

Hey there, this is Prem Swaroop

Let's see

How we can convert words into numbers

When we are dealing with any NLP task

Ur input is text, and machine learning models cant understand text

So u have to get it converted into a number



Let's say we are building an NLP model for this game



The task u have in hand is to recognize entities in a given sentence,

For ex: Dhoni will be player name

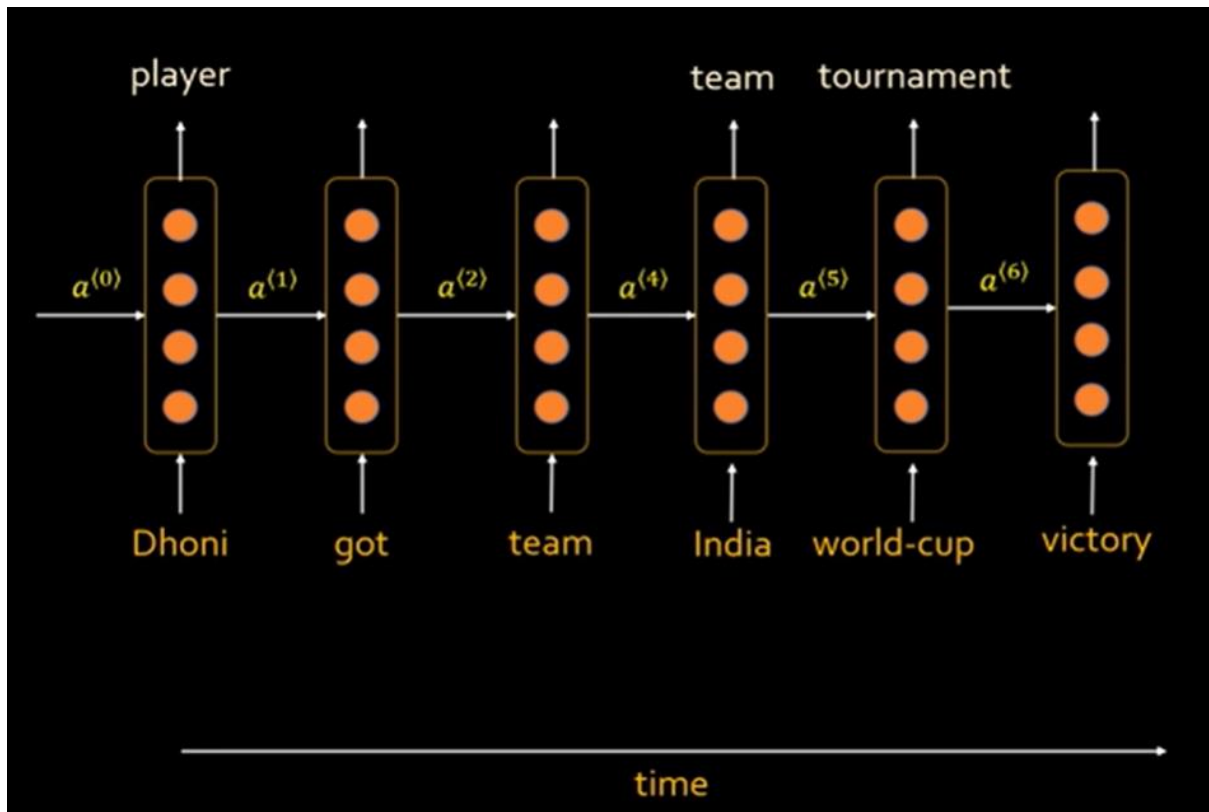
India would be a team name

And world cup would be a tournament name

Similarly u have a different statement nd u want to identify the entities



So this is called Named Entity Recognition

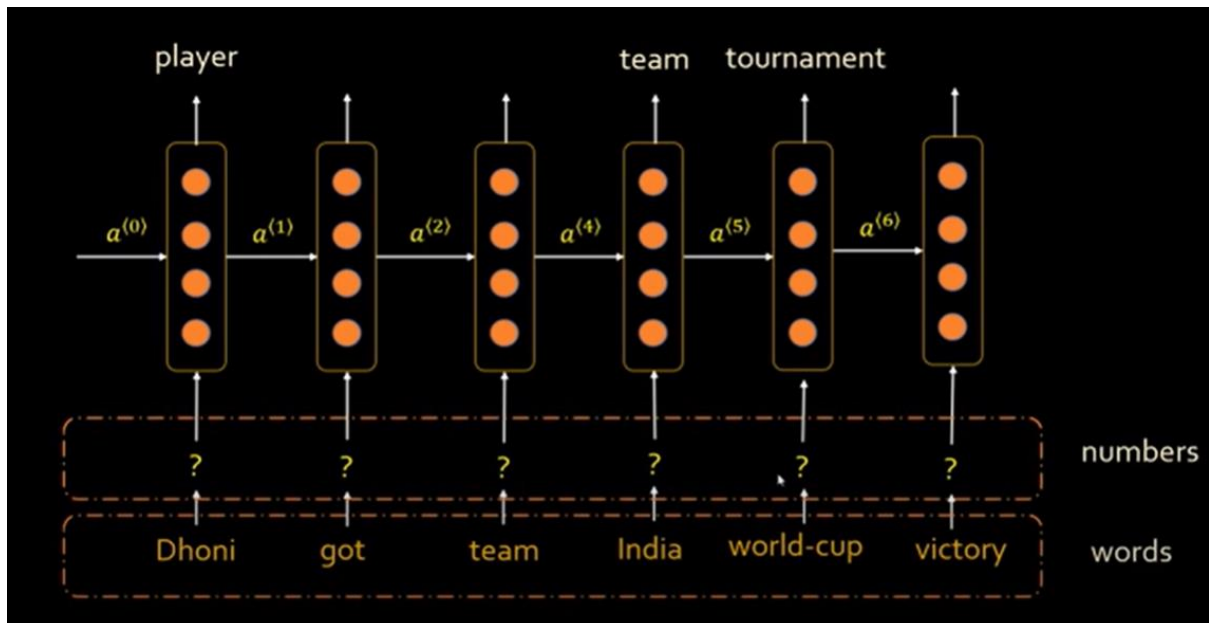


And for Named Entity Recognition u can build an RNN like this.

And there is a problem here

The problem is input – machine learning model can't understand text so we have to convert into a number

So how do we do that



So as we can see we have a text

We have to put this extra layer as an input and u have to convert all these words to numbers

RESULT
4th Test, Ahmedabad, Mar 3 - 7 2021, England tour of India

England (54.5 ov) 205 & 135

India 365

India won by an innings and 25 runs

PLAYER OF THE MATCH
Rishabh Pant
India

Summary Scorecard Commentary Report Videos Coverage Statistics

ENG 2nd Innings Full commentary

India at the 2011 World Cup [\[edit \]](#)

Main article: 2011 Cricket World Cup

As one of the host nations for the 2011 World Cup, India were expected to perform well in familiar conditions, and were considered pre-tournament favourites by the media and press. Like in 2007, India came into the World Cup on a string of strong performances, with back-to-back series wins against Australia and New Zealand at home, followed by a moderately successful tour of South Africa.

