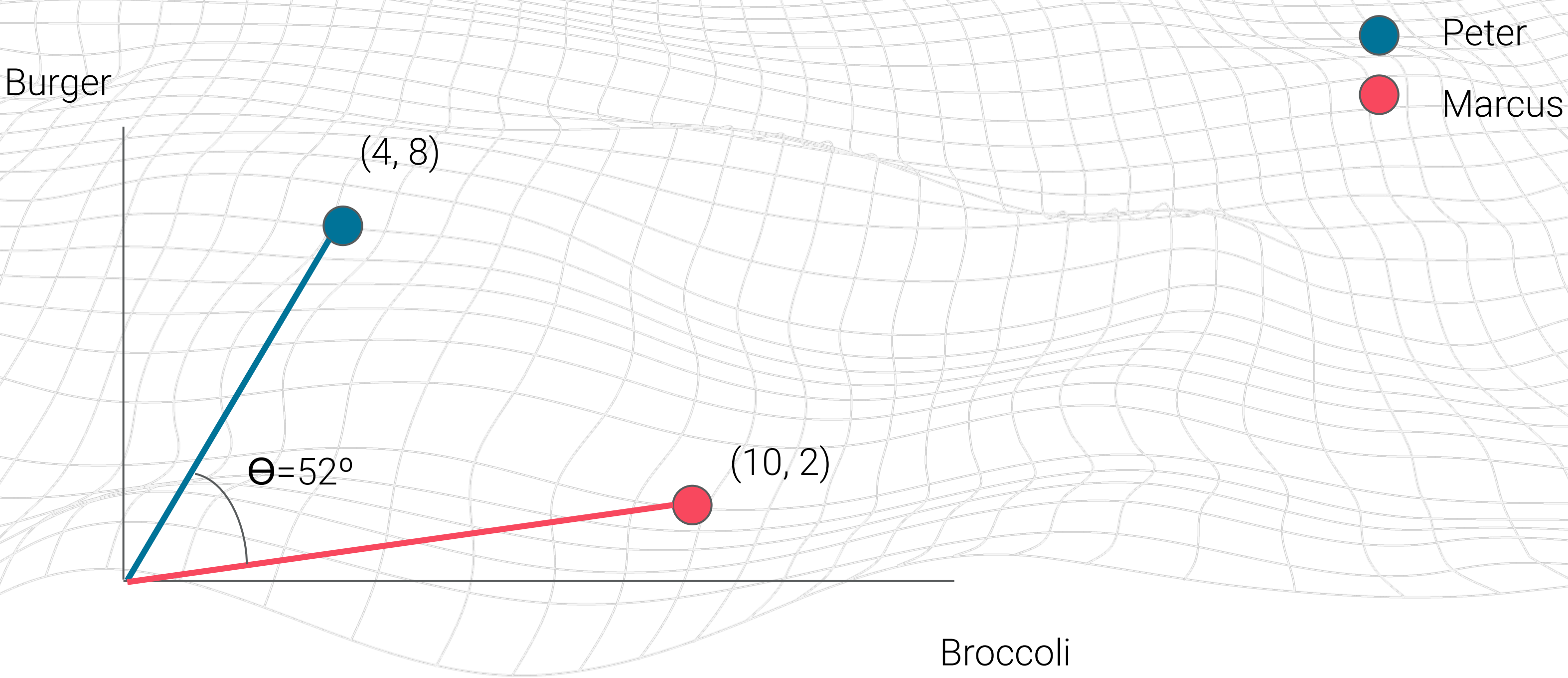
User-based collaborative filtering

Using cosine similarities

	Broccoli	Brussels Sprouts	Hamburger
Markus	8	8	1
Peter	6	6	8
Shikha	7	1	?
Hashim	8	2	9

