Python and AI Power-Up Program Offline Class-20250822_191240-Meeting Recording

August 22, 2025, 1:42PM

2h 4m 19s

Mitesh Rathod started transcription

TJ **Tarun Jain** 0:30

OK.

The screen probably is visible right now.

OK, so here you have projector tensorflow.org.

The template.

OK, so the same example that we looked into last time. So you have king, queen or soldier and then you have some other animals. So every time, whenever you have to create embeddings, it's based on the schematic meaning, right? So.

Where is it placed? It's placed in a multidimensional layers. So here if you come back to the projector so you can click on any word. So this is all the words it has used to train this particular model. I've got to pick any one word.

And then just zoom it.

Now on the right hand side, if you see what is the word that I've picked Jem. So what are the similar words for it?

Diamond, diamonds, cocoa, mineral, alloy colors and then can you see this core?

0.627 that means it is 62.7% closest closest right? Kind of similarity score OK right?

And if you click on any other things now let's suppose you go to different dimension.

You're going to attractions.

Will you be changed?

So I just want to dimension in the sense every single vectors may can be in different dimensions. In simple words in neural networks and all dimension is nothing but a metrics vector.

Right, right. So now these two are in one vector. I'm just clicking on attractions or incorporated. So if you look for attractions, the first one is tourist. Tourist has somewhere around 0.10, 0.41, which is 41% similarity.

So now if you notice there is a keyword called cosine similarity. So this is very important when it comes to search, right? So how many of you know sine, cos and do you remember the values still?

Like what is cause of 0 and cause of 90.
And then cause of 180.
Yeah.
01.
Cause of 0 is 1, cause of 90 is 0, cause of 180 is -1.
So whenever we are dealing with cosine similarity, the range that you have right, the range of one word to another word will be between -1 to 1 if it is 1.
That means the given query and the document. So usually you will compare query with documents or you can do document to document. Is document similar to document two? Is query similar to document one? Is query similar to document two? Is query similar to document three? So you have.
To find the distance between them and when it comes to distance, cosine similarity measures angle. For example, if two of us are going in the same direction, what is the direction? If both of us are going in same direction, it's 0 degree, right?
So if two similar.
Documents are in same angle.
Which will have zero direction.
So what is 0 direction in cost? It's one. That means they're similar.
And if it is similar, it will rank. OK, you asked this question. I checked the similarity in all the 20 documents that you had and these are 5 documents which are similar. Now this top five, how is it decided based on that value and now you have 0.
0 means.
Probably you are in, not opposite, perpendicular. So perpendicular in the sense it will be in different dimension.
And what about -1? Both are in different direction and it doesn't make sense. They are not at all similar, so not similar.
Now, how do we find out like whether we are getting 10 - 1? Obviously we won't get one, right? One is same more than 90% you'll never get. That's for sure, because even if two documents are not at all similar, it will bring down the overall similarity score, right? So now what we.
We need to do is let's just try to understand how we are able to calculate 10 or somewhere between 0 to one, but the range of cosine similarity will be from.
-1 to one this is clear or probably you can add like this.
That means -1 is included and one is included.
So I do have some examples.

So let's start with step one. Here you have 4 documents. OK, the goal is to understand which two documents are similar. So you have document one which have few keywords, Ninja, adventure, fighting, power, friendship. Then you have document two, you have document three, you have document four. OK.
So I'll open Excel sheet on this sideline as well.
So in document one, do you have Ninja? Yes, so we just had one.
Fighting.
Power is also there. Friendship is there. Adventure is there.
And then remaining will be 00.
OK, so this kind of vector is what you call sparse vectors.
So there are two kind of vectors. One is dense vectors and one more is sparse vectors. In sparse vectors you will have zero and nonzero values, right? This nonzero values once you add weights, right? Because usually here we are just checking on document to documents, we are not adding weights.
Once you add weights it will be lower. It will be like 0.50.7 but those are called nonzero values. So whenever it comes to sparse right sparse vectors, you're doing keyword search, keyword to keyword search. But when it comes to dense, it is vector to vector search.
In our example, cosine similarity is dense. I'll show you why it is dense. As of now, we are just getting the count. This is not cosine similarity here, we are just doing count. But this kind of vectors is called sparse.
These two keywords are very important. You'll have to keep in mind. One is parse and one more is dense, right? And when you build the rag, you will have to use just vector database which is just dense. You might use both of them together, which is dense plus.
Spuds and both of them have their own advantages.
OK, now coming to document two, do we have Ninja 0?
Fight is there.
Power is there. Power is there. Friendship is 0.
Training will be one and super Sun is 1. So total you had four words.
And here Soul, Reaper, Sword, Spirit and fighting. So this will be 11111 and fighting. So this will be 000.
And here also I'll give 0.
0.
OK, now coming to the last example, you have pirate, adventure, treasure and

friendship, so this will be 0.
This is 1. Pirate is 1. Treasure is 1. Friendship is 1.
OK so so far what we have done, we have documents, we took the documents and we converted that into sparse embeddings, sparse vectors. So here sparse vectors is nothing but either it will be nonzero value or it will be 0 value. Now our goal is.
To understand what is the similarity between doc one and doc four. OK, so here is where you have the formula of cosine similarity.
So how many of you know what is dot product?
First you will multiply element wise, element wise and then you just have to add it right? So if you see your AB that means your I is nothing but element wise. Once you multiply all the elements you just have to add it.
So this summation is for add. Is this clear what this notation does? So this this thing is the right. This is to add and whenever you see I and then if you see the I repeated, that means you have to repeat it for element wise.
That is element to element. Now does anyone know what this symbol is?
Not mode. Mode is very close by.
We had this in distance measurement, so if you see here one is cosine, one more is Euclidean. In Euclidean we had magnitude.
So this is nothing but magnitude. So what does magnitude do? Let's suppose you have VA which is 1111 and then 000. You just have to Add all of them square, so one square.
Plus one square, plus one square, plus one square, then zero square, zero square, zero square. So now how many squares do you have? What is 1 square?
One and I have total 4 squares. I mean I have total 4 one. Everyone square is 1 + 4. So now sqrt 4 is.
Two, so that is magnitude. So you just have to do the square root of all the elements square. OK, so element square, square root. That's it. So if you see a A is there, A is a list, so you have to go element wise.
For every element you have to square, then add it. Once you add you have to take the square root of it. So understood the formula you just have to do.
So only these two you things you have to remember. One is summation and one more is square root. So what library will I use for square root?
Math dot SQRT.
So now I'll take document four and document one. How many ones do I have? 12345, right? So let's simplify this. Is this the same thing? 12345.

And then document for us.

1234.

So now if in case I have pirate adventure, treasure, friendship, treasure.

Right. During that time, this will become two.

One, then this will be two. So let's suppose I have pirate adventure, treasure, friendship, treasure. So basically this last index was treasure which was one. Now this is 2. In our case there are.

In the same sequence you don't have two words which is repeated. So you might think OK sparse vectors is a zero or one, which is not right. Sparse vectors is 0 and nonzero values.

So this is one. I'll just remove this. Now what is the dot product?

I'll do cosine of doc one comma doc two, sorry 4.

Then the first thing is I need to have a dot product.

The above is well.

Yeah.

No. So if you see 10, this is 010.

So there are actually two, so it's like 1.

So first we have 0000 plus 0 + 0, then plus 1 + 1 + 0 and then till zero. OK, so now the dot product is.