The Indian team were generally considered to be the strongest batting side in the tournament, comprising the openers Virender Sehwag and veteran Sachin Tendulkar, playing in his 6th consecutive World Cup, followed by Gautam Gambhir and Virat Kohli, with Yuvraj Singh, skipper Mahendra Singh Dhoni, Yusuf Pathan and Suresh Raina completing the star-studded batting line-up. While the bowling attack was considered more suspect, three veterans in pacers Zaheer Khan and Ashish Nehra and offspinner Harbhajan Singh were joined by Munaf Patel, Piyush Chawla, Ravichandran Ashwin, and Shantakumaran Sreesanth. The 1996 World Cup format was used for the tournament, following widespread criticism, particularly from the BCCI, over the 2007 format. India were placed in Group B in the Group stage alongside co-hosts Bangladesh, South Africa, England, the West Indies and associates Holland and Ireland.

India's 2011 World Cup campaign started with an 87-run win against Bangladesh at Dhaka. With centuries from Sehwag (175 from 140 balls, 14 fours, 5 sixes) and Kohli (100 not out from 83 balls, 8 fours, 2 sixes) India scored 4/370. Fast bowler Munaf (4-48) took 4 wickets during the Bangladesh reply, including that of opener Tamim Iqbal (70 from 86 balls, 3 fours, 1 six) as Bangladesh scored 9/283 in 50 overs to fall short.

India next played England at Bangalore, which was a thriller. On a batting-friendly track at the M. Chinnaswamy Stadium, India chose to bat first. Tendulkar (120 from 115 balls, 10 fours, 5 sixes) lashed his way through the English attack, ably supported by Gambhir (51 from 61 balls, 5 fours) and Yuvraj (58 from 50 balls, 9 fours). After the 45th over, India were 305/3 and were looking to pass 350 during the batting Powerplay. Instead, English bowler Tim Bresnan (5-48) engineered a collapse with four quick wickets in 16 deliveries, as India slumped to a still-formidable total of 338 all out. England started their run chase by blasting 77 runs off the first 10 overs. Skipper Andrew Strauss (158 from 145 balls, 18 fours, 1 six) decimated the Indian bowling attack with unparalleled ferocity, and was supported by Ian Bell (69 from 71 balls, 4 fours, 1 six). At 2/280 in the 43rd over, England was cruising to an extraordinary victory. However, Zaheer responded by taking the wickets of Strauss, Bell, and Paul Collingwood in 11 deliveries, as England were reduced to 289/6. Tailenders Bresnan, Graeme Swann, and Ajmal Shahzad each hit massive sixes in the final few overs to regain some momentum, and Swann scored 13 runs off the final over to salvage a tie with India (338/8 in 50 overs). It was only the fourth tied match in World Cup history.

Now lets say for training the model u scrap the internet

So u go to website like ESPN cric info, Wikipedia and etc

U try to grab all the articles to build ur language model

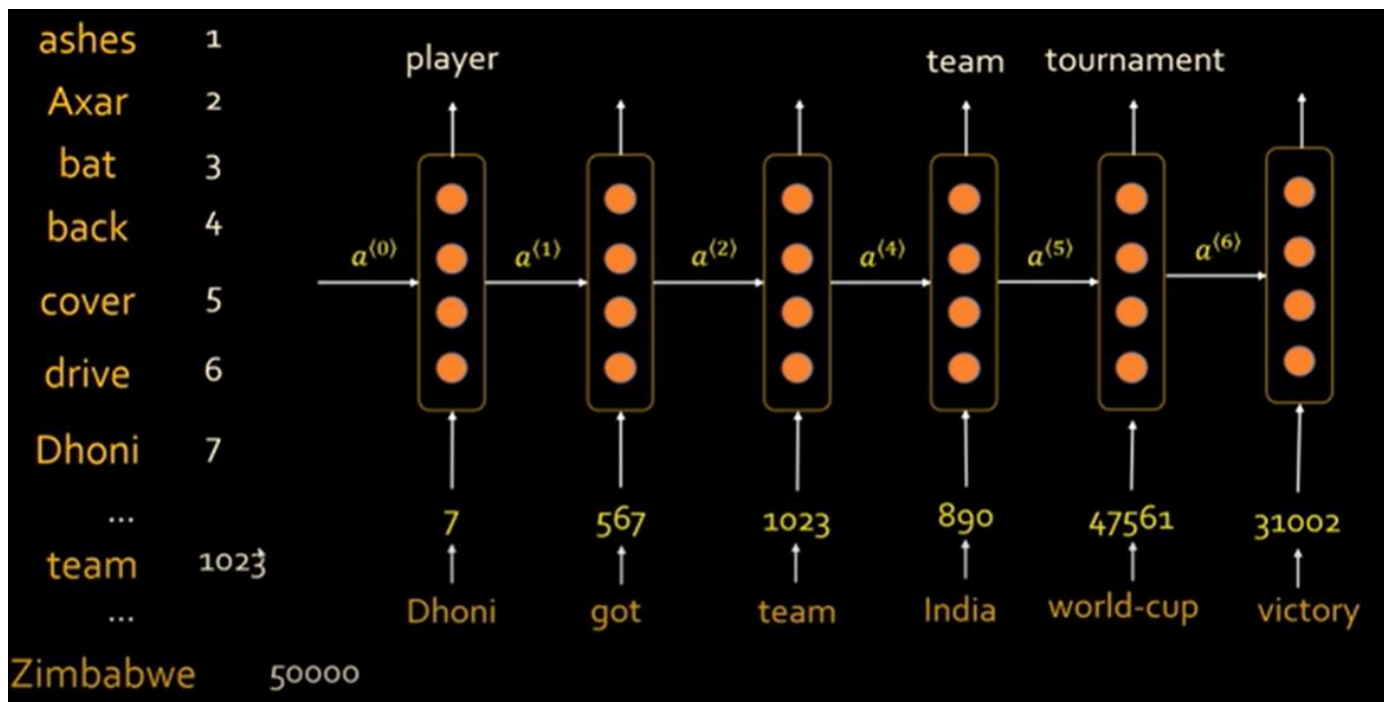
& u build a vocabulary out of it

So vocabulary is nothing but list of words

ashes	1
Axar	2
bat	3
back	4
cover	5
drive	6
Dhoni	7
...	
team	1023
...	
Zimbabwe	50000

So lets say this is our vocabulary. Lets say our vocabulary has 50000 words

U can assign unique number to each of these words and u can use those numbers as an input to ur RNN

