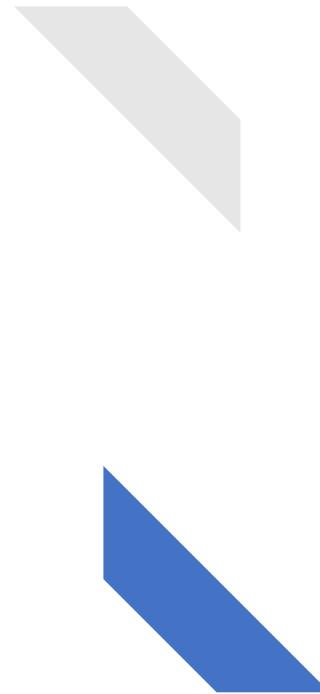


Vector DB & RAG

Sergey Tarasenko, PhD

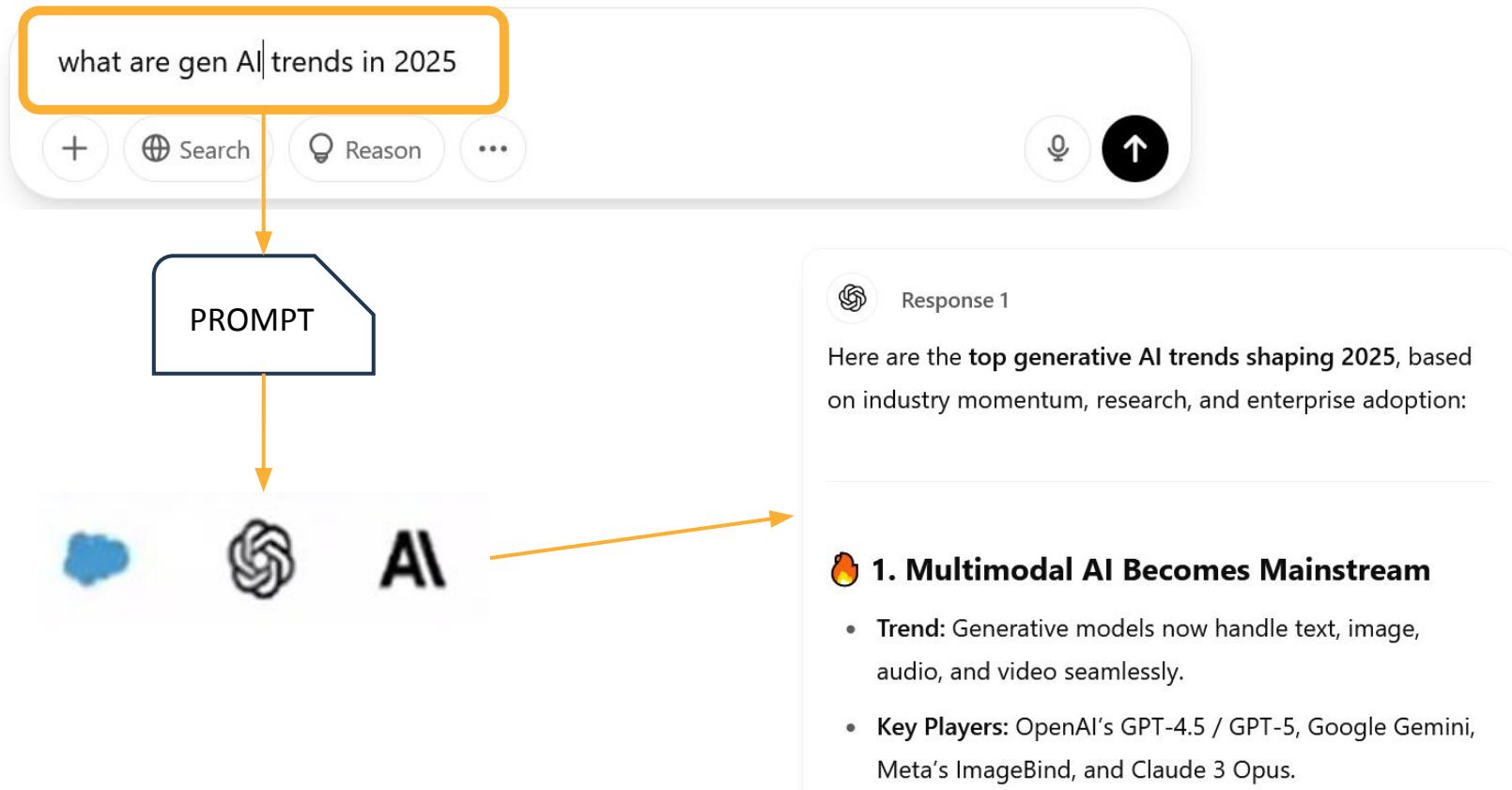
TOC

1. Era of GenAI
2. Word Embeddings
3. Semantic Search
4. RDB Search
5. Vector Space Search
6. Document Chunking
7. Vector DBs
8. Vector DBs and RAG

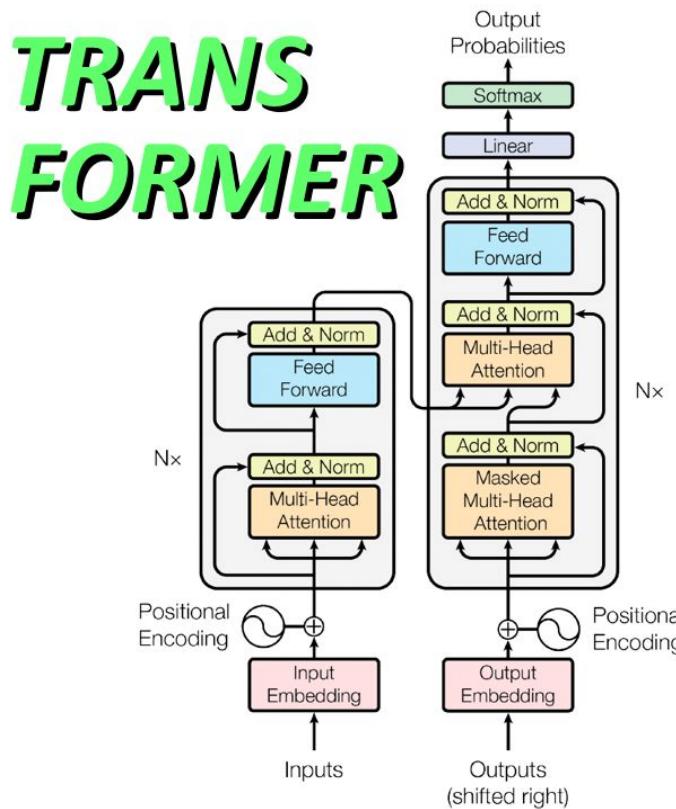


Era of Generative AI

Origins of GenAI: Prompts



Origins of GenAI: Transformer Model



After introduction of Transformer model in 2017 and consequent release of ChatGPT model by OpenAI in 2022.