Now you have to find the magnitude magnitude of doc one.

So it will be sqrt 55.

Somewhere to point.

SURT of five.

2.2 and then you have magnitude of dot 4 which is very simple sqrt 4 which is.

Two. Is this clear? Yes. And now you just have to divide this.

2.23.

Multiply by two. So how much will this be 0 point?

0.448 I'll just add 0.4. So now this is nothing but 40% similar. I'll just add 40 plus.

So this value, whatever you see here, right, is what you're seeing it here.

But it's not happening on just two documents, it's happening on a bigger metrics. So what is the drawback of sparse vectors?

The more the documents you will have lesser value.

Do you guys remember? I told one word curse of curse of dimension, right?

Ajay Patel 16:29

For some dimensity.

I have.

Tarun Jain 16:34

Correct. So here in sparse you'll have it will this number will go on, right? You'll have 50,000, you'll have 10,000 if your vocabulary is 10,000 and if you are creating cosine similarity for 10,000, it's huge, right?

And if you look at the other way around, have you seen dimensions in dense vectors like Open AI? Have you created Open AI embeddings as of now in any product? So by default Open AI has.

Open your embeddings.

Dimensions. You just have to check DIMS. DIMS is nothing but dimension. It will be 1536.

OK, so now what is this 1536?

It's these dimensions.

1536 it's fixed. When it comes to dense, it is fixed and it will be random values like 0.530.54. There is no nonzero in dense, so the difference between.

Dense is it is fixed dimension.

And what point I said it is fixed dimension and nonzero values.

Rarely you will see.

0.

Now in sparks.

It is fixed but also varying. It's not sure you're not confident about whether it will be vary or not, but in most of the cases it will be 0 and.

Non 0 values.

Out of which 50 to 40% will be 0 only. So here also you saw it how much zero we have.

We have so many zeros in real world. Also when you are working with sparse, if you have a dimension of 10, I mean 2004 thousand, you will have just 200 or 300 which actually has some values which we will see in the code demo.

I'm just saying like for spars, is there a different set of dimensions? How they are using open EA doesn't have spars, so usually all the embeddings that you see are only dense. OK, so spars are just limited and these are algorithm based.

OK and uh denser model based you train it.

And sparse is data based based on what new data you pass. Sparse is dependent on

that. So basically open AI and other LLMS other embedding model, they will not allow you to add your data to modify the embeddings.
So you can also tell dense you can modify. Sorry dense you can't modify. If it is already trained, you have to fine tune it to modify, whereas parse you can modify based on your data.
But.
This one, yeah. So here's the like the we have get it thought right like the one into 00. OK, how we get like one under 4th position in the one in the 5th position, one in the one one. Yeah, one into one is like, but zero into one.
So 123.
Then four and five. If you see four and five is 11. So it's like this.
So look at the formula, right? So in formula if you see.
It's OK, OK.
OK, so everyone understood the logic how this scores is calculated. So this is step three, right? And this is what you call us pair wise.
Similarity which is cosine. Now in four step what you do is you create a metrics right? If this is a now this is fine for one to one. Huh. So this is just one to one.
Now just imagine you're building recommendation system.
OK, so you have a query which is ordered.
Now what you need to do is this query has to go for all the documents. You can't do query one document, query one document, query one document. So you need to.
You feed the data, get me the metrics and then I will check which metrics is close. Then I can sort. So instead of doing 1 to one, you can do 1 to one to all the documents, create a metrics and then next time whenever I ask any new question, I will create that metrics, sort it and get the top results.
Right. So now if I give order and if I have 20 documents.
What will I do? I'll create 20 cross 20 metrics. So this is D cross D. How many documents do I have? 20 here 20. So here if I create a metrics for this if I create a metrics.
How much will it be 4 cross 4?
Just a tricky question here. Let's suppose I create 4 by 4 document. You have document one, you have document two, you have document three, you have document 4.
What will be the diagonal?
What will what will be the diagonal? It's the that what we call it identity metrics. If we

just ignore everything else 111111 correct. So this diagonal is there right? It will be 111. Why? Just do doc one of doc one similarity.
So if I just repeat this 12345, I'll get five. This is five. Then sqrt 5 is.
2.23 and below that 2.23 so square root of 5 square root of five. It's five and then above you have one. It's sorry 5 / 5. You have one, so your entire diagonal in cosine similarity will always be 1.
Because obviously doc one will be similar to doc one, right? So this is 1, this is 1, this is 1, this is 1 and what is this 104.
0.4.
I need to.
Not identity matrix completely because this value will be different right? That will be the same. So identity matrix is like 101 then 010 then you have 101.
101 right? So here technically you'll have a different value. Even if you have 0.2 then two and one. If you keep it 0.2 over there, this will be same. Yeah, but this two again will change. These values are changing.
So it is. You can say the identical matrix, but it's not completely. So identity matrix will have either zero or one only. OK, OK, there won't be different values I guess. OK, I did wrong. This should be 0.
This should also be 0.
Only diagonals are one.
Three cross 3.
Only diagonals will be 1, remaining will be 0, but very similar when your X&Y is always similar in that direction, nothing different. Sort of identity metrics, but it will be different.
Is one as per the.
It follows the diagonal is 1, but the non diagonal values are nonzero. So identity matrix is the non diagonals are 0. Here it's not zero. This will change right?
So you understood the step one. Step one you have document. Step 2 is the formula which is very simple. Step 3 is distance measurement which is pair wise. The best word is pair wise.
Wherever you see cosine, just relate the keyword is. And now what is this? This is parse vectors and once you have this entire dimension, it's your dense vectors. Sorry dense.
Once you have everything, but you understood the flow rate, how the breakdown is done.

So like why did we created this distance measurement? For which one? Why did we created the distance measurement metrics?
Yeah, let's suppose you're building a recommendation system one-on-one. If you're doing it, it's fine. But what if you need? What are my top five that I want to watch? So during the time, if you have a metrics of all the documents, it's easy for you to.
Oh.
Create the top five, top ten. Like what would I do with this matrix? So let's suppose this is your query.
Right. So with the query, what you're doing is you have all your documents. OK, the OK, I get it. OK, now yeah, once you have that query, you know, hey, which is my top five movies and it it usually won't be top five. Huh.
But we'll just sort it out and correct. OK, so now you'll say OK, what 3-3 also I'll just use it for top three. But what if user change its top K to 10? What if he changes to 20? What if he changes to 25? So you can create that metrics at once.
Then you can let user to pick any K value in it. So usually K will be predefined. But if you're building a system where you can give a filter, hey I want my top ten, I need my top 20. So during that time you can just filter it out.
So this is your indexing. Once you have a metrics you can index.
Index is like you start with one and then how basically you know if you frequently what product you get the recommendation at below correct. So recommendation system also uses. There are three ways. One is you have content based.
And then you have collaborative way.
And then you have hybrid. So in most of these approaches you use cosine similarity.
I just built one astrad only two days back.
So this is one more search technique. So whatever you saw now, it's just that relevant. If it is relevant, you're getting it. But what if it is diverse? So what we saw is this kind of search.
You ask Indian food for first time visitor. It will do keyword search, it will get you search. But what if you need diversity right? So I need this one XYZ. For diversity you have something called as MMR.
This we have in rag, but I'll just show first what you need to do is first you should at least have normal search. Once you have normal search then you apply MMR to rank it right for normal search. What are we using here?
Cosine. So by default most of the rag or vector database that you use will have distance and you'll have only three options. Either you can pick Euclidean or you can