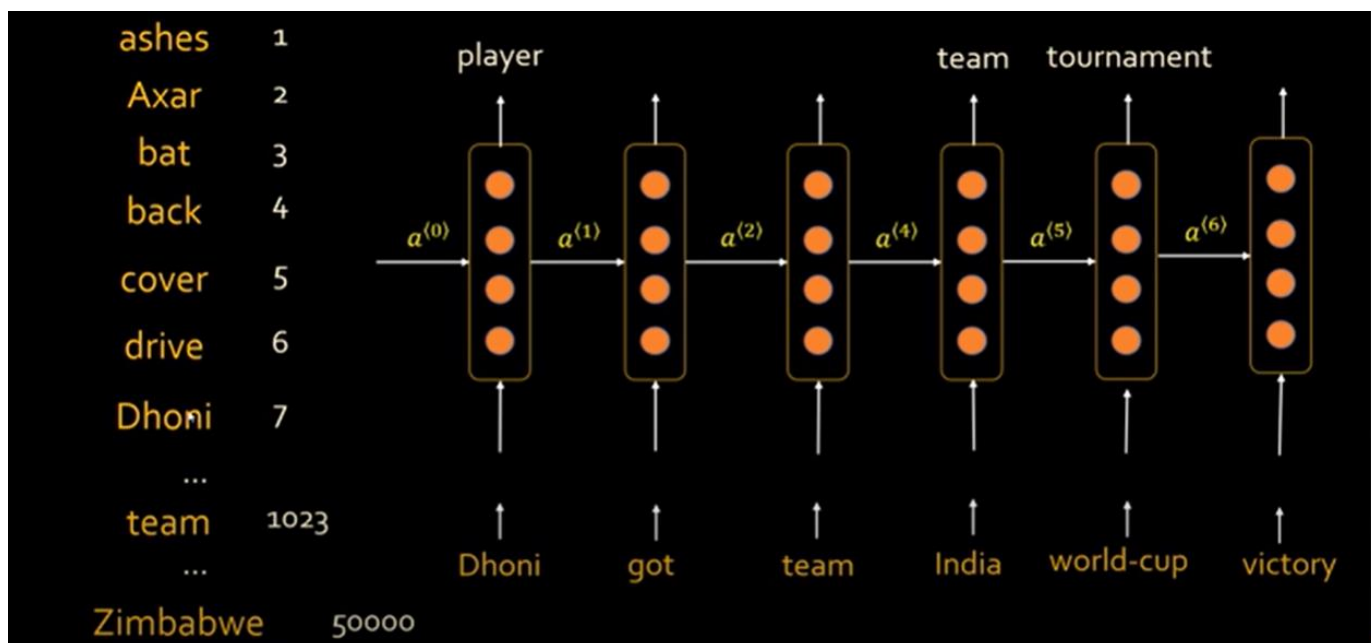


Now that our **option number 1** basically

**** Converting words into unique numbers based on ur vocabulary**

Issue with option 1: unique numbers

Numbers are random. They don't capture relationship between words



For example in this case

Dhoni's number is 7, Axar number is 2

Dhoni and axar both are players so may be these number should be similar but they are far apart

Whereas Ashes is name of the tournament and it is nearer to Axar

So these kind of random numberings can't capture the similarities between the words

Option 2 – One Hot Encoding

	ashes	Axar	Bat	back	cover	drive	dhoni	..	Zimbabwe
ashes	1	0	0	0	0	0	0	...	0
Axar	0	1	0	0	0	0	0	...	0
Bat	0	0	1	0	0	0	0	...	0
Back	0	0	0	1	0	0	0	...	0
...									
Zimbabwe	0	0	0	0	0	0	0	...	1

Basically, u take all ur words

when it is ashes - for ashes u put 1, remaining words u put 0 and u get a vector

for example the vector for axar would be 0100.....

so this is a popular approach in machine learning but this also have issues like

Issue with option 2: one hot encoding

1. Doesn't capture relationship between words
2. Computationally in-efficient

The 1st issue is they don't capture the relationship

If u see if I don't tell u the words

If u just look at the numbers u cant tell if two words are same

Like banana and grapes for example these are 2 fruits, so if u have these 2 fruits maybe u want their vectors to be little similar

The 2nd issue is it is computationally inefficient

If ur vocabulary has 50,000 words for each word u get a vector whose size is 50,000.

Often In machine learning problems the number of words could be in millions or trillions

So the computational cost will become very high

Option 3 – Word Embeddings

Word embeddings allow u to capture relationship between two words.

Lets see how do we capture similarities between two words

Lets take the example – if u have 2 homes and u want to figure out if these homes are similar or not

Well how will we do that

Well we look at the features of the home

What are the features – its simple right

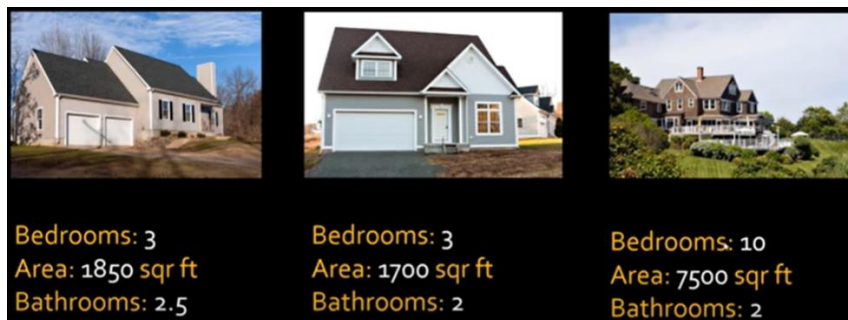
Like the bedroom area , bathrooms and so on



So if u look at these two homes, the bedrooms are same , area is almost same u know 1700 vs 1850

Bathrooms are also kind of near to each other

So then we can say that these two homes are similar based on the features



If u look at third home, it's a big mansion, and if we look at the features of this home u know the feature vector would be 10 ,7500, 2

And when I compare this vector with this vector, I will obviously figure out that these two homes are not similar

Can we apply similar concepts to the





The words, lets say these are our words Dhoni, Cummins, Australia

How can we compare these words, now as a human we know that Dhoni and Cummins are similar bcz these are names of players

But Australia is not similar bcz its the name of the country or name of the team

So then just think about is there a way to retrieve features from these words

If u think hard u might come up with some features such as

		
Dhoni	Cummins	Australia
Person: 1 Healthy/fit: 0.9 Location: 0 Has two eyes: 1 Has government: 0	Person: 1 Healthy/fit: 0.87 Location: 0 Has two eyes: 1 Has government: 0	Person: 0 Healthy/fit: 0.7 Location: 1 Has two eyes: 0 Has government: 1

So in the case of Dhoni – yes he is a person so 1

Yes he is pretty healthy so 0.9

Is Dhoni a location – no so 0

Does dhoni have two eyes – yes so 1

Does dhoni has government - no that's sounds weird so 0

When I put similar features for cummins, I will probably get similar kind of vectors for both dhoni and cummins

For dhoni I will get 1 0.9 0 1 0

For cummins I will get 1 0.87 0 1 0

So both are kind of similar vectors.

So of Australia when u put similar features for it ,

Is Australia a person – no so 0

Is Australia healthy or fit – yes 0.7

Is Australia a location – yes so 1

Does Australia have two eyes – sounds weird so 0

Does Australia have government – yes so 1

If we have any other countries their feature vector will match the feature vector that we have for Australia

So we converted word into a vector

Vector is just a bunch of numbers right.



