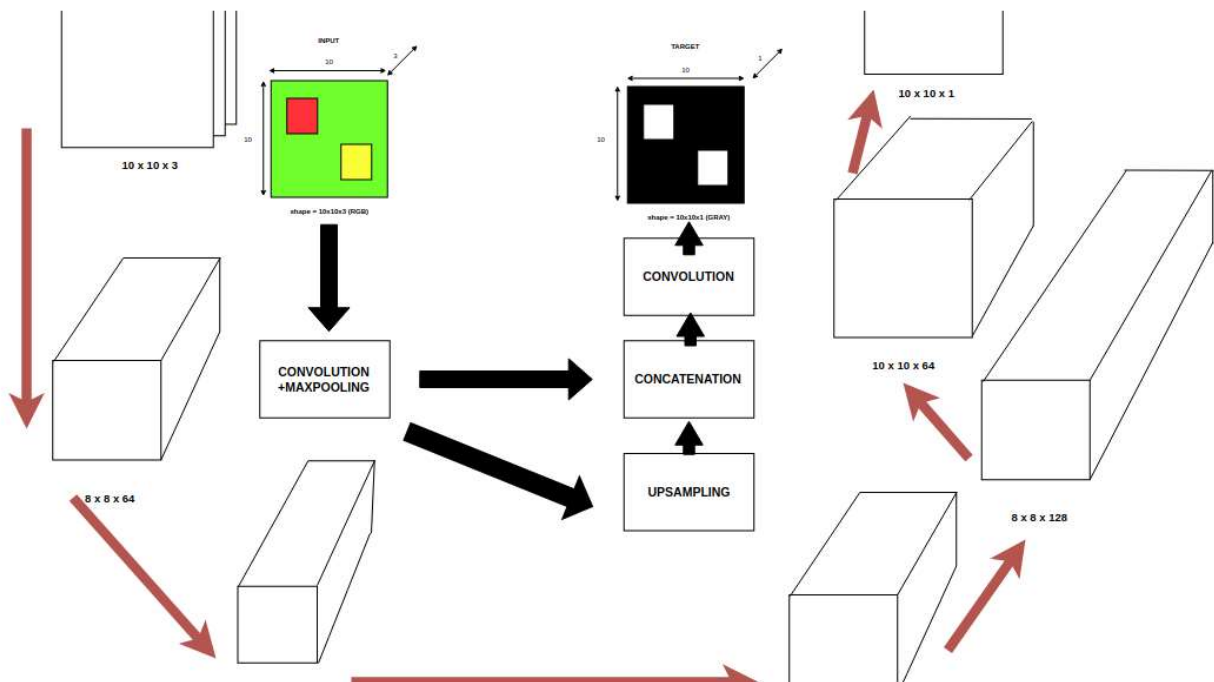
## Best Python program architecture ?

Unfortunately, the most common metaphor for software development is building and construction. [...] Rather than construction, software is...

9 min read · Nov 28, 2020



32



Kirill Bondarenko

## Understanding UNET

How to understand U-Net in the most simple way.