pick cosine, or you can pick something called as ENN approximal nearest neighbor. So these are few of the distance measurement and the default is cosine.
Is this clear why we are creating metrics? Then how are we when creating the values similarity scores value? The formula of cosine is very simple. OK, so now we'll go with the code implementation. I have two code today. One will build a recommendation. System. There is a IMDB data set which has around 250 movies. So you ask any keyword and then which is the most nearest we'll print it. But that is not production grade recommendation, it's just to show how cosine similarity works.
And how PFID works?
So I'll just show PFID of math as well.
This probably I showed in the second dev itself when we discussed math.
So if you remember when we did import math, the first thing that we tried was import log and when I showed import log I said we will use that in TF IDF right? So now TF IDF TF is nothing but.
Term frequency.
So what is term frequency? We already completed that. So given a word, what is a word here?
So the formula of TF is TF.
T comma D so that is TF of ninja.
In document one.
What is it?
It is 1. So whatever you did so far, it is TF.
OK, So what next step does is TF also has its own limitation. What if there are too much of similar keywords? Then how do you rank right? During that time you also check their probability likelihood of it, right? So this is where they introduce something new called IDF.
So far what we did was TF.
So this is TF. Whenever we are building recommendation system, we have a data, we are creating a term frequency of it. This is fine till here we know, but the advancement for that was IDF which stands for term frequency.
Inverse document frequency.
So basically now what we are trying to do is you created the term count. Now we will apply weight to identify the likelihood of it in the entire document, right? So for that likelihood, the formula of IDF is.
IDF of T Now what is TR?

What is tea?

I mean here.

So instead of.

T is nothing but term.

Ajay Patel 32:17

Come.

Um.

Tarun Jain 32:20

Or you can also define W as word. So this will be either T or it will be W. So this annotation changes, but the formula is same. Do I have the formula here? I have it. So TF is nothing but T total terms in the D which is document and count of the T in the document, right? So how do I calculate it?

PF of Ninja.

In document one, how much is it? 1 / 4 four or six? I guess in first it was 51234 OK five.

If you appointed.

Zero point.

So we will break this down now and then we will cross check it with the once we write the code.

Ajay Patel 33:14

.2.

Tarun Jain 33:22

0.2.

And now you have IDF IDF is to find the likelihood the probability in the entire document.

So see IDF equal to 0.2 which one?

Oh no, IDF that OK, so IDF we applied log. Now what is N here? N is how many documents do we have? How many documents do we have? Four. Four. And what is the DF of T? That is nothing but how much time it has occurred.

How many times it has occurred? No TT. What is T Ninja One?

What is DF? DF is a derivative function. No DF. It's. What is it called?

Ajay Patel 34:17

Differentiation.

Tarun Jain 34:22

Document frequency. Document frequency of Ninja in that document. Which is 1 OK. So now what is the that is the one that we calculated? No, no, we didn't calculate. What is log of four of one?
OK, this will be very difficult. Uh base with base two it will be two. Uh base two it is 2 right? But it's not base two. Uh then what is the base N? This is log of N.
No, no. But the base will be 10 some, uh, this is binary log or what kind of log are we looking at?
In some for 10 decimal you're looking at decimal so you'll find 6. So one second if I do import math it is very important. I just want to check what this Python folks are doing log off.
So technically log of 10 should be 11, log of 100 should be two.
So this won't be one, I guess it's 2.02.3. So if I make 10 now this is 1, right?
This we already checked log E huh log EI was adding in but it's log E.
01.
What is E? Exponential.
The value one I forgot, but by default it's either log N or it's log. It's not log of base 10 OK.
No, it is not E we are looking at right now. So now what is the formula here? It will be log of four is.
1.38 So what is log if you guys are not aware?
The base value that we're talking right is 2. How many times would you power that to achieve that number? OK, so if you have a base 10, how many times would you for log of 100 for base 10 is basically.
You will do 10 into 10 at the power two, so log of 100 at the base 10 is 2. That is what log is.
This entire thing.
Six into How much did we get earlier? 0.2.
And then 0.2 you're like right.
0.040.27 So this is your actual value when you will work with then uh sparse embeddings.

So what is TF IDF of Ninja?
In document one.
We did log N divided by DF. So we are doing that log IDF before the IDF is 1.38 and then TF IDF. OK, so TF IDF is like first is term frequency, then we have IDF, then the actual TF IDF is you just multiply it.
So 0.2 into 1.38 so this is your actual sparse embeddings value. Now if I do ninja of document 2.
0.
Clear, right? What is here? What will be your count?
It's 00 so here also if you multiply 0 so you don't even have to calculate it. So now what is happening in sparse? Either it is non 0 value which is not exactly one but it will be some decimal value or more than one also.
Right. So the only thing is don't confuse one or zero because there will always be weights associated, right? And then if you use any libraries, after this they will apply normalization.
So if you're learning regression right machine learning, you will have L1 and L2.
So these are like penalties that you are adding for your document so that you can use some people. What they will do is not some people, I mean researchers. Let's suppose you got 0.27. Repeat it for the next word. What is the next word?
We are training training. I'll put document one. It's again 1 / 5. This will also be 4 / 1. Same value right? So now what you will do is you will add all of them and then you will create a square root of it. That is your L2 normalization.
So what L2 normalization will do is sum of square of all TFIDF values in the document. So this is applied in frameworks, but as per the logic of TFIDF it ends here. Is this clear? This is clear. So either you'll have L2 normalization or you'll have L1.
Or to bring down the bias?
So this is typically it was supposed to be used only for ML algorithms on neural network, but since SK learn is mainly used for machine learning, they applied that for all the algorithms, so predicted values need to be closer to.
I feel what is what is going to be the target when we want to get as close as possible. So it's like the goal is let's suppose you're building something and here is your predicted value.
And.
Here you have your actual value. So when you're building the models, you usually want this value to come lower, right? So either you add some kind of penalty, do

square root or in L1 normalization. What you used to do is probably you multiply by 0.1 something.
To the weights. So these are some tricks to bring down the values so that you don't do over complicate, I mean over multiplications. So all these are compute heavy.
Now we will see like once we upload our data.
Huh. This is still child what we're doing, but it will take too much of calculations. Today when we save our data right, it will literally take one to two minutes just to save and it's just 250 movies. Building recommendation system on 250 movies is nothing. There are people who will build it on file lags data. So you can just imagine how long it takes just to save the data.
So that is fine because this happens on offline document processing, right? Once you process it, you use the save data and then you do the inference.
But these two concepts are clear. Yeah, just a question will be out of context.
If somebody has multiple likes.
You're not just searching for one recommendation. If you correct multiple like based on that multiple likes, how do you get a new recommendation? So again here the search algorithms will differ again simple cosine similarity. It will just give whichever is similar to it.
So now even if you look at that this thing right, let's suppose attraction, attractions can be different, but still tourist has some value, right? It doesn't make sense why tourist has so much of score values. So based on whatever documents you have.
If in case there are, let's suppose we both have some common similarity and now if you're asking order plus comedy, you'll get some closeness value, right? But if there is someone who has never seen order plus comedy, you will get wrong responses only.
So that will only take one consideration. So that's purely based on keyword and the documents that you have already seen. But if you need diversity, there are again different search techniques. Search is not just one search. There are people who uses graph.
Databases. So graph is again a different search. Google uses graph page rank, so there are different search techniques. Page rank is there. So that's why many people they just build LLM's for, but the real game is in search.
How well you can improvise search?
And this article was also purely for the.
So we actually looked into two parts. One is cosine similarity and one more is TFIDF,