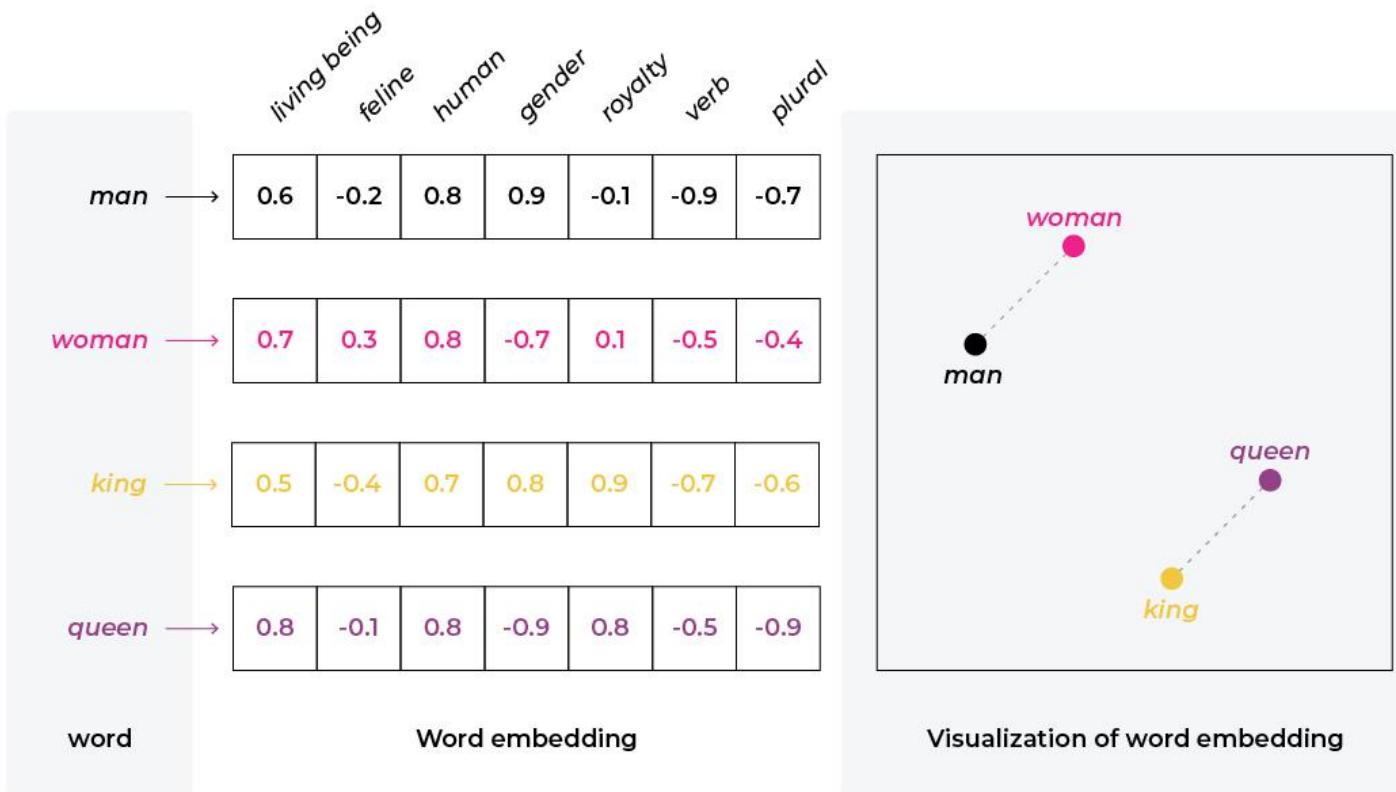
The world of AI has changed forever.

It became possible to interact with Large Language Models (LLMs) like with normal human vis-à-vis collocutor.

Word Embedding

Origins of GenAI: Embeddings

One of the key features enabled by LLMs is embedding -> a way to transform natural words and sentences into the multi-dim numerical vectors.



Origins of GenAI: Prompts

We use recently proposed techniques for measuring the quality of the resulting vector representations, with the expectation that not only will similar words tend to be close to each other, but that words can have **multiple degrees of similarity** [20]. This has been observed earlier in the context of inflectional languages - for example, nouns can have multiple word endings, and if we search for similar words in a subspace of the original vector space, it is possible to find words that have similar endings [13, 14].

Somewhat surprisingly, it was found that similarity of word representations goes beyond simple syntactic regularities. Using a word offset technique where simple algebraic operations are performed on the word vectors, it was shown for example that $\text{vector}(\text{"King"}) - \text{vector}(\text{"Man"}) + \text{vector}(\text{"Woman"})$ results in a vector that is closest to the vector representation of the word *Queen* [20].

Mikolov et al. (2013) Efficient Estimation of
Word Representations in Vector Space
<https://arxiv.org/pdf/1301.3781>

Origins of GenAI: Prompts

Sentences:

- 1.The cat runs fast and the dog is slow.
- 2.The dog runs but the cat is very fast.
3. Slow is the dog, but the cat runs fast.

Unique Words: cat, runs, fast, dog, slow, but, the, and, is, very

cat	runs	fast	dog	slow	but	the	and	is	very
1	2	3	4	5	6	7	8	9	0

Origins of GenAI: One-Hot-Vector

Word Embeddings: Word2Vec



In this case, we have **4** unique words in the training data, so we have **4** inputs.

Training Data

Troll 2 is great!
Gymkata is great!



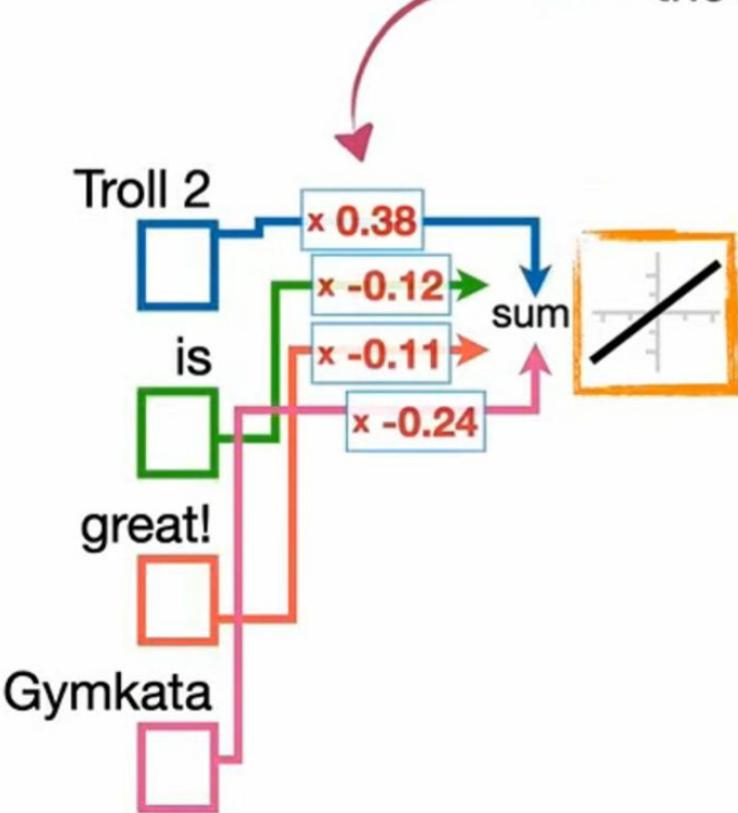
Word Embeddings: Word2Vec



...and the **Weights** on these connections will, ultimately, be the numbers that we associate with each word.

Training Data

Troll 2 is great!
Gymkata is great!



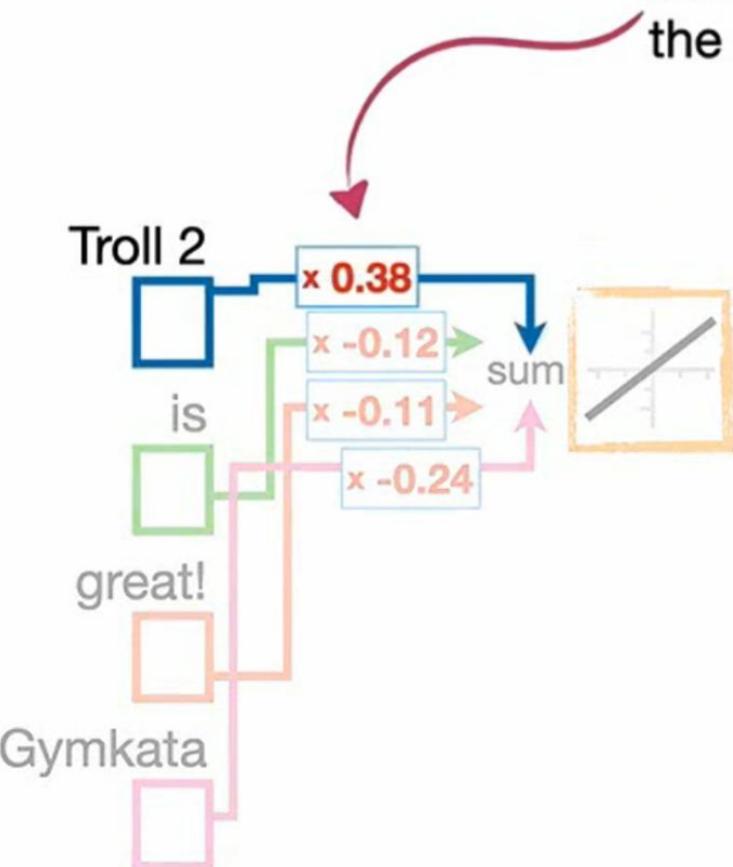
Word Embeddings: Word2Vec



...and the **Weights** on these connections will, ultimately, be the numbers that we associate with each word.

Training Data

Troll 2 is great!
Gymkata is great!