6 min read · Jul 2, 2019

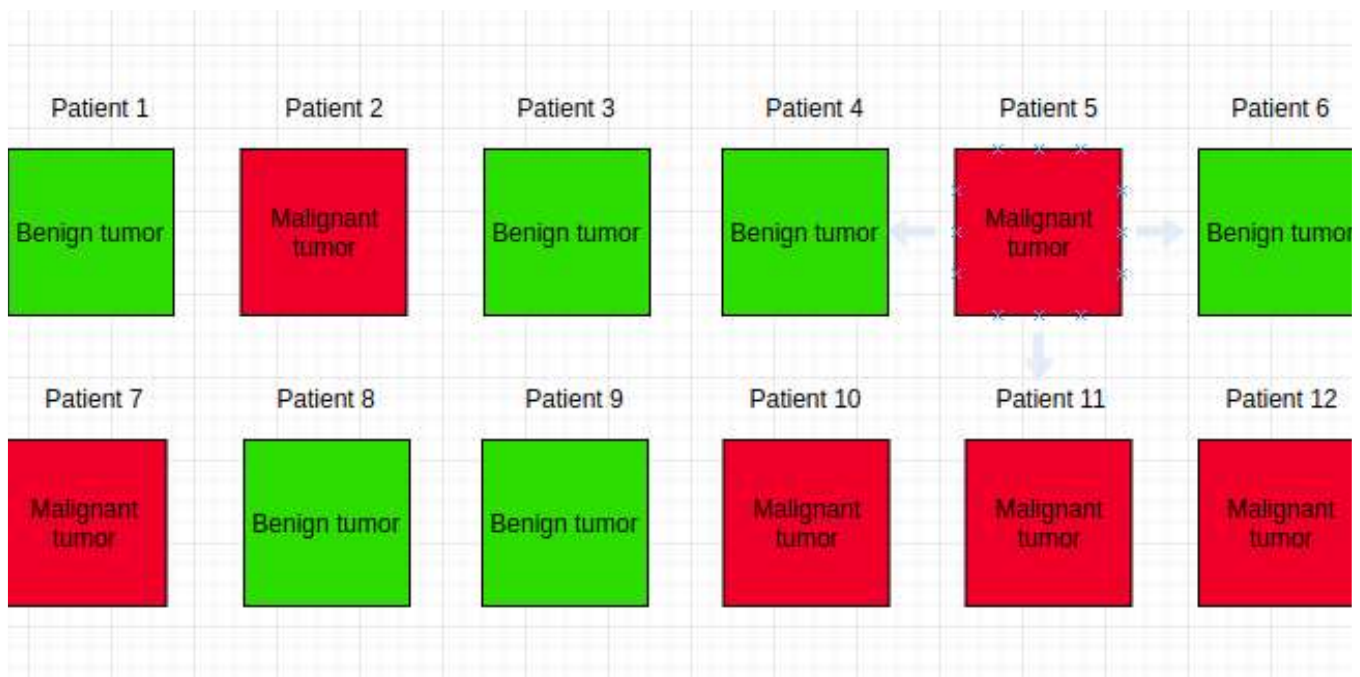


81



2





Kirill Bondarenko

## Precision and recall in recommender systems. And some metrics stuff.

Hello everyone! This story is about how to define quality in recommender systems.

12 min read · Feb 22, 2019



38



3



...




Kirill Bondarenko

## LSTM for real-time recommendation systems

How to prepare your data for LSTM models.



10 min read · Mar 26, 2019

 76







See all from Kirill Bondarenko

Recommended from Medium



 Alexander Shmyga


Vectors similarity. Jaccard Similarity.

In one of my previous posts, I described the essence of vector similarity search and there I explained the two most popular vector...

4 min read · Dec 19, 2023

 6









 Python Programming

## Recommender Systems with Python Code Examples

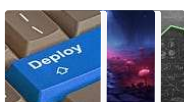
A Comprehensive Guide to Recommender Systems