right? So using these two concepts, what we will do is we will build a recommendation system. But before that, let's just understand how to create embeddings.
So do we have this notebook?
You might have to leave for 10 minutes and 5 minutes, OK.
Oh, so the code that I have.
So basically what will happen? I'll just tell in template code, we'll just define some model and we will convert that into embeddings, right? After that, probably whatever we just saw, I mean these four documents.
We will use some code. For example, these are the four documents that we took. We will define TFIDF, then we will define cosine similarity.
Here we have cosine and then we are checking that score. So this is what we will do. Whatever we just saw, we will cross check with code.
Because here recommendation system I've added in functions so we might not check this line by line. So only this one is to debug it. OK this is the actual code. So while you are away we'll just play around with this thing.
But this is the same code here. We'll use Excel data. Here is what we actually did just to cross check how close we were.
Here is actually what we will do. So probably we've once you come here it will be better.
Can you see this line pip install fast embed? Are we still in the Google meet? OK.
I will just close the unwanted tabs, but anyone has any questions in this one?
No question Sir. This formula is very simple of TFIDF and as well as cosine similarity.
OK, so here we are using a library called fast embed. So basically every single embedding model you have right has their own name. Let's suppose you have open E end. Open E end have their own embedding. So anyone remember the name by any sense?
Small text. I love small, right? So you have open it. What is the element? GPT, right? What is the tokenizer?
Let's suppose CPT pipe to or to or to and K to basic that is tokenizer. Just like that you have embedding as well. So embedding is text add a small. If not it will be text add a medium text add that.
So it's a model. Same goes with right? So if you want to use the embedding model, again you have to use API key of this. So usually these are the only two things you would use. One is API and one more is embedding. The API of a little API of

embeddings.
Here what we are doing is we're using fast embed. Now the reason of using fast embed is AVC argum phase library. In argum phase library there are tons of models. These are open source models. They are running on GPU.
They're running on CPU, but if you want to reduce the size of it, there is a framework runtime called ONX runtime, right? ONX runtime, what it does is it will have data parallelism. So when you are creating embeddings, obviously you'll have multiple data, you need batch to process.
So you need to have some kind of runtime which will reduce the time of it. So this is where we will go with fast run. So whatever models you have in ID phase which is open source, you are just converting that into ONX runtime which is running.
You know, lesser speed, right? But the performance is same.
This is open source circuit and what was the leaderboard name? MTEB leaderboard. So if there is any requirement for you guys that OK you need to build an entire text stack or rack or AI model.
Just by using open source what you call technology, LLM's are sorted. Most of we know Mistral, we know GPT OSS. But what about embedding model? And even if you pick embedding model, which embedding model to pick right? So the safe bet is go with JINA or Nomic.
Right. These are a very good players in the embedding space. Why Zena? Because Zena.
Shift between the Gina model, Gina embedding models and the second one is Nomic right? And you will also find some of the embedding models here like when.
The reason why we don't go with Chinese models are there are too much of Chinese tokens within the embedding model, which sometimes causes issues. And not just that when it comes to GD based models, right? If you see GD, that means it's Chinese models, either it is GD or it is quite.
The issues with this models are either there are more tokens. The second issues are they're overfitting. So it's like they already know the data, they train on the data and then they use it on later code. So that's the reason why they always have a high benchmarking score.
Because they're overfit generator. So most of the models that you see here in beginning, beginning, it's only Chinese. Now when you see Gina model, Gina, very legit, Gina, right? So can you just search for?
Fast embed supported models.

Can you see this page?
So you can use ugging face as well. Ugging face has a library called Sentence Transformers, right? The only better thing with ugging face, this thing is if your entire system is working on GPUs then you can go with ugging face because it has very good support with GPU.
But if you're running it on CPU then you can use fast embed. So here there are some of the supported models. 512 is not that great. You can go with 768 dimensions. So this Gina embeddings, but this is for German.
You can pick this model. This is the model that we will be using. Can you just copy this?
It's at the 15th Uh length.
What 52? OK, so either you can use.
Jina or you can use Nomic. Do we have any better model than Nomic?
No, let's just use Gina. Now if you see there is also one more model called here base code. That means this particular model was mainly built considering if you want to use it for code, code, and here you can.
It is used for multilingual and here if you see it's not multilingual, it's only for image, right? So you can't specifically use if you're using it for different language, right? And here you select Nomi.
No, it is for multimodal and we just have to check if it is for multimodal or not.
So this one is multilingual. If you see the last one in float multilingual E5, you can also use this model.
We will go with Gina AI. Copy this, come back to template.
But first we have to install it. What is the command? Pip install fast embed.
So what will be the dimension now?
What is the dimension of Gina Gina? So as you can see dim either it will be dim or it will be dims at the end of the day it's dimensions and it is 768.
And here you can just add model name equals to whatever you have copied. Just cross check if the name is correct.
Give a message.
OK.
Please.
It shared the match check match check match check. We are still not doing. We'll share it shortly.
But let me share it.

Match check. We don't have to create a copy of it, that is just to cross check whatever we did earlier.
OK, so now what we can do is this is model name for dense.
Now the model name for sparse is.
Parse model name.
Quadrant BM 25. So BM 25 is a ranking algorithm which is a again ranking keyword algorithm.
Just cross check if you are on the same lines. So text embedding is for what kind of embeddings?
What kind of embeddings?
Dense and then sparse text embedding is for dense, sorry sparse and this is the model for keyword related.
And why is parts used?
What kind of values will sparse have zero and 0 and nonzero values?
What about dense fixed?
And nonzero values. Rarely you will encounter 0 values, 0 not value 0 and sparse can also be fixed, but in some cases it will be varying. So it is fixed plus varying.
Now we defined model names. Now we will define embeddings dense.
Embeddings equals to.
Text embedding whatever the auto complete is and then you have a variable called model name equals to dense model name.
And then you have sparse text embedding model name equals to sparse model name.
So it will ask you for adding some HF token, but for this open source model we don't need so we can just click on cancel.
OK, I guess beyond 25 should be in small letters.
So is it downloading? Yeah.
Can anyone recall what might be inside this file?
Configuration of special tokens and vocabulary.
No, I mean it will be within that only. It will be in this one, yeah.
So are you able to also download this TARTXTTXT, Dutch, French?
Are you able to see this as well? Hungarian, Romanian, Russian. So that is for.
From this line right fetching 18 lines. Whatever you see below this is your BM 25.
Whatever you see above which is fetching 5 files. So your JINA has only 5 files whereas VM 25 has 18 files.