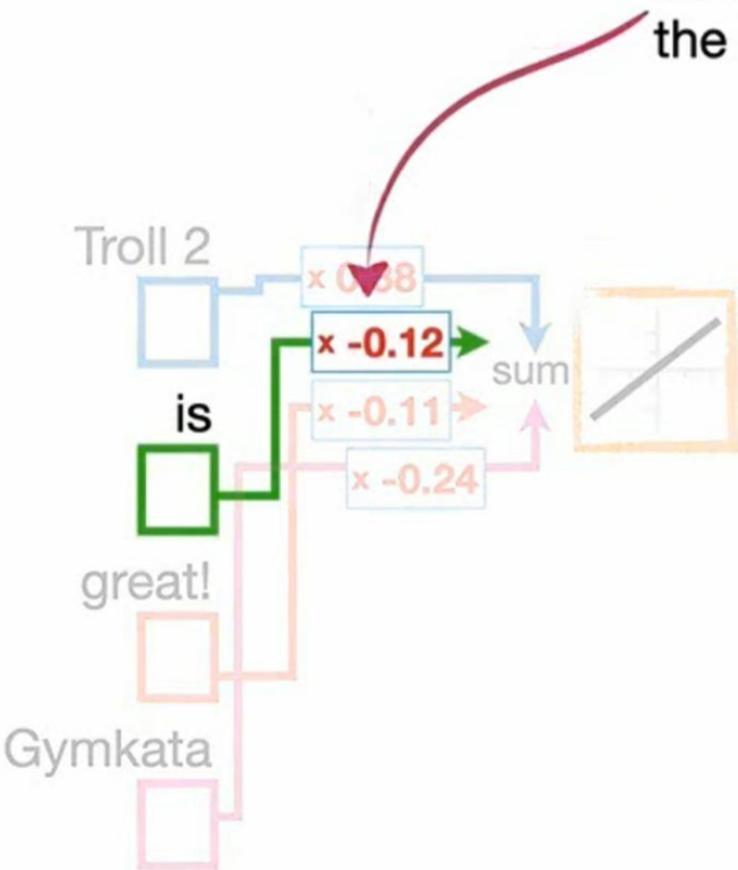
Word Embeddings: Word2Vec



...and the **Weights** on these connections will, ultimately, be the numbers that we associate with each word.

Training Data

Troll 2 is great!
Gymkata is great!



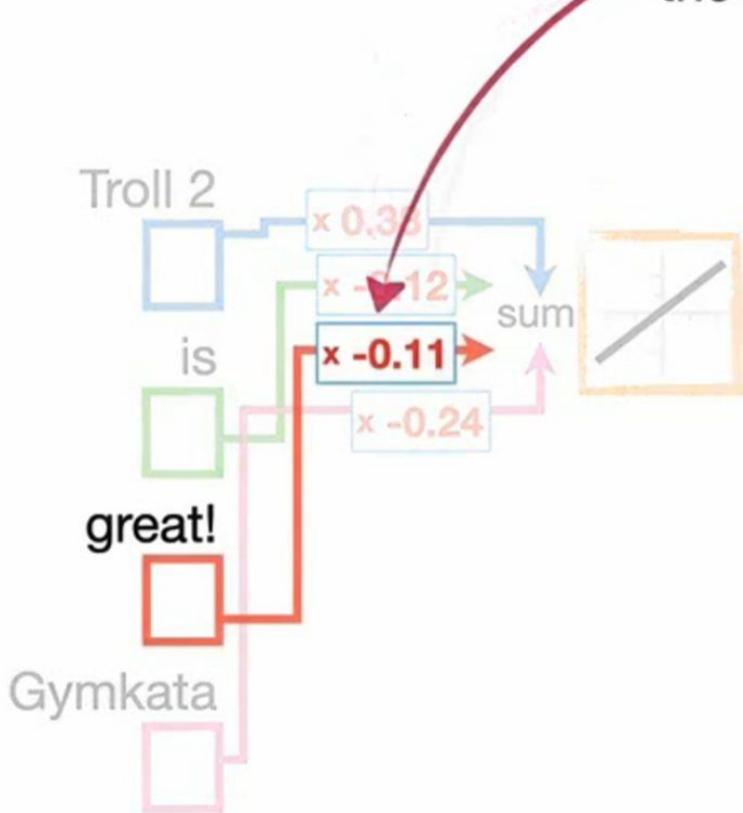
Word Embeddings: Word2Vec



...and the **Weights** on these connections will, ultimately, be the numbers that we associate with each word.

Training Data

Troll 2 is great!
Gymkata is great!



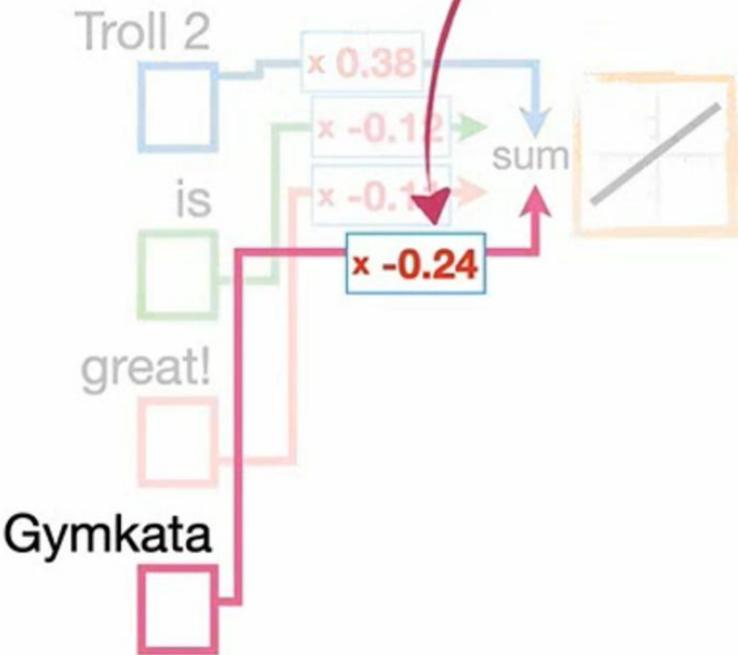
Word Embeddings: Word2Vec



...and the **Weights** on these connections will, ultimately, be the numbers that we associate with each word.

Training Data

Troll 2 is great!
Gymkata is great!



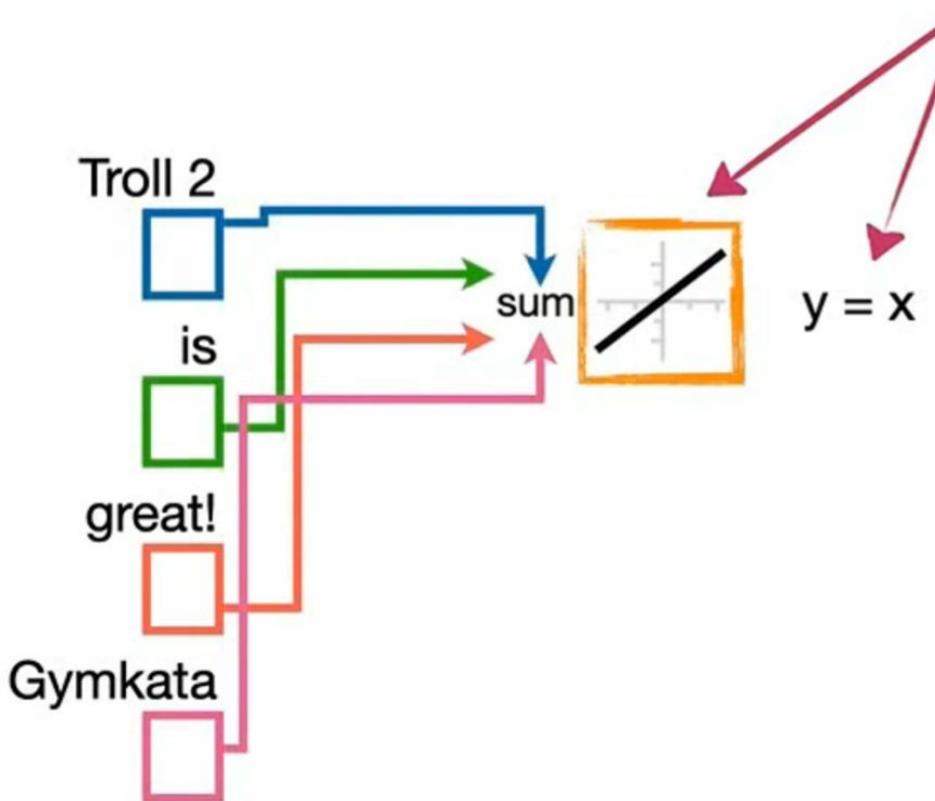
Word Embeddings: Word2Vec



Training Data

Troll 2 is great!
Gymkata is great!

NOTE: this Activation Function
uses the **Identity** function...