🌟 • 6 min read • Dec 16, 2023

 53 

### Lists



#### Predictive Modeling w/ Python

20 stories • 1181 saves



#### Practical Guides to Machine Learning

10 stories • 1426 saves



#### Natural Language Processing

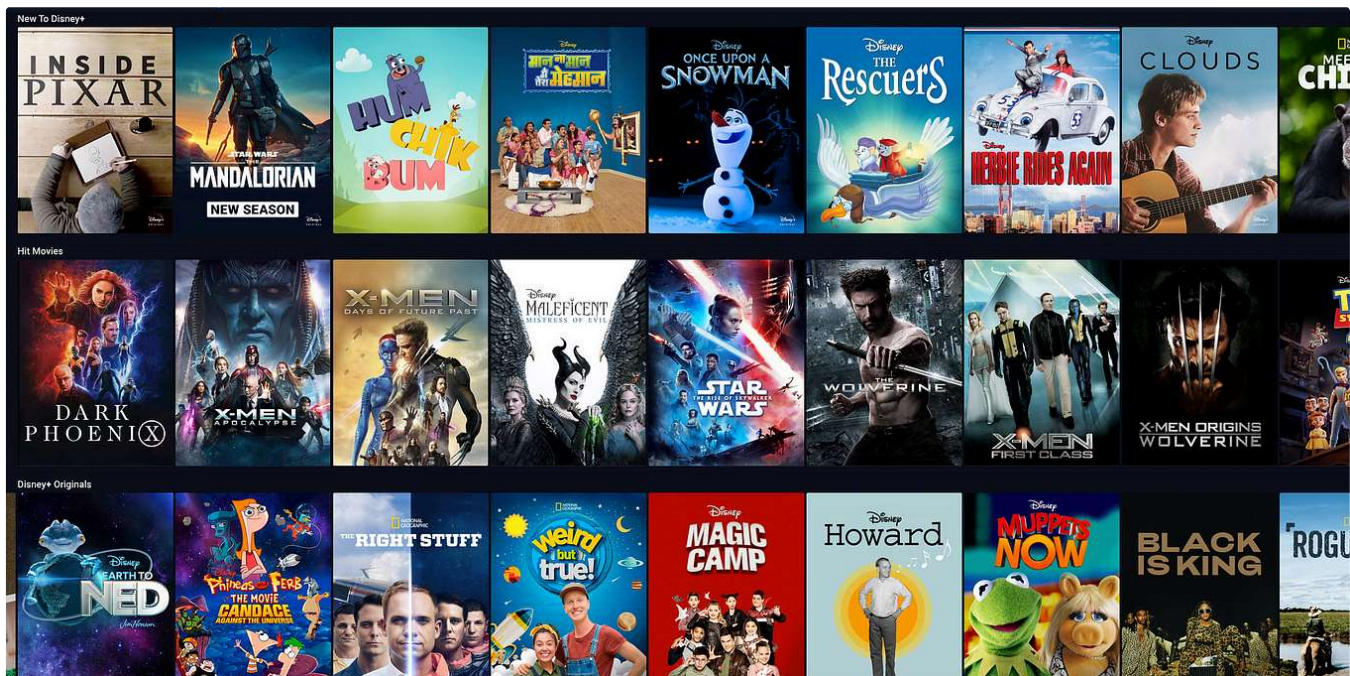
1444 stories • 946 saves




#### data science and AI

40 stories • 153 saves





 Jishnu mohan

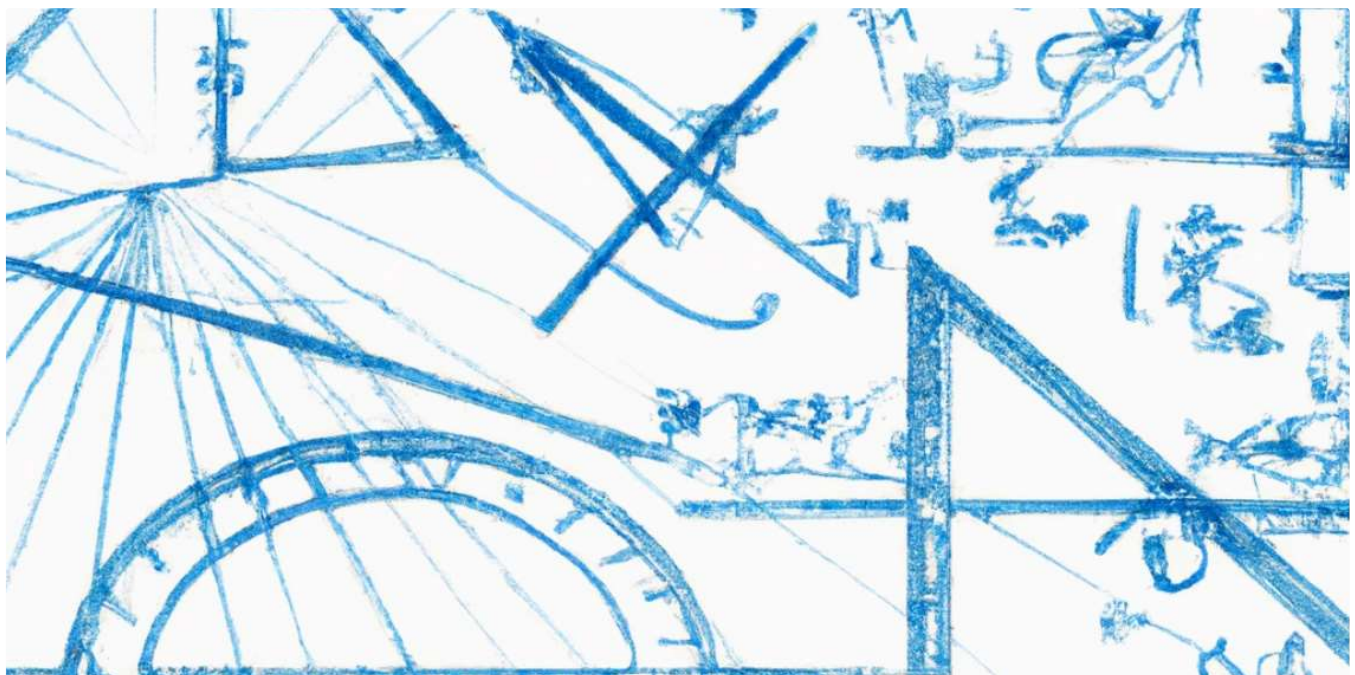
## Movie Recommendation System using Cosine Similarity