And we converted what we found is Dhoni and Cummins are kind of similar

But Cummins and Australia are not similar, some of the numbers may match but if u compare the entire vector , they are not very similar

Dhoni and Australia are not similar , some of the numbers may match but if u compare the entire vector , they are not very similar

So now we found the feature vectors for these words and that could be very effective to solve variety of problems in NLP tasks

And this is basically word embeddings

So we look at our original vocabulary

May be u can come up with this kind of a feature vector

	ashes	Australia	Bat	Cummins	cover	Dhoni	World cup	..	Zimbabwe
Person	0	0.02	0.1	0.95	0.03	0.96	0.67	...	0.04
Country	0	0.97	0	0	0	0	0	...	1
Healthy & Fit	0	0	0.3	0.87	0	0.9	0	...	0
Event	1	0.1	0	0	0.4	0	1	...	0
gear	0	0	1	0	0	0	0	...	0

Where we have features like person, country, healthy & fit, event, gear

When u train ML model u might come up with these random numbers so if it is Australia country then 0.97

1 means its perfect

Using ML u can derive these kind of vectors

	ashes	Australia	Bat	Cummins	cover	Dhoni	World cup	..	Zimbabwe
Person	0	0.02	0.1	0.95	0.03	0.96	0.67	...	0.04
Country	0	0.97	0	0	0	0	0	...	1
Healthy & Fit	0	0	0.3	0.87	0	0.9	0	...	0
Event	1	0.1	0	0	0.4	0	1	...	0
gear	0	0	1	0	0	0	0	...	0

	0.95	0.96		0.02	0.04	
	0	0		0.97	1	
Cummins	1	1	Dhoni	0	0	Zimbabwe
	0	0		0.1	0	
	0	0		0	0	

Ofcourse when u look at feature vectors for cummins vs dhoni they are matching

So u know that these two are players

And when u look at feature vectors for Australia and Zimbabwe they are also kind of matching

So u understand that these two are countries or team names

So u machine learning to convert words into these kind of vectors

And those vectors are fed again into RNN and then we solve variety of problems like the 'Sentiment Analysis' or 'Named Entity Recognition' and so on and variety of NLP tasks

So just to summarise we looked at three methods of converting words into numbers

Converting words to numbers	
1. Unique numbers	
2. One hot encoding	
3. Word embeddings	
1. TF-IDF	
2. Word2Vec	

How word embeddings are calculated ?

Embeddings are **not hand crafted**. Instead, they are learnt during neural network training

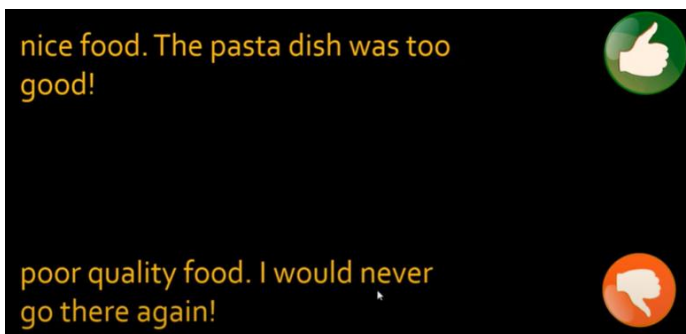
Techniques to compute word embeddings

1. Using **supervised** learning
2. Using **self-supervised** learning
 1. Word2vec
 2. Glove

1. Using **supervised** learning

Take an NLP problem and try to solve it.
In that pursuit as a side effect, you get word embeddings

Let's see u r doing a food review classification, whether the review of the food is positive or negative



This is a NLP task

Now in order to do this Food Review Classification u need to train ur neural network and after the neural network is trained u will get word embeddings as a side-effect

The problem u r solving here for food review classification is almost like a fake problem. U don't care about this problem very much

U care more about the word embeddings

Lets say we have 100 food reviews , for that what we will do is we will decide the vocabulary

So vocabulary is all the words that appear in the full review

So lets say u have 5000 words in ur vocabulary

After	food	nice	...	poor	...	zonal
1	2	3				5000

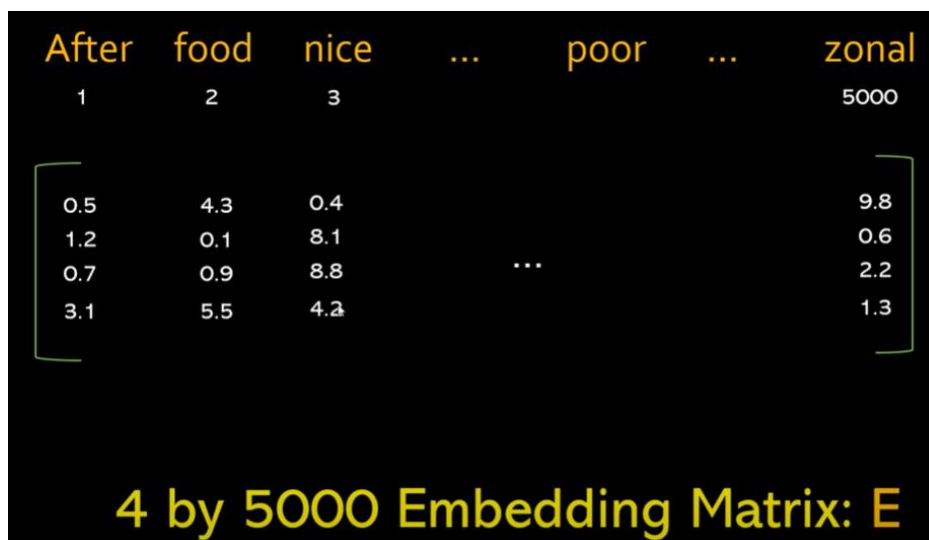


Ur goal is to come up with this word embedding vector with dimension 4

So this is something like trail and error method

We want a word vector which is of dimension 4

And these numbers will be calculated in such a way that similar word will have similar type of vectors.