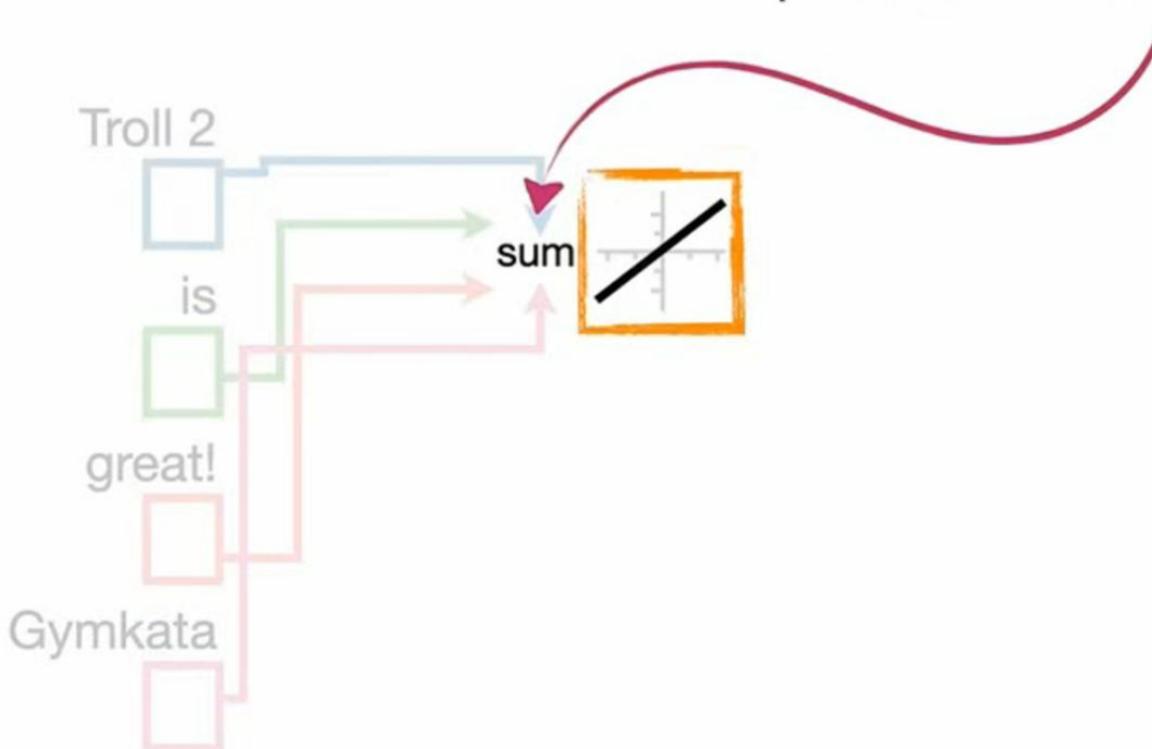
Word Embeddings: Word2Vec



Training Data

Troll 2 is great!
Gymkata is great!

In other words, this **Activation Function** doesn't do anything except give us a place to do addition.



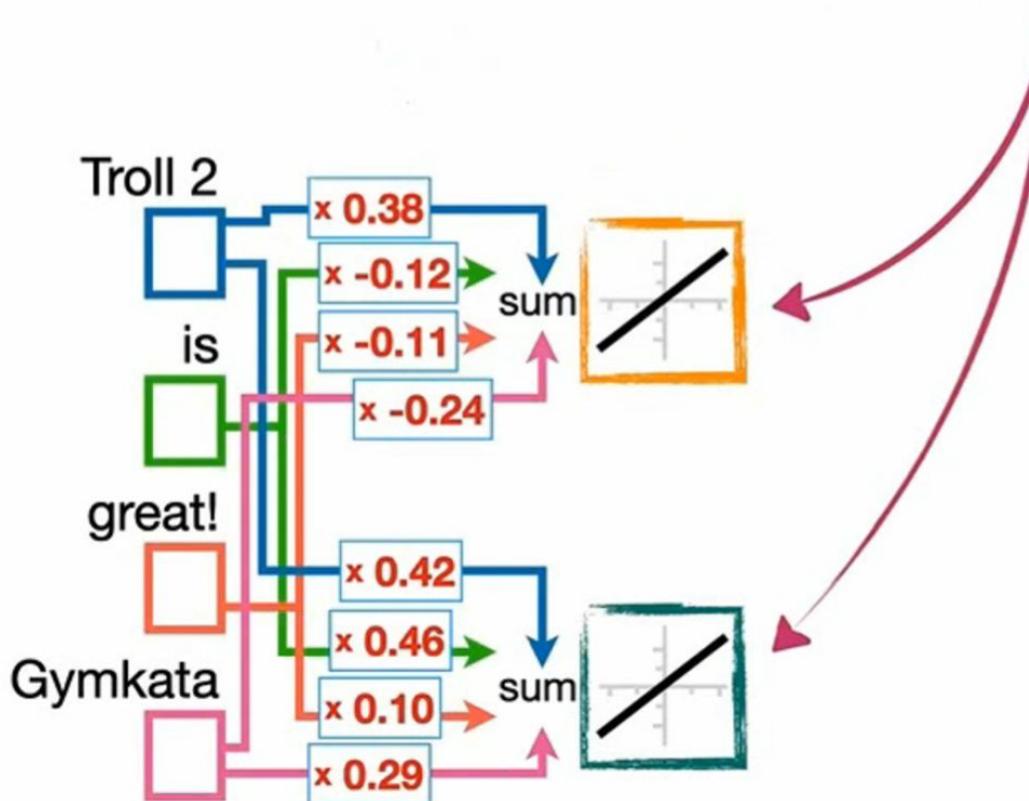
Word Embeddings: Word2Vec



...so that means we'll use
2 Activation Functions...

Training Data

Troll 2 is great!
Gymkata is great!



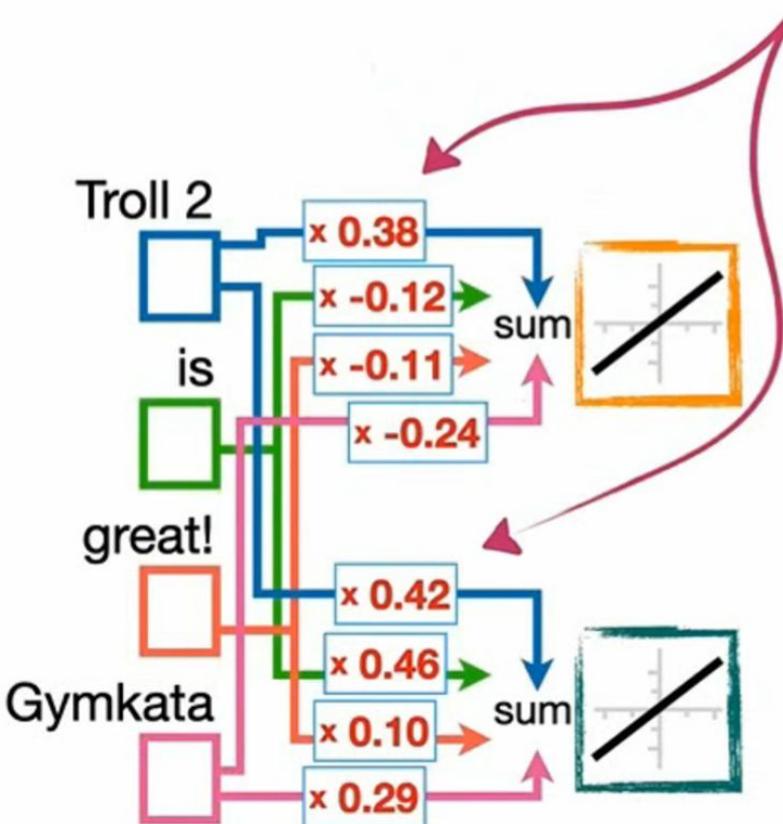
Word Embeddings: Word2Vec



However, like always, these
Weights start out with
random values...

Training Data

Troll 2 is great!
Gymkata is great!