Imagine having a movie buddy who knows your taste in films better than you do—always ready with the perfect suggestion for your movie...

5 min read · Jan 30, 2024

 50 



 DataStax

## How to Implement Cosine Similarity in Python

By Phil Miesle

4 min read · Nov 30, 2023



20



Miftahul Ulyana Hutabarat

## Comparing Text Documents Using TF-IDF and Cosine Similarity in Python

How can we know if two text documents are similar? Humans can see the differences, but how can computers know if two text documents are...

6 min read · Dec 17, 2023



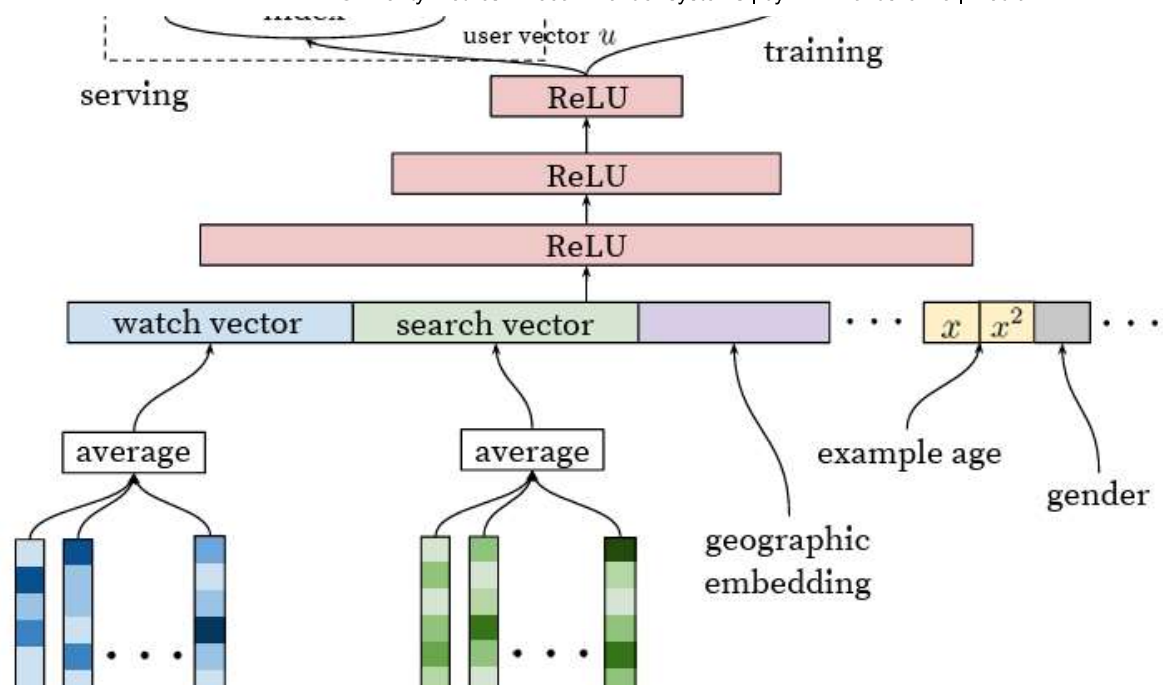
36



1







Michael Roizner in Towards Data Science

## Two-Tower Networks and Negative Sampling in Recommender Systems

Understand the key elements that power advanced recommendation engines

7 min read · Nov 24, 2023



346



3



See more recommendations