So eventually our goal is to come up with 4 by 5000 Embedding Matrix: E

	After	food	nice	...	poor	...	zonal
After	1	0	0		0		0
food	0	1	0		0		0
nice	0	0	1		0		0
...							
poor	0	0	0		1		0
...							
zonal	0	0	0		0		1

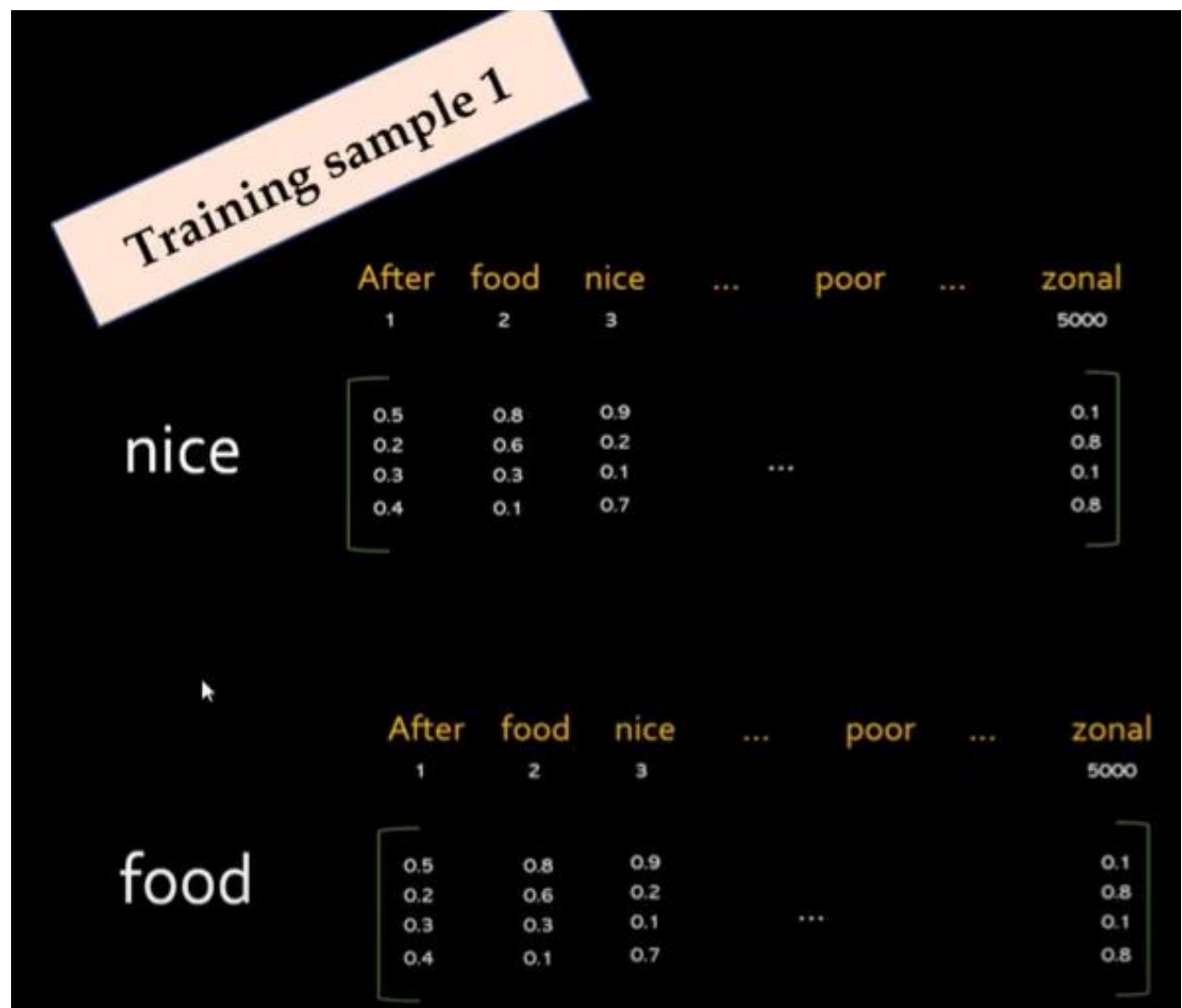
So the first thing we need to do is, we need to come up with one-hot encoded vector

So now u start training ur neural network



U first take ur first training sample, just to keep things simple

I am saying my review is on only two words 'nice food'. Actually it will be long 20-40 words



Then u come up with random weights in ur embedding matrix

Now u take one hot-encoded vector for word 'nice'.

Training sample 1

nice

	After 1	food 2	nice 3	...	poor	...	zonal 5000
	0.5	0.8	0.9				0.1
	0.2	0.6	0.2				0.8
	0.3	0.3	0.1	...			0.1
	0.4	0.1	0.7				0.8

 \times

0
0
1
0
...
0

food

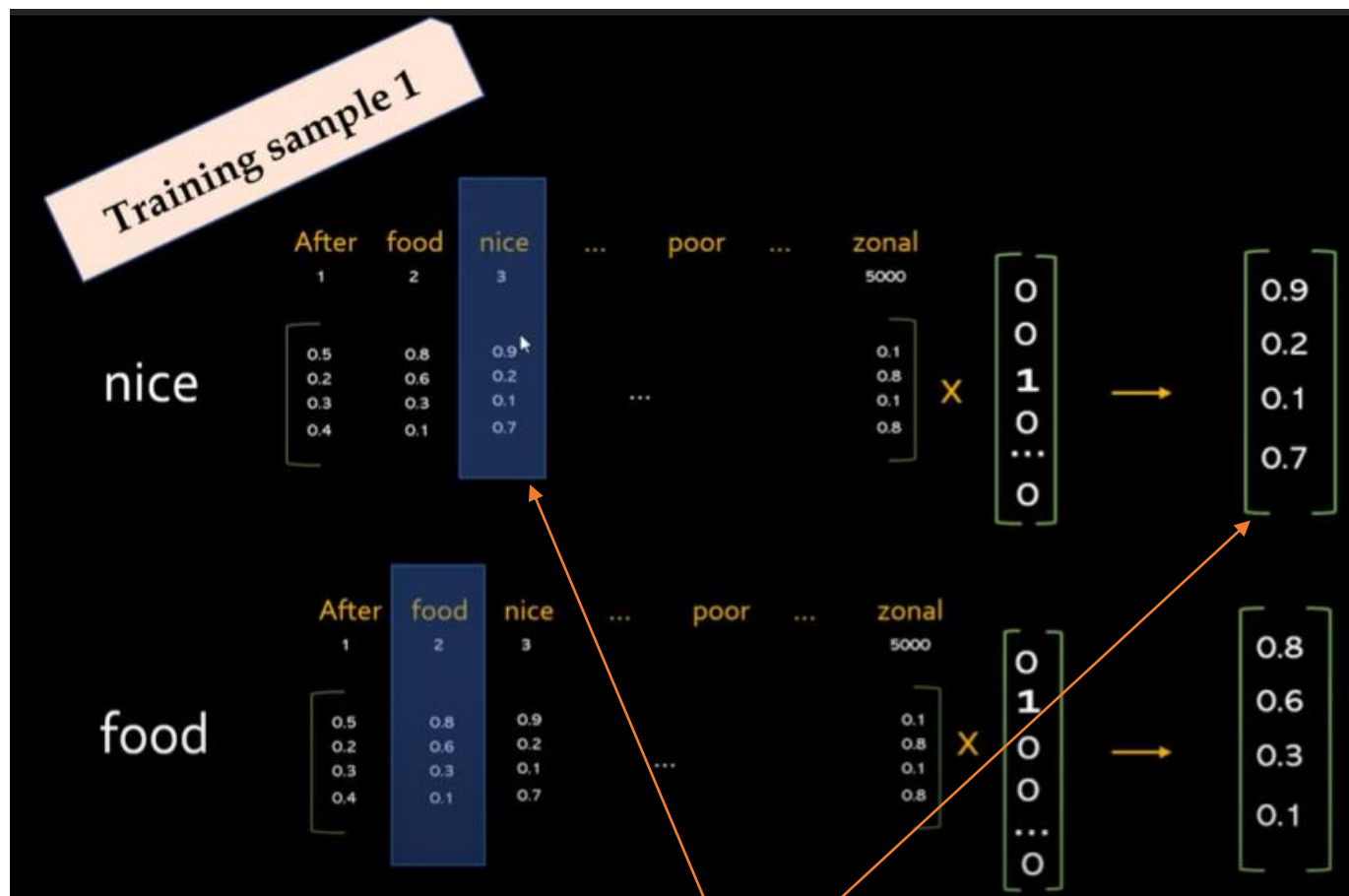
	After 1	food 2	nice 3	...	poor	...	zonal 5000
	0.5	0.8	0.9				0.1
	0.2	0.6	0.2				0.8
	0.3	0.3	0.1	...			0.1
	0.4	0.1	0.7				0.8