So what have we done so far? We defined the model name for both sparse and dense. We instantiated the models. Now what we need to do is we just have to pass this text.
So we need to pass this text and check what is the embeddings of this.
I'll just check.
Text embeddings.
Or you can also have convert dense embeddings.
Equals to.
Dense embeddings. Then can you just write dot?
And here you have embed.
So what is the input for embed document?
You can do batch processing as well.
It is documents. So what you can do is you can just pass text. Why?
It's a document, but can you check this part? It it rebel. That means you have to pass a sequence which has string list is also tuple also.
Now what I need to do is whatever output you're converting, I just want to convert that into list.
And then do length of.
Convert dense embeddings. So what should be the length of this?
Bye.
Do we have four or five? Four.
And now if I print this zeroth index dot.
Shape.
What will be the shape of it?
If I do 0 index shape, what will be the shape of it?
768768.
It will be 768.
Now you can also print it. Shape is shape and dimension. What shape and dimensions? What we get in the output dot shape shape will print you dimensions OK.
So if you remember love when we work with tuples right, I tried with an image. I uploaded an image and I wanted to check the shape of it to get width, height and channel, but the command was size. So here for if you want to print the dimension. You need to get the shape. Same thing. What you can do is you can use any other model, copy this mix, mix bread, mix, mix bread and if you had it it should be 1024.

So the default values are it starts with 368.
Sorry, 384.
I guess Google also has 384. I mean Google by default it is 384. Then you have 512768 is best. Then you have 1024 and then you have 1536. Who has 1536?
Searching previous is open AI.
There is also, but this is usually the best sizes.
7384 is fast, but if you have very big data, 384 is not ideal, right? So somewhere between 768 and 104 are good values. I don't know why Open AI went with 1536, but for them it works well.
The and.
Yeah, wedding large dimensions is something that can be changed.
Yeah.
Text embedding 3 large default dims.
Default embedding is 3072.
When did they launch this?
With the OK cost is also more, yeah.
So even though you are seeing it as a 0.13, it should be 0.0 when it comes to embeddings. 0.1 is huge when it comes to yeah, open, open AI. Even if you see 0.1 somewhere for some limited thing, it's a very huge count because it's 0.02.
We do compare.
This is the model that we usually use, which is Ada 002. It's the same.
Daminci is very old. Daminci is updated.
So till here we are done and can you also print?
OK.
So now this is in one dimension in Excel sheet. Also it is in one dimension, right? So let's proceed with the second model. So that's it. If you want to convert your embeddings, define uh, I mean instantiate.
Give the model name and then if you want to convert it just use dot embed and if you want to use batch there is batch function as well.
Fitted type of the. It will be array, array object. I guess it's uh ha Numpy dimension. So Numpy we were supposed to cover, but we'll come to Numpy. So Numpy is nothing but numerical Python.
And this is nothing but array.
So numerical Python is same arrays are there, you're just adding that in a object. So numpy is an object here.

No, it's array only. OK, so this is package. So it's like import Python.
It's a code Python library. So now here numpy is that right? Whenever you work with arrays you will use numpy right? And most of the frameworks if you're dealing with tensors.
Tensors is nothing but vectors, so you can call tensors, vectors, metrics. All three are same, but the only thing is objects are different. For tensor you'll have tensor which can be Tensorflow or it can be Pytorch.
For array either it can be a common array or it can be numpy array or people will call it as list right? But the difference between list and array is arrays can't store multiple mixed data type.
OK, list can store multiple mixed data type. That's the difference between list and array. OK, Numpy will come. Actually Numpy was supposed to be covered once we completed Python, but since I'm in person, I thought this is similar to Python. We can push after NLP.
So this is numerical Python. ND is nothing but.
N dimensional array and it can be 1-2 or three.
So let me check if ending works.
This is 1 right? Now if I give an array which is a equals to numpy.
OK, wait, we'll take Numpy later because we have more code today. So the next thing is we have to do it for sparse. Can you write it for sparse? The logic is same.
I don't see.
Shape won't work, right? Can you tell me the reason?
Because in sparse you'll have nonzero and 0. So basically for nonzero it is targeted as values which is indices.
So the first line we can write convert sparse embeddings.
Equals to list of sparse embeddings dot embed text. This is the same syntax. Now what you can do is you can print it for zero index.
You will have values. Valis is your actual non 0 values.
So if you see it.
You have the indices at this particular indices. You have this value remaining indices as 0.
Now what is the shape of it? It's just five or six. That's it. It's a platform.
Which one? Uh, you have to use dot values.
dot.
dot values is your actual embedding data.

And what is the shape of it?
4.
And indices is nothing but hey at this particular place you have a nonzero value, right? It's just telling this in this you have nonzero.
It will change based on the data, but it will be fixed for that particular document, fixed or varying depending on the data that you use. But for embeddings, if you use that model, it's.
Fixed. There is no change in that.
Dense embeddings.
So how many? So here what you can tell is you can tell sparse embeddings has four non zero values. Remaining all are zeros.
Till here is done. The only logic what we wanted to understand here is what is dense, what is parse, what kind of models to use for dense, what kind of models to use for sparse, and then how to convert our text into.
Array which is our embeddings right? This is very useful topic when we will work with keyword search, vector search, keyword search plus vector search which is I right? And then there is also new technology which is currently using I mean new algorithm.
Which is called neural sparse retrieval. First it was just vector retrieval. Then you had hybrid search retrieval. Now there is something called as neural sparse retrieval right where you will use a model called mini called.
So instead of BM25 you have something called as mini coil. So this is a algorithm again. So BM25 is an algorithm. If you CFIDF, PFIDF is an algorithm. So in sparse every time you define anything there is an algorithm.
Whereas in dense, whatever you define that you can actually train the model. So sparse is only for retrieval, not retrieval. You can tell it for search. Yeah, keyword.
What is it in denses? That's a keyword or?
Like a that is a vendor. No. OK, sparse is an algorithm. OK, like PM 25, so PFIDF. Those are algorithms.
It's mainly used for keyword search. OK, so you will be using sparse embedding model to search.
Uh, OK, it is more than search. It's used for search. I mean, you will technically use a very small vector search, right? Yeah. So to clear, we'll start with the basics.
Dense. It's a modeling which is used for vector search. OK, is this clear? Yeah, sparse are algorithms which are used for keyword search. OK And what are these

algorithms?
BM 25, TFIDF and BM 2042.
BN 42 I guess BNBN 40 BN 42.
PM go to DM DM.
No, there is BN also uh quadrant.
BN 45.
Shouldn't.
Let's what is the keyword search. Let tell search. It's like you have dense embeddings. You have a trained model. Keyword search is. Let's suppose I ask hey search about neural network. So keyword search will just use well.
Neural network and then it will only get neural network. Now basically for neural network you will need deep learning, ML, then something which is relevant to neural network. Vector search will get those results right right? But if you want to reduce hallucination, you combine both of them.
So you can say vector search has some kind of diversity, but it stick to relevance, whereas keyword search PR mapping you can say filter. So when it comes to filtering, what will you use? Will you use exact keyword or will you use some kind of dense embedders?
The keywords, right? So in our vector databases as well, for filtering you usually use exact keyword to keyword match.
So that's why 0.
The 0 for so that's why we start for in particular 0 values. We have more 0 values is because we want. Oh correct, because that that wording document doesn't exist.
Yeah. So even if you see there were so much indices in sparse, but only you had.
Four or five non 0 values.
There is one more algorithm. I forgot the name, but it starts with BN. If I check or quadrant document, I'll get that, but we'll probably proceed. It shows BM 42 BM 42.
Yeah, that is in BN also. So these are algorithms. For time being, let's just write BM 25.
And TFIDF. Is this clear? What's sparse and what is? OK, so now can we open this math check?
Yeah, I've shared it in the chat.
Is Ajay Sir active? We'll just see if he replies yes in next one minute. That means Ajay Sir is active. Yes. Nice.
Yeah.