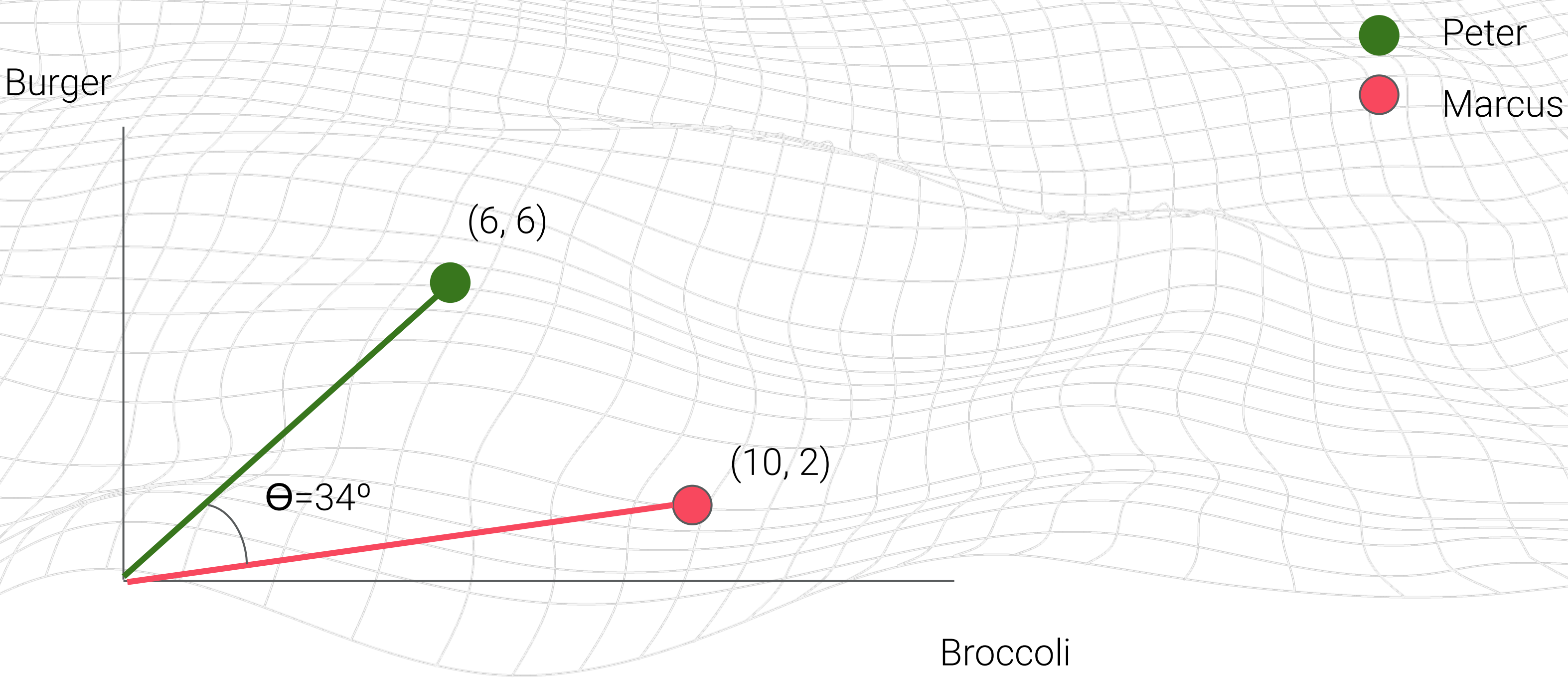
Cosine similarity

Peter likes burger (8) but does not like broccoli (4), while Marcus likes broccoli (10) and does not like burger (2): Their cosine similarity is 0.61:



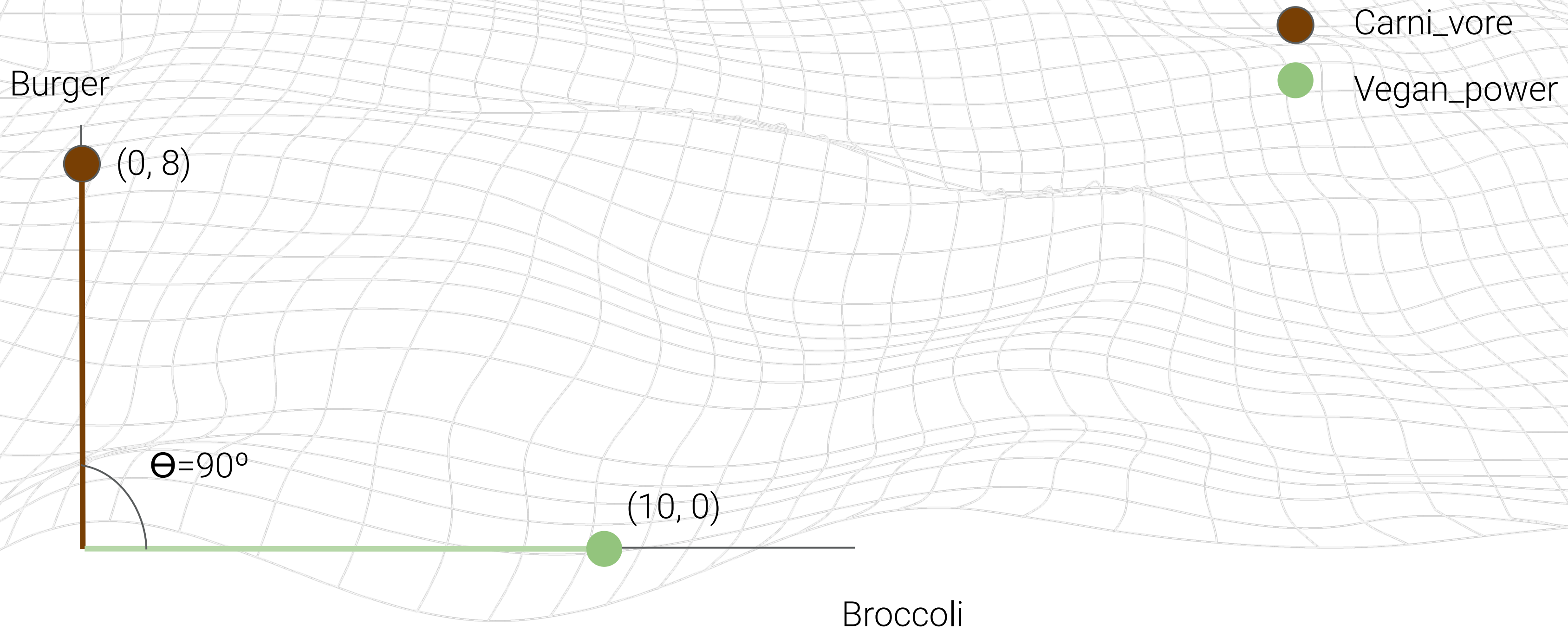
cosΘ = cosine similarity = **0.61**

Peter likes broccoli and burger the same. That makes him more similar to Marcus compared to Shikha: Their cosine similarity is therefore greater (0.82):