 \times

0
1
0
0
...
0

So u see 'nice' is a third word in embedding matrix , so 0010...0 in hot encoded vector

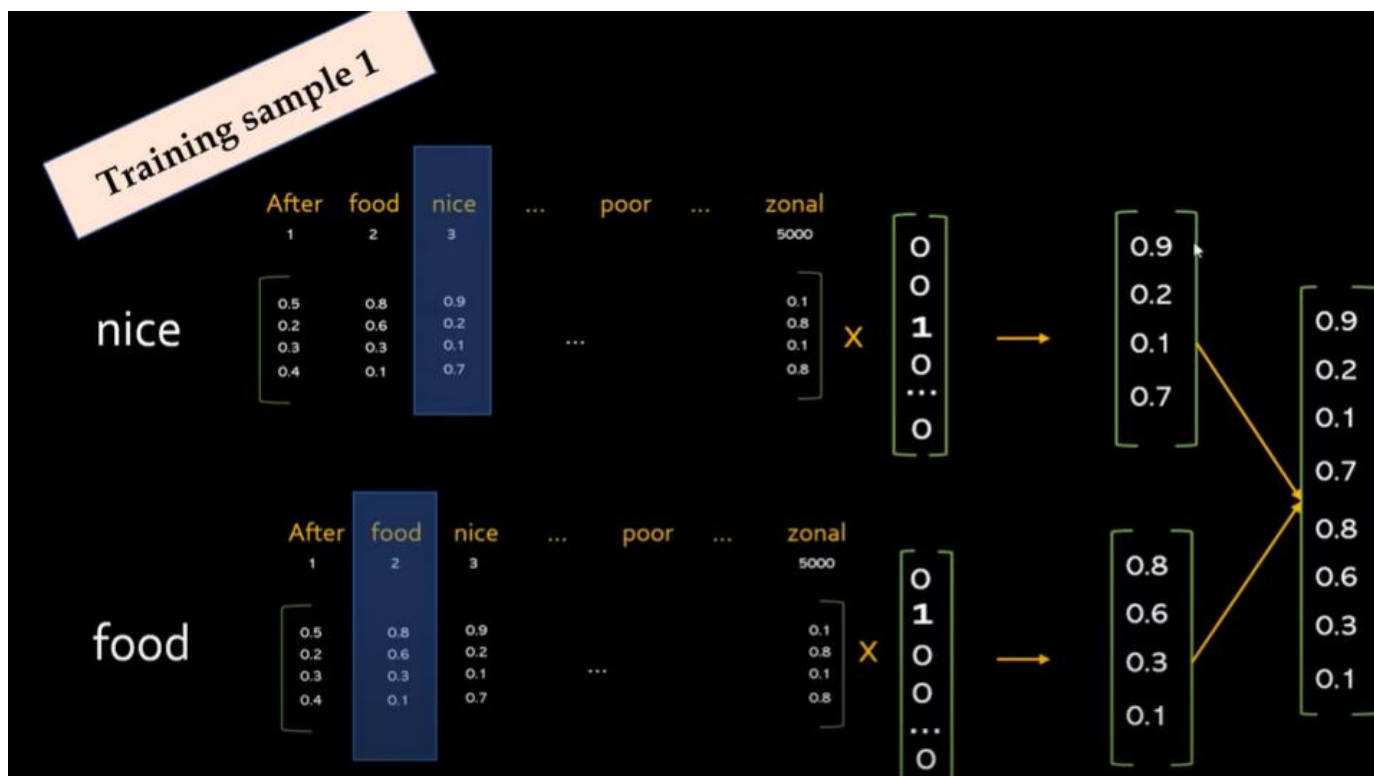
And u multiply the hot encoded vector with embedding matrix



So in matrix multiplication what will happen is u will get this particular column in result. Bcz in matrix multiplication we will take the first row of embedded matrix and multiply it with the whole column of hot encoding vector

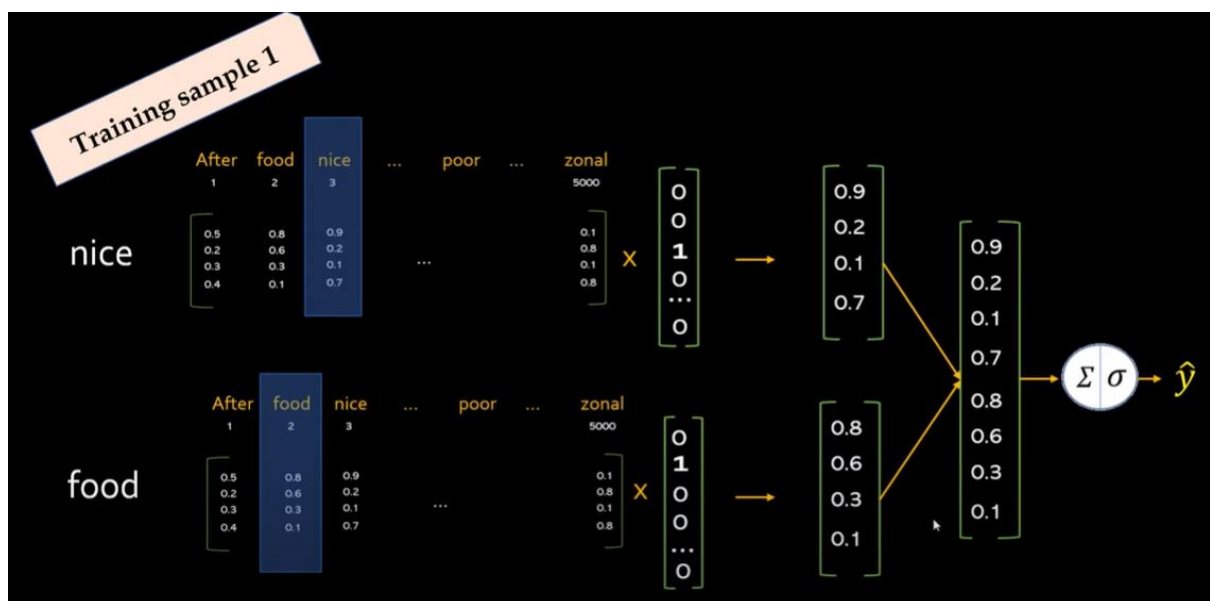
& similarly we will take each row of embedded matrix and multiply it with whole column of one-hot encoded vector

So in that way we will get the resultant vector.