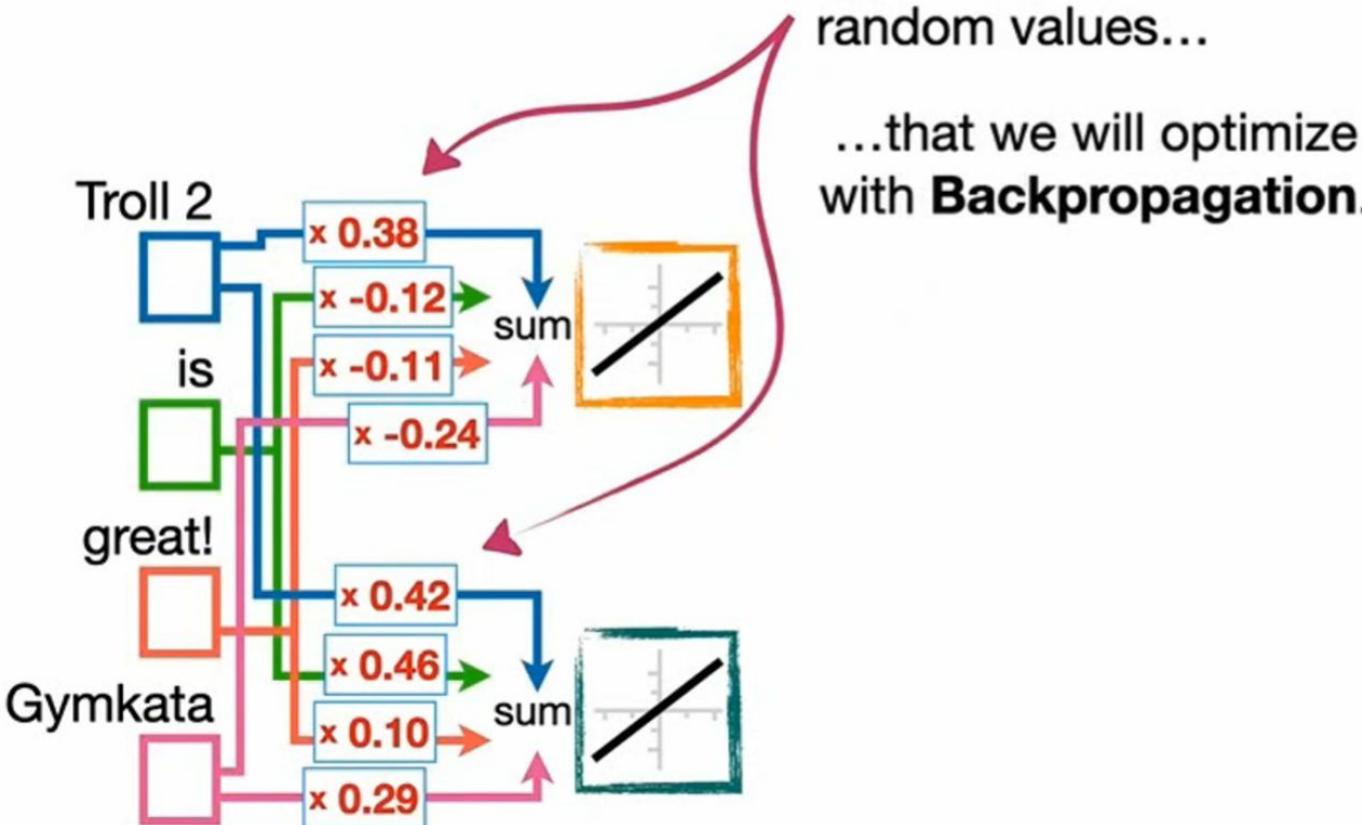
Word Embeddings: Word2Vec



However, like always, these **Weights** start out with random values...

Training Data

Troll 2 is great!
Gymkata is great!



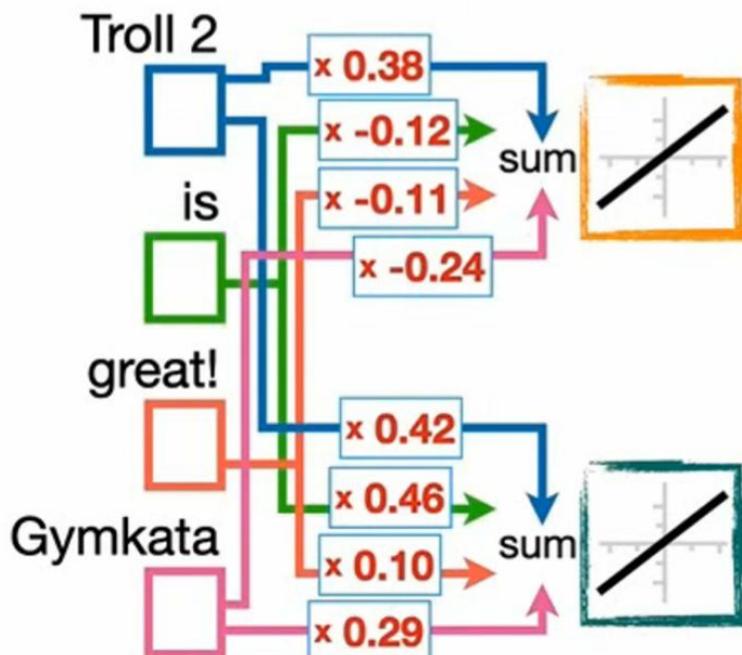
Word Embeddings: Word2Vec



Now, in order to do
Backpropagation, we have to
make predictions...

Training Data

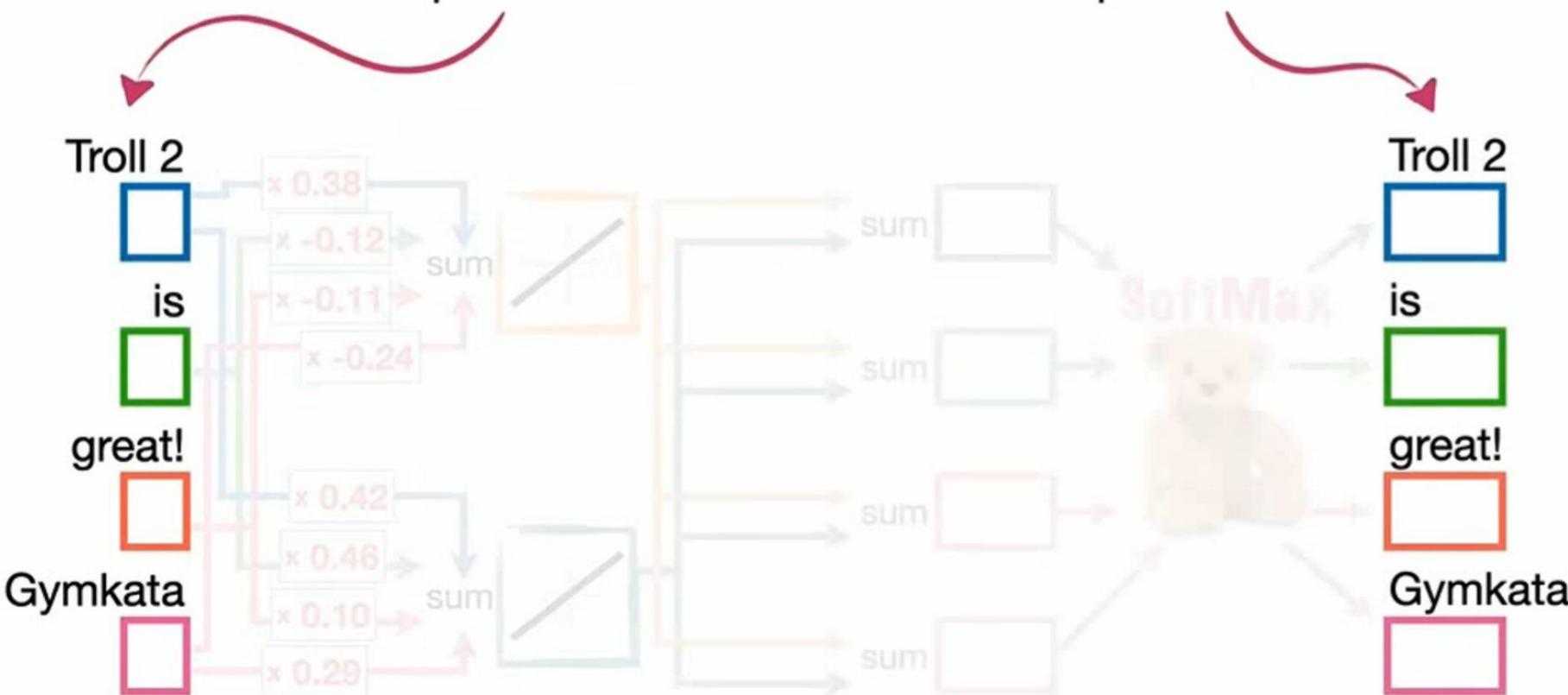
Troll 2 is great!
Gymkata is great!