Oh, any questions you have in dense and sparse embeddings?
OK, so so far whatever we did in this notebook was we just wanted to see how Denson sparse works. Now in this notebook, are you able to see the code?
So here I just I was just doing match check. I I actually didn't wanted to write this. So this for that notebook I only open to test this particular line right? So how does dot product work? It will add all these things. Now if you see everything is in one line and then if you just use MP.
dot, dot, dot. And then if you pass any new, you'll get the results. Now what is the syntax for dot? You should have saved length. Is this clear?
And let's start with.
This line import pandas as PD. So usually for data science you have 3 core packages or I'll say 4.
You have numpy.
Which is for data creation.
And it's called numerical Python.
Then you have a package called pandas.
Which is mainly used for data manipulation.
Collation. So let's suppose you want to do any filtering, grouping and then indexing, slicing. So most of the time if you have 2D.
Data which has rows and columns. You use pandas which is your structure data.
So let's suppose I have Netflix data which has the rank. Then what is the genre, genre of this particular movie, then title of the movie, tagline, description, all these things in a tabular format, right? So tabular format if you want to visualize it, we use.
Pandas right? So this is mainly used for structured data and then you have matplotlib.
So Matplotlib basically is used for data visualization.
If you want to create any charts, graphs based on the given data like bar plot, line plot, scatter plot, so during that time we use data visualization.
And then you have something called as scikit learn.
This is mainly for predictive maintenance.
Now what is predictive maintenance? This is mainly used for ML algorithms if you want to build prediction algorithms, classification, regression based models. So the base starts from scikit learn. So these are core 4 Python libraries that every Python developer needs to know whether it is data.
Scientist or not, but if you are getting into AIML, these are first four you start with.

OK, so here we are using SK learn. SK learn is nothing but scikit learn. This S and this K they just combined it and they made it SK learn if you want to install SK learn.
In your VDS code, the command will be pip install.
Sai Kit Learn.
It won't be a scaler.
Let me just confirm if I give this space properly.
Uh, it's correct. It is scikit learn.
But in Colab it is already installed, so I'll just comment this.
OK, so now when I mentioned curse of dimensionality, that means what kind of data do I have? So the data is always referred as features.
Right. One thing you have to keep in mind when I discuss text, text, text sequence that you are creating, right? You have padding. So what are those texts? It is dimensions. Dimensions is nothing but features. So here TF IDF.
We need to get it from feature extraction and then what is cosine similarity?
I told a keyword called pairwise. Now what is pairwise used for checking the similarity right? So that is inside a sklan dot metrics pairwise. Pairwise usually has similarity based algorithm which is evaluation metrics we are using for sense similarity.
Everyone knows what is TFIDF and what is cosine similarity. Any questions in these two things? No right?
OK, now what we are trying to do is I want to create a Excel right? The Excel of total 4 documents and the text of this. And once you create a dictionary, if you see it's a dictionary, my key is.
Is doc query and then text. Now this is your column one, this is your column two.
And then you use something called as data frame. Data frame is nothing but create.
Row and column visuals. That's it.
Can you just print DF?
So this is your column one and this is your column 2.
So this is when you have your own data, you're converting it into DF. If you have CSV right, you can directly do PD dot.
Read CSV and you can add your CSV file. So why is CSV file used? You have it in rows and columns on Vivek CSV.
You can do it like this way. Then you also have Excel.
Which is PD dot read XLSX and then the particular file path. So what pandas will do is you just add your path and I will display the table for you and then it has multiple operations that we can perform.

OK, so now what we have to do is we have to instantiate vectorization and then use a function called fit to transform.
You are early.
Is this clear what fit transform will do? So once I add, once I instantiate it, whatever takes I have, I just have to pass it inside that and it will create the metrics. That metrics will have the values that we calculated.
But if you print it, you won't be able to rectify it, right?
This one should be clear, right? What is your first coordinate? Can anyone pick up this? What is the same first zero document or first document or or 0 document? Then this is your first document. Sorry, first, second, third and fourth. Yeah, right, zero document doesn't, right?
But it starts with 0 index. Now can you tell me what this value is going to be? You know the third word, so 01234 word correct. So if you look at our Excel sheet we had total 4 documents and we had total 13 words. So this means it starts with zero and it goes to.
Well, there are total 30 words and what is the TF values of it? It's added right now.
How do we rectify dimensions? What do we call these dimensions as features?
So text you convert text into dimension. Dimension is a features in ML and all these algorithms. Now this is our features. What I need to do is I need to convert this ID into the actual word of it. So if you want to do that you can tell hey this is the ID, get the feature name.
OK so the next command if you see what is vectorizer, it is TFID of vectorizer and I'm just telling get feature names out. This is the function that it uses and now if I run feature names you have adventure. So this thing right third document zeroth index. Where is that?
In my third that is nothing but this one. You have adventure, so the TF of adventure. Kama.
dot document 4 is.
Where did that go? Is 0.43 something?
No, this will be adventure, no.
So let's also print that out. I'll just print this now. So if you see this three adventure is 0.4377.
You understood every single dimensions is referred as features in a scale and library or if you're using Tensorflow or Pytorch.
So far what we did, we just defined our documents and then convert that into TFIDF.

Now the next step is if you want to display it, we can use pandas again so that you can check it in the tabular format.
And what was the next step? Once you have the idea of what do we do?
So we are at step one. This was done. Step 2 we create parse vectors which is from TFIDF which is done. Now what is the next step?
We have to calculate the pair wise similarity, which is your cosine similarity.
So how did we import cosine from a scaler dot matrix dot pair wise import cosine similarity, cosine similarity you have to pass your vectors.
We can find cosine, but not cosine similarity. That is the reason we have the scalar.
So the cosine metrics, it's usually defined as cosine similarity.
And here you just have to pass both your PDFIDF metrics and once you pass that, what will be your what you call?
Cosine matrix. What will be the shape of it?
What will be the shape of the matrix? No, whatever documents you have, it will be double. So one thing you have to notice cosine similarity will always have same rows and same columns.
But if you're doing it on the inference level, it will be your inference query and top key.
Let's suppose as order movies, order is 1 and if I'm telling I need top five, it will be 1 comma five. Is this clear? Yeah.
You can also display this.
Now if you look at your diagonal is it is 111 and one. What values did we get? We got 0.4 but here it is 0.3 but we measured it based on term frequency.
After IDF it got 0.33.
If you cross it only document one and document four as two variables, 2 keywords same right? So now if you check which is the similar document for document one, it should give document 4.
That is the logic here. So in pandas if you want to do indexing it is called LOC which is location. So LOC is based on values. ILOC is based on index. Now here what I'm trying to do.
One is to stop it. That means if I need three, what is indexing this? It will go zero to three and it will ignore the last right? So whatever you're doing it here plus one. Why is it starting from one? Can anyone say the first one is?
First one is itself because it will be 1.00 so I want to start the first index right and then you just have to return your recommended.