Then u flatten the vector

So they were two individual word vectors for nice and food , u flatten them so u get like 8 dimension vector then u feed it to sigmoid function



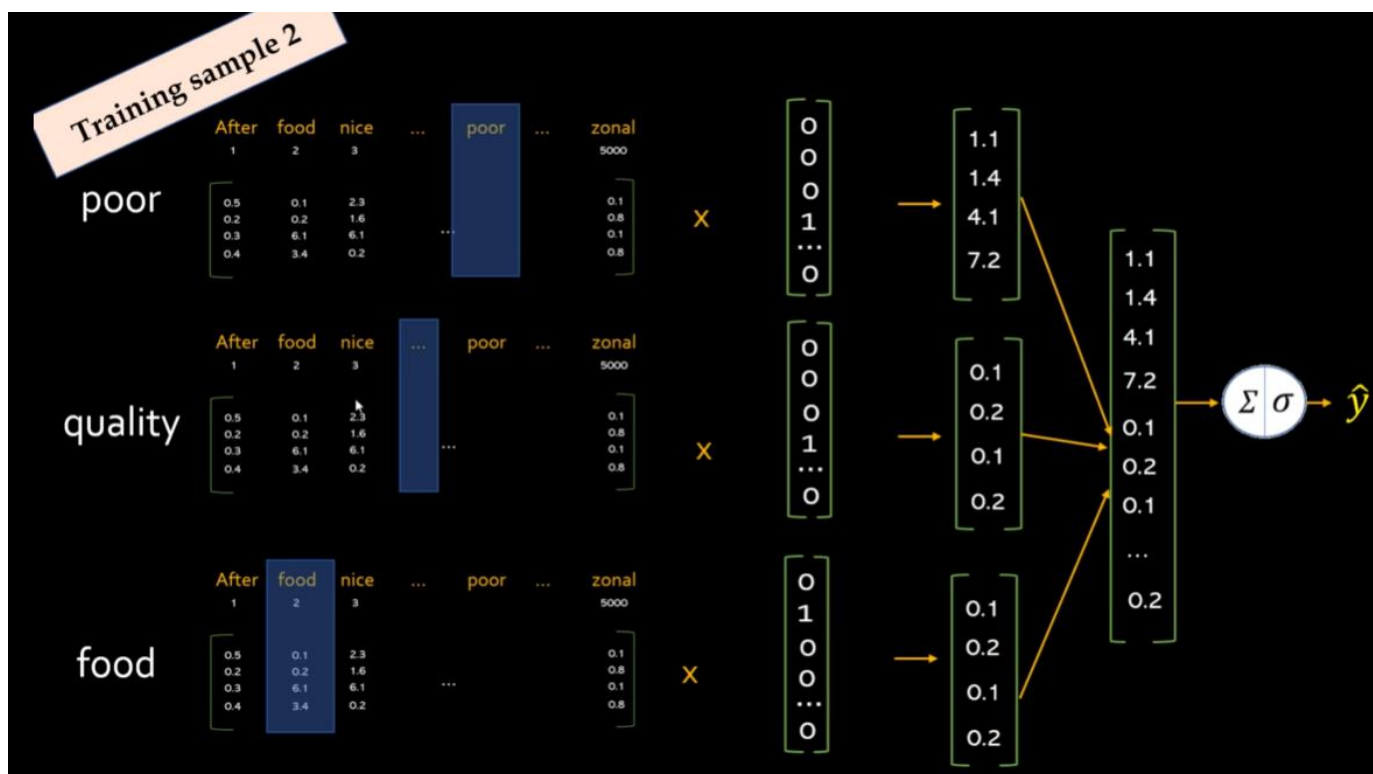
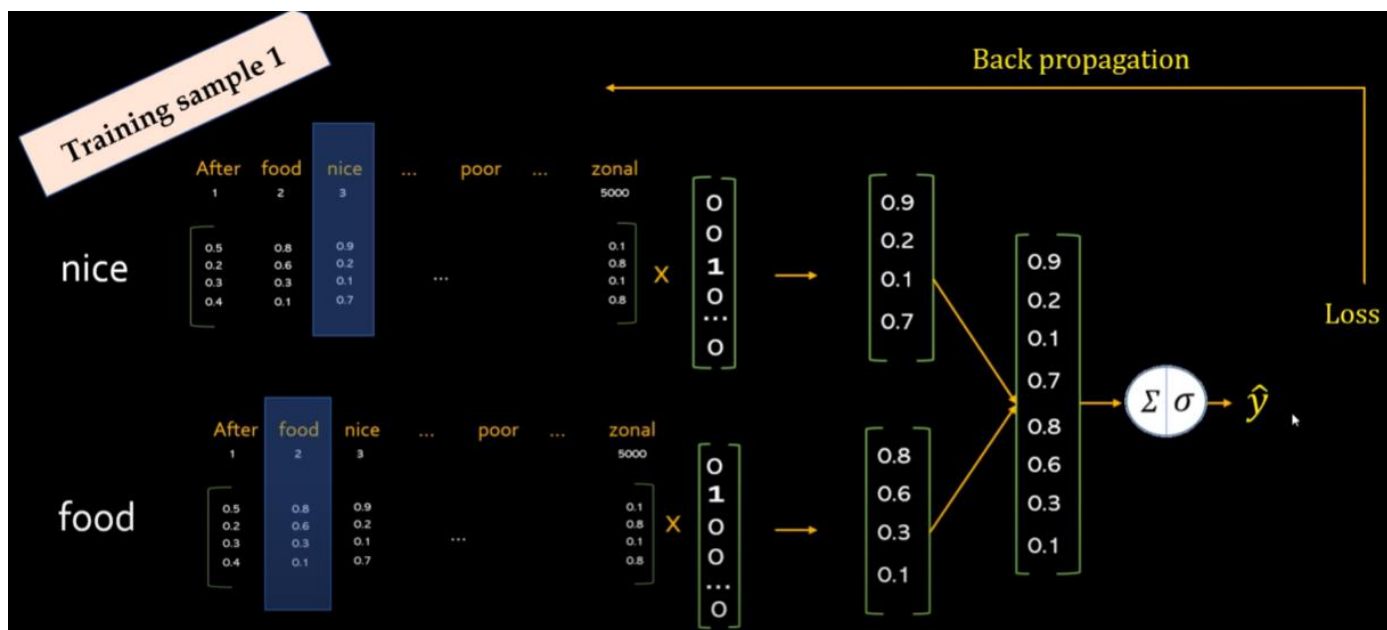
Its like a single neuron sigmoid function.