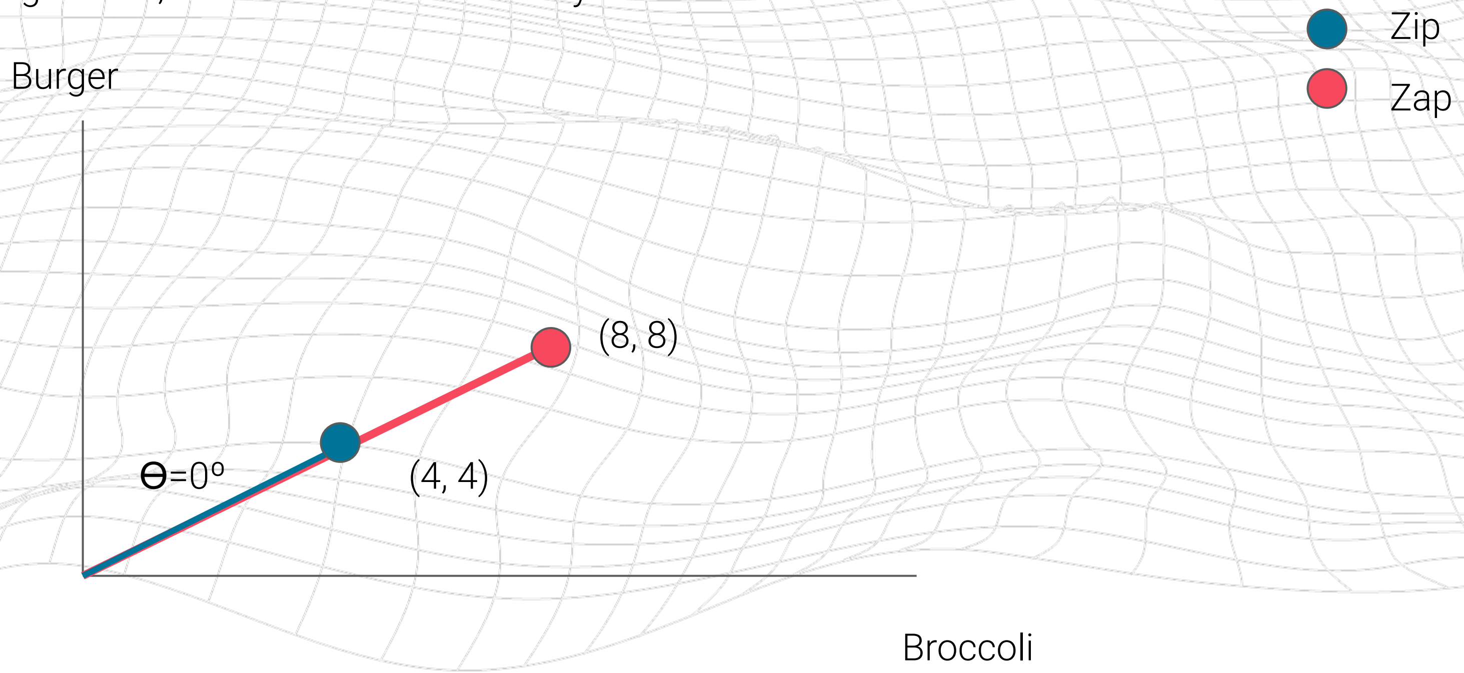
cosΘ = cosine similarity = **0.82**

Carni_vore and Vegan_power have opposite tastes. Their vectors form an angle of 90º, and their cosine similarity is 0:



cosϴ = cosine similarity = **0**

Zip and Zap both like Burger as much as Broccoli. Even if Zap likes both ingredients much more, their vectors form an angle of 0º, and their cosine similarity is 1:



cosϴ = cosine similarity = **1**

We can easily compute cosine similarities between each pair of users using `cosine_similarity` from `sklearn.metrics.pairwise`

	Markus	Peter	Shikha	Hashim
Markus	1			
Peter	0.79	1		
Shikha	**0.80**	0.58	1	
Hashim	0.64	0.93	0.67	1

How would Shikha rate the hamburger?
We will compute the weighted average of the ratings others gave to hamburger. The weights will be the similarities others have with Shikha (divided by their total so that the weights add up to 1)

	Similarities with Shikha	Weights	Burger ratings
Markus	0.79	0.39	1
Peter	0.58	0.28	8
Hashim	0.67	0.33	9
	2,04	1	

Shikha’s rating to burger = (1 * 0.39) + (8 * 0.28) + (9 * 0.33) = **5.6**

Recommending movies

We have used **`pivot_table()`** to create the **`movie_user`** dataframe:

	mov1	mov2	mov3	mov4	mov5	mov6	...
1	5	NaN	NaN	NaN	2	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	4	NaN
3	2	4	4	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	3	NaN
5	NaN	NaN	1	NaN	NaN	NaN	NaN
6	NaN	NaN	NaN	5	NaN	NaN	NaN
7	5	NaN	NaN	NaN	NaN	NaN	NaN
...	NaN	NaN	NaN	NaN	NaN	2	NaN