Word Embeddings: Word2Vec

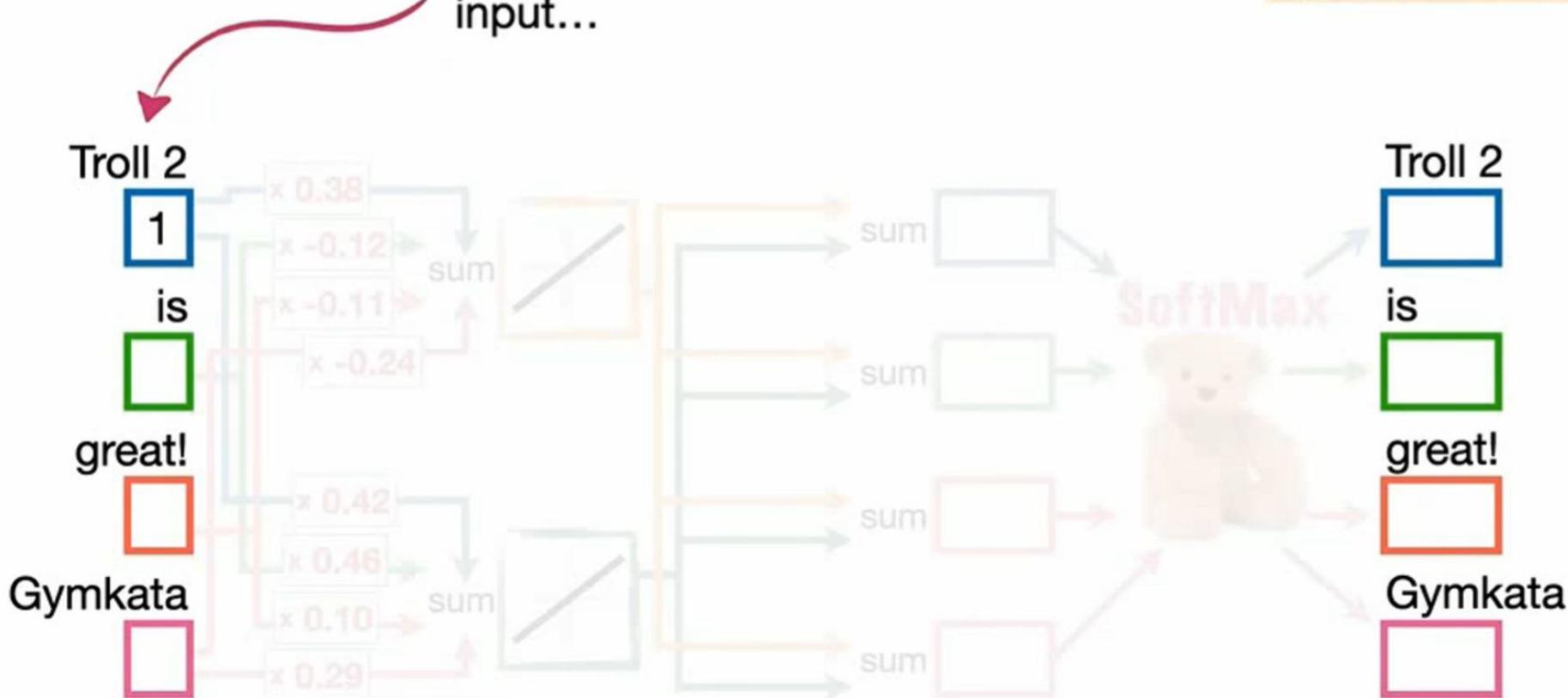


...so we'll use the input word...





...and we indicate that by putting a **1** into the **Troll 2** input...



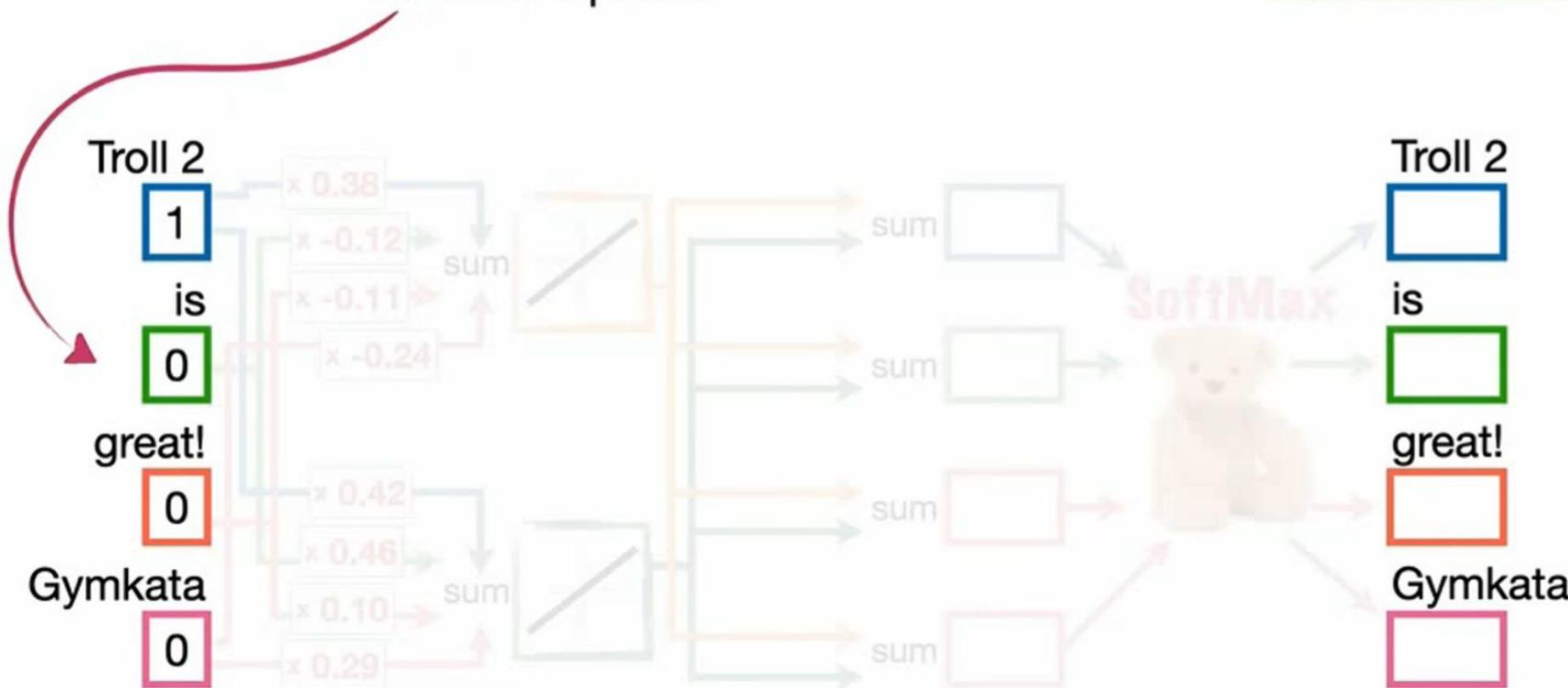
Training Data

Troll 2 is great!
Gymkata is great!

Word Embeddings: Word2Vec



...and **0s** into all of
the other inputs...