Which will give u \hat{y} i.e y predicted, u compare it with actual y , so nice food is positive review

So u compare \hat{y} with actual y which is 1

Then u get loss and u back-propagate that loss now

Now when u back-propagate that loss using gradient descent



Then u take the 2nd training sample, this review is of course negative

We repeat the same process, flatten out the vector, compare it the way we had, do back propagation and u change these weights

So this time u would be changing weights for poor quality food

Now a question arises ?

In the first training sample we had only two words and in the 2nd training sample we have 3 words.

So the neural network architecture needs to be fixed

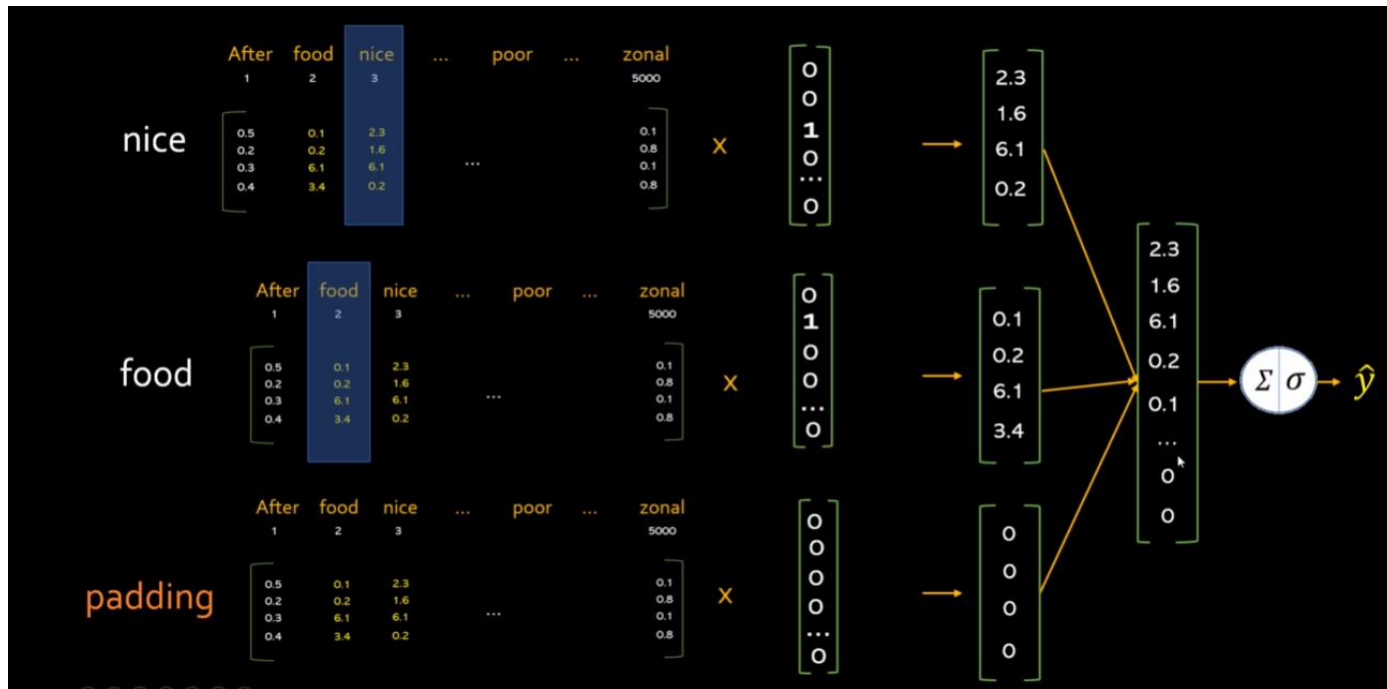
We can't change it

In the first training sample we had only 8 neurons in my first layer

In the second training sample we have 12 neurons

How is that possible

Well we take the maximum sentence size, and we do padding for the remaining words



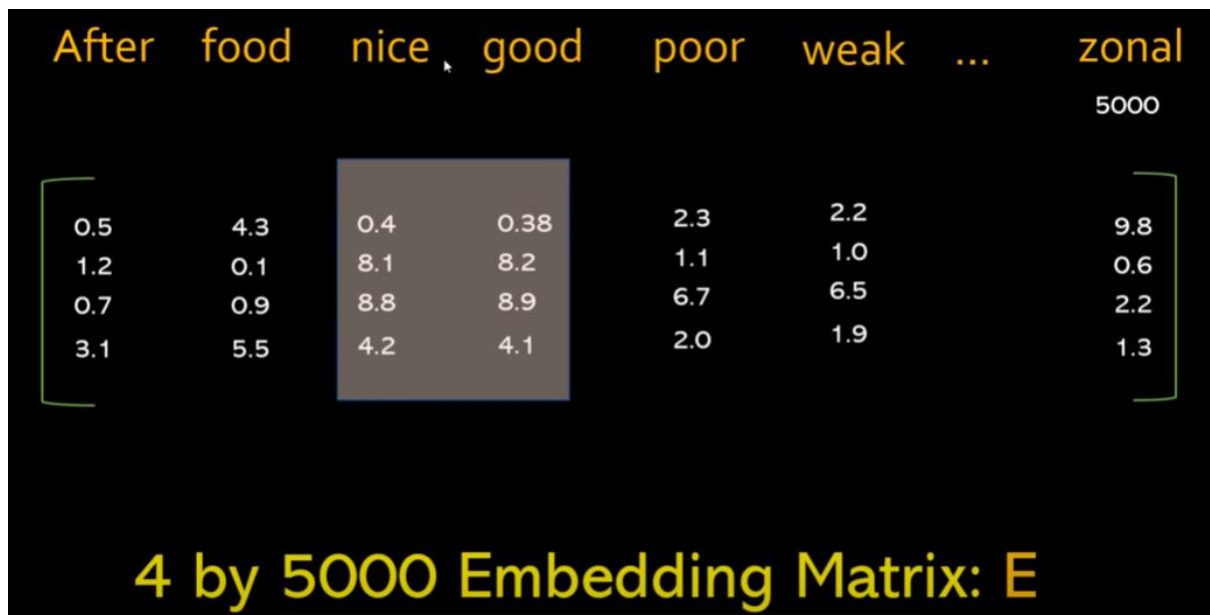
So let's say the maximum sentence size is 3

When I have nice food (only 2 words) we will add padding, so padding means all zero's, so all zero's we get

So that way my neural network architecture doesn't change, the first layer has fixed number of neurons which is equal to the word embeddings of the maximum length sentence.

& we feed all our samples, let's say 10,000 samples we keep on feeding it, neural network keeps on training.

Eventually it will come up with an embedding matrix, that can represent words nicely



So if u look at this particular example, nice and good are kind of similar words so the numbers are kind of similar u see,

0.4 0.38

8.1 8.2

8.8 8.9

4.2 4.1

So when u compare these two vectors u will see these two words are similar



Similarly, these two vectors poor and weak are similar

So this is a supervised learning approach for embedding matrix – this approach is not very popular nowadays the more popular approach is word2vec