We have used `pd.DataFrame.fillna()` to replace NaNs with zeros in the `movie_user` table:

	mov1	mov2	mov3	mov4	mov5	mov6	...
1	5	0	0	0	2	0	0
2	0	0	0	0	0	4	0
3	2	4	4	0	0	0	0
4	0	0	0	0	0	3	0
5	0	0	1	0	0	0	0
6	0	0	0	5	0	0	0
7	5	0	0	0	0	0	0
...	0	0	0	0	0	2	0

We have used `cosine_similarities()` to create the **user_similarities** table:

	1	**2**	**3**	**4**	**5**	**6**	**7**	**...**
1	1							
2	0.15	1						
3	0.14	0.19	1					
4	0.3	0.077	0.21	1				
5	0.25	0.081	0.003	0.033	1			
6	0.21	0.001	0.05	0.21	0.04	1		
7	0.17	0.03	0.02	0.001	0.003	0.006	1	
...	0.002	0.06	0.15	0.002	0.02	0.001	0.21	1

These are user 1's similarities with the other users:

2	0.15
3	0.14
4	0.3
5	0.25
6	0.21
7	0.17

Total	1.22

0.15 / 1.22 =

0.14 / 1.22 =

0.3 / 1.22 =

0.25 / 1.22 =

0.21 / 1.22 =

0.17 / 1.22 =

We transform the similarities into weights:

0.12
0.11
0.25
0.20
0.17
0.14

Total	1

Now we want to compute the rating that our **user 1** would give to each movie:

`weights`

2	0.12
3	0.11
4	0.25
5	0.20
6	0.17
7	0.14

`movie_user`

	mov1	**mov2**	**mov3**	**mov4**	**mov5**	**mov6**
2	0	0	0	0	0	4
3	2	4	4	0	0	0
4	0	0	0	0	0	3
5	0	0	1	0	3	0
6	0	0	0	5	0	0
7	5	0	0	0	0	0

mov1 = (0.12 * 0) + (0.11 * 2) + (0.25 * 0) + (0.20 * 0) + (0.17 * 0) + (0.14 * 5) = **0.92**

mov2 = (0.12 * 0) + (0.11 * 4) + (0.25 * 0) + (0.20 * 0) + (0.17 * 0) + (0.14 * 0) = **0.44**

Now we want to compute the rating that our user 1 would give to every movie. We can use the dot product, `.dot()`:

`weights`

2	0.12
3	0.11
4	0.25
5	0.20
6	0.17
7	0.14

●

`movie_user.T`

	2	**3**	**4**	**5**	**6**	**7**
mov1	0	2	0	0	0	5
mov2	0	4	0	0	0	0
mov3	0	4	0	1	0	0
mov4	0	0	0	0	5	0
mov5	0	0	0	3	0	0
mov6	4	0	4	0	0	0

=

mov1	0.92
mov2	0.44
mov3	0.64
mov4	0.85
mov5	0.60
mov6	1.48

mov1 = (0.12 * 0) + (0.11 * 2) + (0.25 * 0) + (0.20 * 0) + (0.17 * 0) + (0.14 * 5) = **0.92**

mov2 = (0.12 * 0) + (0.11 * 4) + (0.25 * 0) + (0.20 * 0) + (0.17 * 0) + (0.14 * 0) = **0.44**

Challenge:

Find the top recommendations for `user 80`

Train - test split “manually”:

	mov1	mov2	mov3	mov4	mov5	mov6	...
1	5	0	0	0	2	0	0
2	0	0	0	0	0	4	0
3	**0**	4	**0**	0	0	0	0
4	0	0	0	0	0	3	0
5	0	0	1	0	0	0	0
6	0	0	0	5	0	0	0
7	5	0	0	0	0	0	0
...	0	0	0	0	0	2	0

Challenge:

Replace 2 restaurant ratings from user `U1001` for zero, and predict the rating.