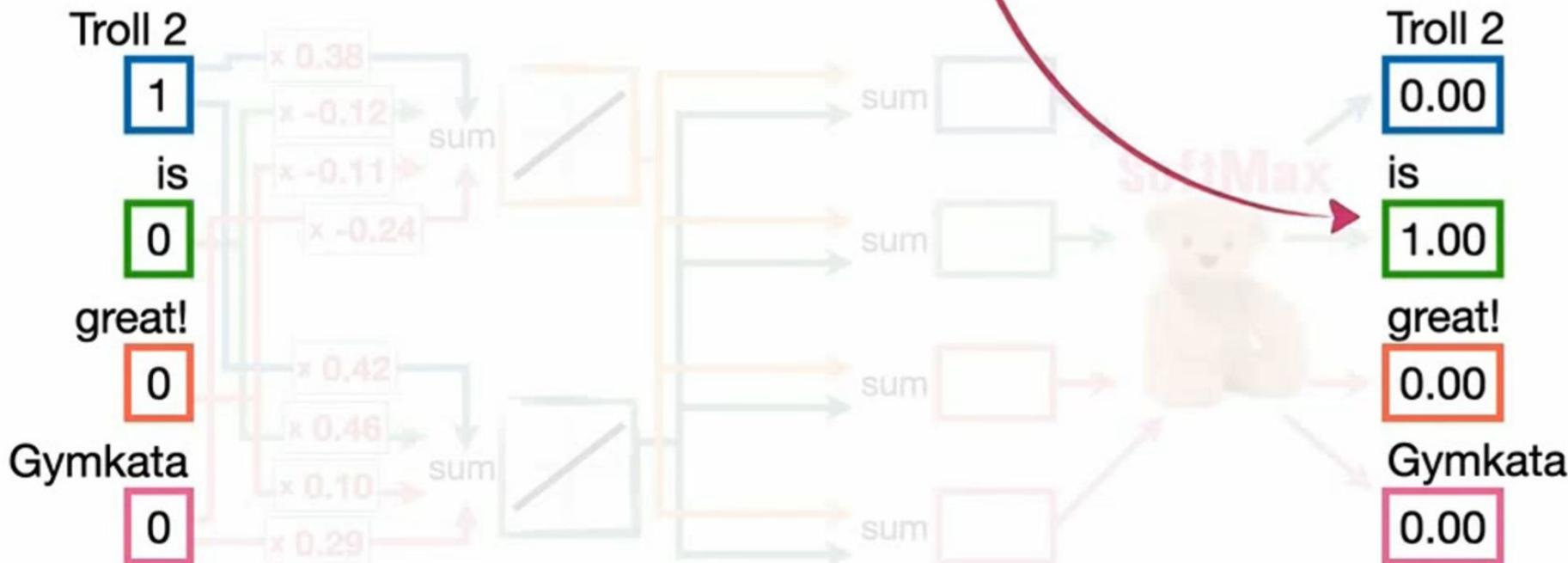




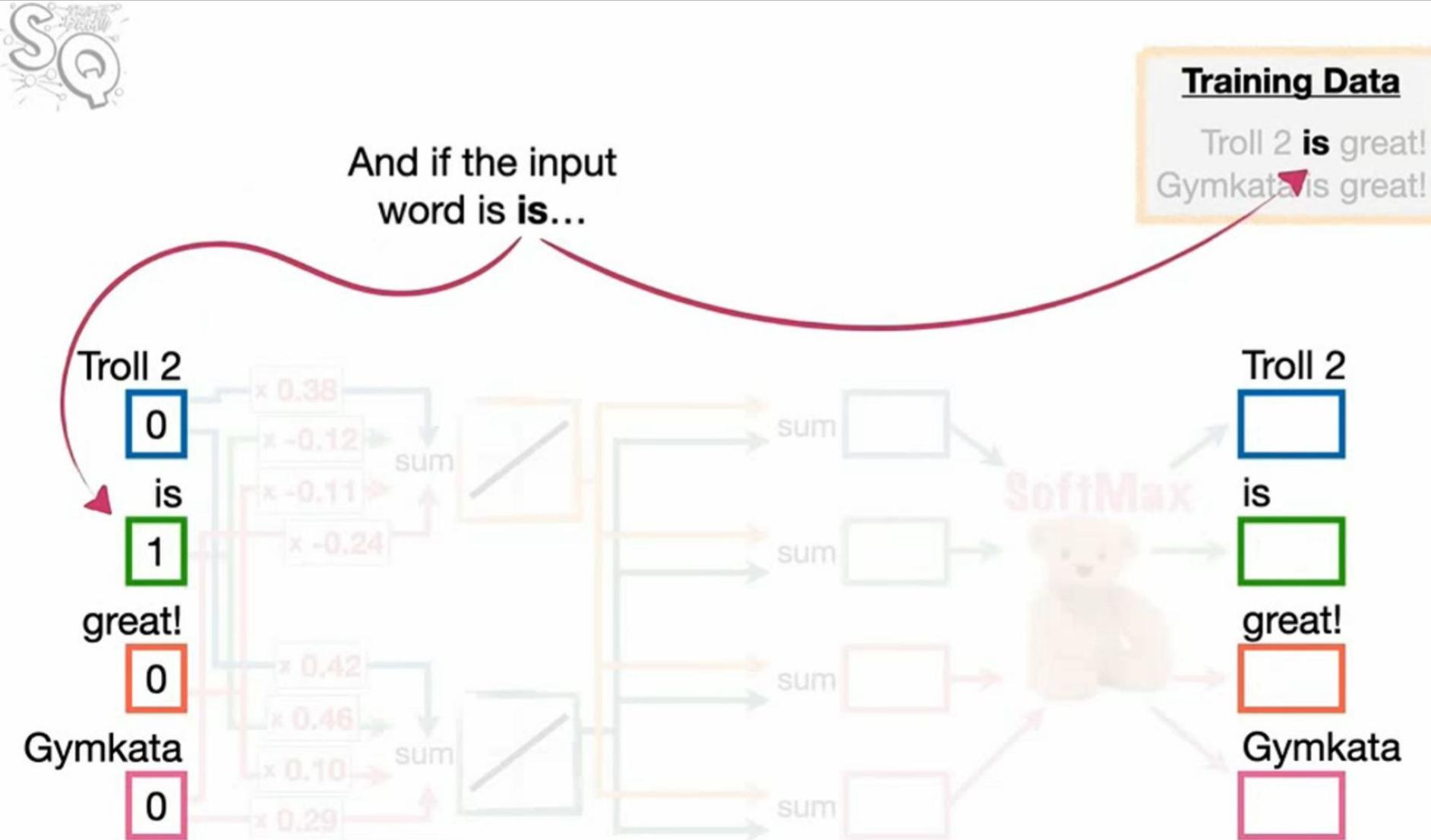
...then we want the output
for the next word, **is**,
to have the largest value.

Training Data

Troll 2 is great!
Gymkata is great!



Word Embeddings: Word2Vec



Word Embedding and Word2Vec, Clearly Explained!!!