The reason why we are breaking down here is because we'll use that for recommendation and there it is using the same syntax.
And when you're using sort values by default ascending is true, but we need the top values which is ascending.
If I keep it 0.
It will be one.
Till here is it clear? So the only thing is how you define vectorizer and after vectorizer what is the next step? Cosine similarity. So this line that you see here right feature selection.
Let's suppose we are doing preprocessing. For example, I have 3 tables, Titanic, everyone of us not Titanic. So now your goal is to predict whether the given person will die or not. So what were the conditions?
2.
Gender, gender, gender, then the novelty. If you're class, if you're class, class, like no, no, no, no. I mean if you want to predict whether the person will, how will they survive?
They are women, women and children that first. So gender, gender came into picture. The second one was how high class you are. So they will be given that jacket and will be taken to the boat. What is the third thing that was deciding that it is your age, age, right? So now you have a data set for.
OK.
Yeah.
That is also the class in the sense it will see like what class to belong. And there will also be chances that suppose you are a female who is 60, sorry, not 60, we'll keep it 40. If you're at the lower, you're dead.
It will come up. There might be that you will survive, right? So you'll also have what you call cabins in what cabin you are currently. So this is data, right? So all the things that I said are features. So now where I'm getting at this, let's suppose you want to. CFIDF. Apart from this you'll also have something for this. So let's suppose for the same class gender it will be male, female, right? Which is a string. But if you're training a model, you will not need strings.
That should be integer. Either it will be 0 or one. If you have three plus it will be 012. If not it will be 01234.
During that time you have to use the same logic. Every time you use a scaler you define I need standard scaler and I need to feature extraction. The next step should

be fit and transform. That means I'm defining this feature extraction.
I will pass my data. Every time you pass your data, the function will be fit in the next.
OK, so quit and then transform. So what happened here? I applied this data and I got 0 comma 30 comma four. OK only when I use get features out then I'm getting these values.
So fit is a very important function when it comes to ML or if you're using SQL. So what does it mean? You're passing your data and you're transforming it.
So I'll share this collar notebook.
It.
Is it possible to upload ACSV file?
Thank you.
No, you might not have the right to do that. I can't. You cannot. How do I pass the CSV file? You give it to me by WhatsApp if you can and then.
OK.
How much?
OK.
Yes.
So this the same IMDB dot CSV, IMDB dot CSV. OK, these three files are same.
What is? Whatever.
So here by what it will do once you upload any CSV file, it will convert it into vectors, the one that we need and then percent similarity. It's used to similarity based on our recommendation. So this is called E one search.
I'll repeat these two parts. We'll just use it for keyword. Then we will compare it against. What kind of embeddings is this? So this is for keyword search and this is for.
Is this fine? So we just want to compare what are the different results that the give and what keyword we give. So we'll have two different class, one is from TFIDF and one more is from X.
Now where are we importing this now? I'm not using sparse here, right? Because I'm already using. So what algorithm did we use for sparse here?
Algorithm VMVM 25 and what other algorithm I said it's here only, right? So since we already have TFIDF, there is no need to use.
So the first step is we have to install it again. You don't have to create a copy, you can run it in the same collab.
That works. If you want to modify anything then it will be useful.

And here until that is downloading, can you see these files? Click on files then upload and that imdb dot CSV.
So in previous code, what did we use PPD dot?
What was the command we used for creating 2D data frame right? Here we are using read CSV because our data is not in dictionary. We already have a data which is in CSV. We first want to read it and then you can convert it into data from.
But if not, you can directly use this line itself. Even this is a data frame.
Now if I do DF dot shape, can anyone tell me what this is mainly about? 215 rows and 13 columns. This 13 columns is mainly labeled as features. Is this clear?
What features is?
And here you can use different ways to display the data. One is DF dot head which will display top five rows.
So you have rank, then you have name which is the shawshank redemption. You have your rating, genre certificate and there are other features, right?
So what kind of use cases can we build? We can build recommendation system and if in case you have a season column rate, we can also predict whether this particular series will have the next season or not. OK, so predictive maintenance algorithm we can build it.
And if you want to display the bottom, what do you call it? Bottom rows, you can just use tail -5. No, it will throw an error. What did it give? -15 it give top.
-5 should be the bottom right at -5.
No, that didn't work. That didn't work. So if this if you give right, if you you just empty, oh it does, it will print the same thing it does.
We use both the top five and last five or -5 -, 5. What have you -2 then?
No, no, that didn't work. OK, it is doing everything and it is just doing the last. So one is tail, one more is head, but if you want to randomly generate anything, you can just use sample.
Sample and I'll use 6.
OK.
Now we have to do is what table? Suppose you want to build a recommendation engine, what columns will you pick?
Rank will ignore because that is just index rating rating, genre rating genre but rating for search will you use because dense and sparse we are using which is mainly on.
But if you're doing filtering like based on this thing, then you can use rating.
Genre we can use. Genre we can use cast. Actually, cast like if someone likes this

character, you can just recommend directors, directors and writers, writers, yes.
Tagline also you can use. Tagline is also very so you have some context in there.
So you can combine tagline. So now what we will try to do is we will create a new column, right? We will create a new column by combining existing columns. So whenever you're creating a new column, you just have to ensure that.
From wherever you're concatenating it, even it has to have a same number of rows.
So now if you're combining some column which only has 249, it will throw an error, right? So how can we find the find the empty data?
You can just do DF dot is null.
Not some. If not we can also use info, but this will give you 000.
OK, so the function is DF dot is null dot sum.
We have no empty data. Now what we can do is we can just create a new column called overview.
You can also create anything. It can be summary or it can be overview or it can be just prompt. So this can be anything.
I'll just give overview.
And here what I will do is I'll combine name tagline.
Was it Ari?
Right.
OK, now here you might have seen one more new variable called as type, so I'll just show what as type does.
What is the data type of year?
It's a integer, yeah. So there is a function called here equals to DF dot here as type STR. So now what this as type will do is it will convert that entire column into.
String. Now if I run this again.
Here is a object. Object is nothing but string.
And what is axis? Axis is nothing but if you give to axis equals to one that is for column like your you want to modify a column.
If you want to modify your row, it will be 0.
I'll show this in a different cell. Let's suppose I want to remove rank. Rank doesn't make sense, right? So I'll just do DF dot drop.
Rank and then I need to define axis.
Now what is axis? What am I dropping? A column? It is one. If I'm dropping a row, it is 0.
And if you want this to permanently delete it from the actual data, if you see I didn't

saved it in a variable.
That's.
No, this is error.
So here you have to define zero.
So here if you see I want to drop rank, it removed it, but if I print it in a different cell, it's still there. Why? I didn't save it in a variable.
Ha, but there is one more command called in previous.
To be true.
So now it's gone.
OK this you have to use by cautious. OK OK in place equals to true. Now in this case what you're supposed to do in dictionary list tuple we saw one function. After you define a list, what was the second step we need to do?
Creator.
Hi dictionary there is a function right dictionary list item copy. So let's suppose you create any list. You have to create a copy of it. Same goes for your also. Once you create any data frame, create a copy of it.
Then use that copied data frame to do any modification. Don't touch your action data frame. So now in place true in the sense rank is gone. Same if I use for certificate.
Certificate is also gone. Now what is the DF dot shape?
Is this clear? OK, now we are adding a new column which is DF overview and we are combining some of the existing columns. Here if you see how do I want to apply that logic when I'm combining.
After every single column I need to have a space and then I need to join.
Writers. So now if I do DF dot shape it should be 12.
I just started a new column. So now if I do DF dot shape it was 11, it is 12 now. Till here is it clear? Now what we will do is when we create TFIDF and cosine similarity we will pass this column.
So this class I've already explained.
I'm defining this and nonstop words. I want to remove the stop words. Max is 5000, that means it will be a metric of 5000.
And then return transform we have already seen what is supposed to be passed, which is our mix.
And once you click on recommend, you're using cosine singularity logic and once you use cosine singularity, you're just getting the sorted result which is a.

And for indexing, what we are doing is a verification top three. If user needs three, I have just taken top three. Is this clear this line up for what it does?
I didn't know. Who are you?
Like below these three lines are clear right in it.
Thank you.
2.
So TFIDF is mainly for keyword. You can just add one comment here. This is for keyword and below that you have dense. This is for vector based and what model are we using?
What model did we use for dense earlier?
Model name Jina.
So this one closely notice the text that I'm passing. It is a it should be string and this is a list and what is the column the overview that I've used.
And fit is supposed to be used only for data. Once you have to give it for the algorithm, then you can use any other function which is predict or recommend.
Of what error?
Anyone got any error message in DF for you?
Oh.
OK, we're going to after London, there is no for us.
OK, yeah.
That will do. Is it clear? Yes.
And once you define TFIDF recommender, we also have dense. The logic is same.
What are we trying to do? First we are defining the embedding model. The model is Xena. What is the embedding size of this?
Dimms.
768 I know fit is the same in fit. Now how are you modifying it? Instead of it transform, what is the function?
How do I convert text into embeddings in fast embed? What was the function we used?
So let's suppose I have dense embeddings which is already defined dot embedded.
So for TF IDF it is fit transform for fast embed which is your dense embeddings it is dot embed.
Is this clear?
And in recommendation the logic is same.
You have a new user query. We are converting that into embeddings and then using

cosine similarity. Now how are we mapping it? It is query against embeddings. It is query against documents.
Is this clear? So if you remember in Excel sheet I showed if it is one to one document, it is document document. But once you go to inference it will be query against document that is for keyword. But for dense it is just embeddings.
Right, so you're searching based on.
Query embeddings to sorry, query embeddings to actual embeddings. Is this clear?
These two lines here it is metrics, here it is embeddings. Now this new function you might have seen flatten, flatten. It's like if you have 2D array, convert it into 1D. That's it.
So this will download the particular model which has five files similar to what we saw here.
It will download 5 files. I mean that five files. The major files are tokenizers.
Is it downloaded? It will take around 30 seconds.
OK, So what is the column we need to target now? Overview because that has most of the context, right? So in that context, once you have your data, the first step is fit. You always have to follow this approach, define your model.
Fit transform and then predict right. In fit transform you have to actually give your data in in recommended or predict you have to usually use your user query which is for inference. So this is for training.
If you see TFIDF, it was very quick, but dense embeddings might take 30 seconds to one minute or more than one minute also.
Does anyone recall how to switch into GPU runtime then not directive for?
Runtime then and then T4, yeah.
GPU to be faster. This time will be first.
No, you'll have to run it everything again. You'll have to upload the data again.
Everything will be flushed up then, but I'm gonna show the output. I actually asked a query called funny comedy movie and if you check out the TFIDF results, the 1st result is a horror movie.
OK, but if you look at this second.
The Great Dictator and Toy Story Three. It comes under comedy, but if you look at Dense, all three are comedy.
And if you look at this course, it's pretty high.
Combination of both is better the result of.
And now we change the data, right? When I train the data, I just use name and tag

link.
Let's see after we do the, you know, we added more features to it. Huh.
But the best combination is combine both hybrid search. How do you combine both?
Like no no. So usually you'll have modules for that like ensemble. Ensemble is nothing but combine.
And the similarities will be way different like how how do they manage this? So here usually you define weights. Let's suppose you combine them. OK, OK, so now what will happen is you give certain weightage OK which is a list of two.
Now you want to give 0.7 for vector search, 0.3 for keyword search.
And then based on that, the similarity score will be calculated. OK, so that weightage is dependent on you. When you're doing TFID, you're doing it with the query, right?
You cannot break it down like are you breaking down the query in the in terms of by space.
You're not breaking it down. No, no, no, no. That will take care. So here if you see I'm using transform. So what is the role of vectorizer transform? It will calculate that algorithm. So this will take care of it, but it will look for the whole sentence.
Huh. Old sentence because it also has to apply the L2 normalization. What I mean is you're you're already passing the list as a query if you split it.
Query dot split with space and convert it to list.
Query dot sublet and you can type of it to list. Oh, you mean instead of transform?
No, it won't work because here if you see when you do fit transform, it is adding your L2 normalization.
So technically here also it's better to apply it, right? If not, this is this will generate different weights. Here you have different weights.
So here the weights are different if you understood because.
So what will happen like if we do for, uh, you know this within the list? We can try, but this course will be.
There won't be much negligible difference because we didn't have much data, but there won't be. No, there won't be that much.
So if you look at the scores that we had, it's very less like 0.33. Even if you add it in the list, even if it makes improvements, it will be plus 5 or -5.
Can you still you can please? No, not you still need to change the.
Check in there. Yeah, I'm just checking the results.
Did you get the same results or was it different? But these are comedy only or so now here what I need to do is.

Hello.
In this like.
Uh, very dot split, right?
That's it. Yeah, that's it. Then we can just type it. This is a list. This is a list.
We can just pass it down. You still need to do that, which one?
Yes, L dot vector is dot transform instead of array of query or a list of query on this one. I'm just trying to this one.
If we just do this as split over here, would it make a difference?
Oh, you mean that one? Yeah, yeah, no. Would it? What will that do? One second.
Funny comedy.
Movie. Oh, sorry, but one just.
This will add the entire thing. OK, this is one element. Yeah, this is one element. If you do split it, this three element. It is 3 elements.
It will throw error in my assumption.
It run.
Eddington.
Now it's different. The length the length does not match the index.
Because it multiplied it that many times. Huh. It is 3 now. OK, OK, got it, got it. But what if here I pass? OK, you have nothing to do. This is infant side.
So one more thing. If you notice here we used a new function. This was not at all required which is fill in name.
Basically, if in case there is any empty data, it should just fill it the empty string, but we didn't have any empty data.
Istep I already showed what Istep does.
And the remaining pandas functions will take once we learn numpy pandas and matplotlib.
But the combination of vector search and keyword search will be used heavily when we use rag.
Who's that?
Yeah, any questions? I guess I've shared 1-2. OK, we can check the assignment if anyone has done. I did it with the data that was shared or?
So how many files do you have in your project folder?
Why one? It should be three. I have main dot UI and we have download dot UI. No three. How validation file? I didn't like it then.
Uh, so the file you're asking about the decent file is OK, OK, so me is not UI for

entering and then tokenizer and pick what is not Jason and we keep the addition what we start. I put in four points, four points, one is data, one is tokenizer.
One is your training and one more is your reference. So you have to give me that.
The only thing that I implemented separately was I gave title, header and step and substance OK which are part of the Ctx. We didn't think it added value.
They give them as a special token. So from where did you get those token names? I think I saw the data. I saw the data.
It's all the data. It was part of. It was coming through this one. CPXA. No, no. So this doesn't matter, right? CPXA. This data is different. Special tokens are used for your model training. Yeah, but I wanted to, you know, just keep it as a separate token. Rather than use it for creating the dictionary vocabulary. OK, that means you want this as a common token, yes, yes.
So uh, this one that will be finally generated and they give it pretty high for it will have a non set mask and this will be their title. Their step. Oh can you talk to inference video client giving address? I did it uh so I did this inference.
No, I wish you would like to be at that. I mean, when you add any people here, right? If you get with the. Oh, no, no, no. I didn't want to be at that. OK, let me do this. Let me do.

◉ **Margi Varmora** stopped transcription