<https://www.youtube.com/watch?v=viZrOnJclY0>

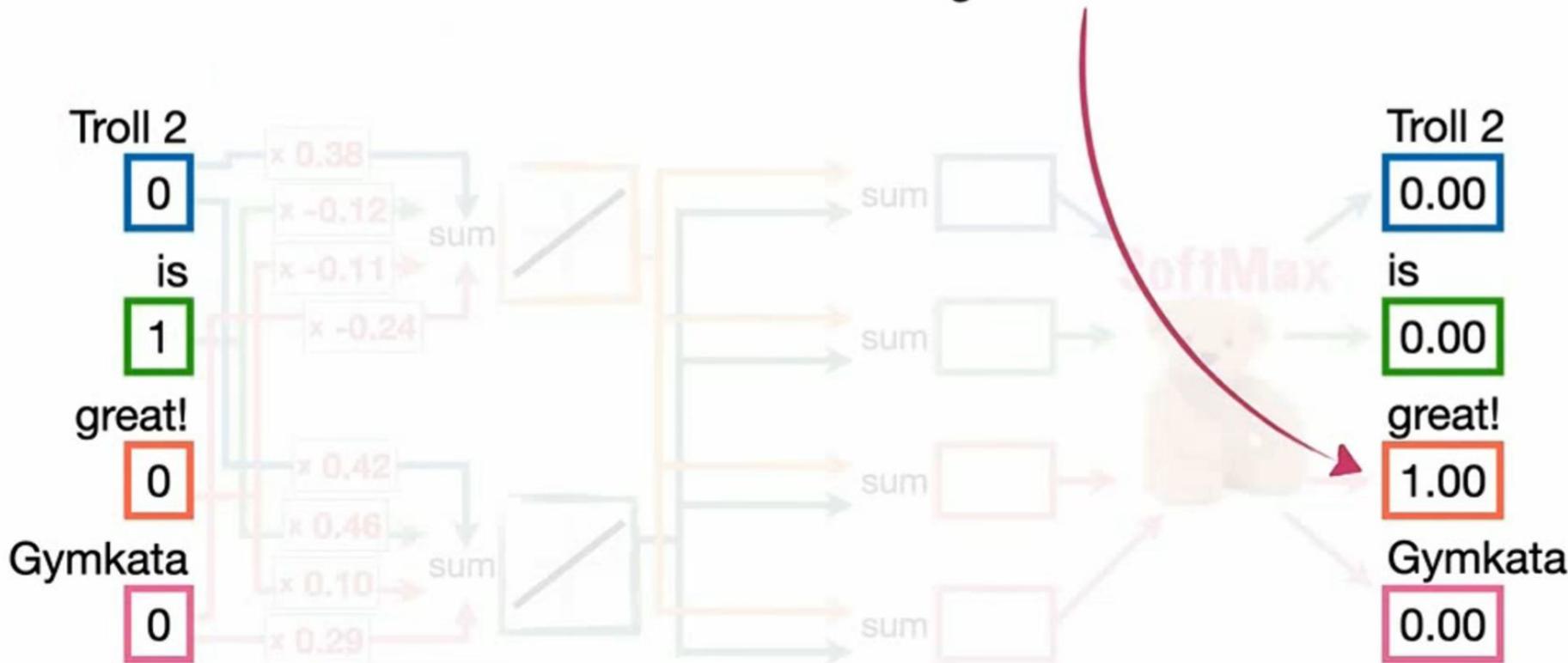
Word Embeddings: Word2Vec



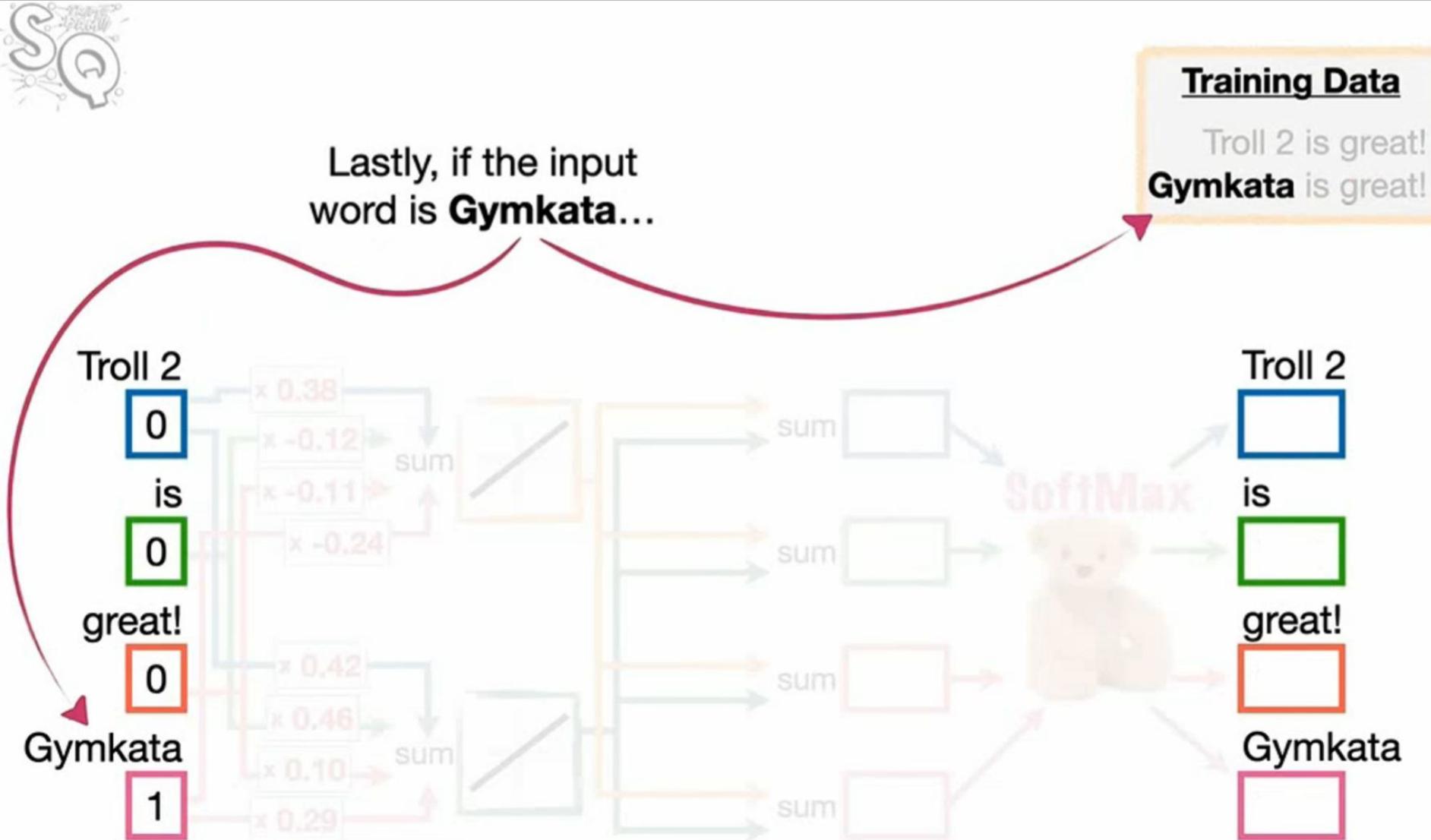
...then we want the output for the next word, **great!**, to have the largest value.

Training Data

Troll 2 **is** great!
Gymkata is great!



Word Embeddings: Word2Vec



Word Embedding and Word2Vec, Clearly Explained!!!

<https://www.youtube.com/watch?v=viZrOnJclY0>

Word Embeddings: Word2Vec



...then we want the output
for the next word, **is**,
to have the largest value.

Training Data

Troll 2 is great!
Gymkata is great!

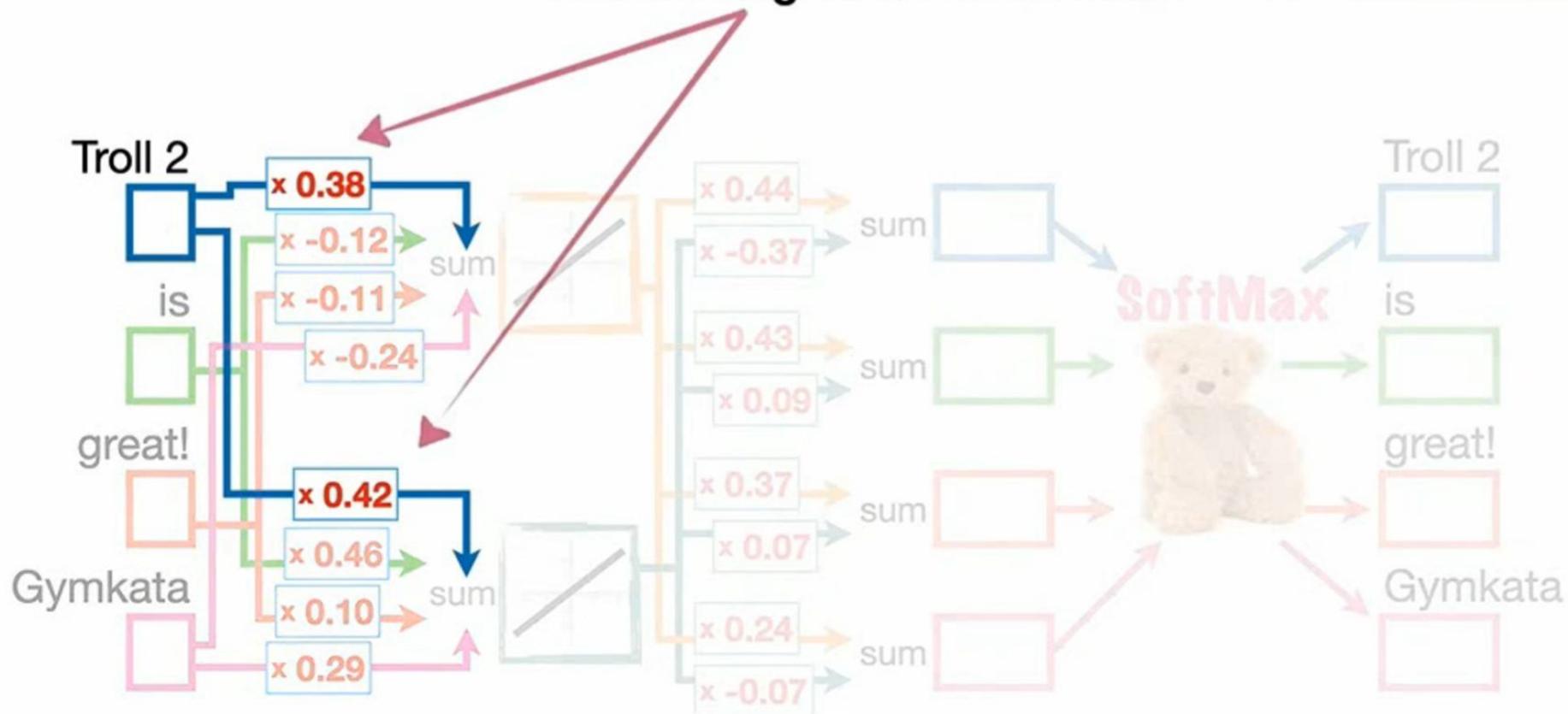




And since, in this example, we have **2 Weights** for each word...

Training Data

Troll 2 is great!
Gymkata is great!



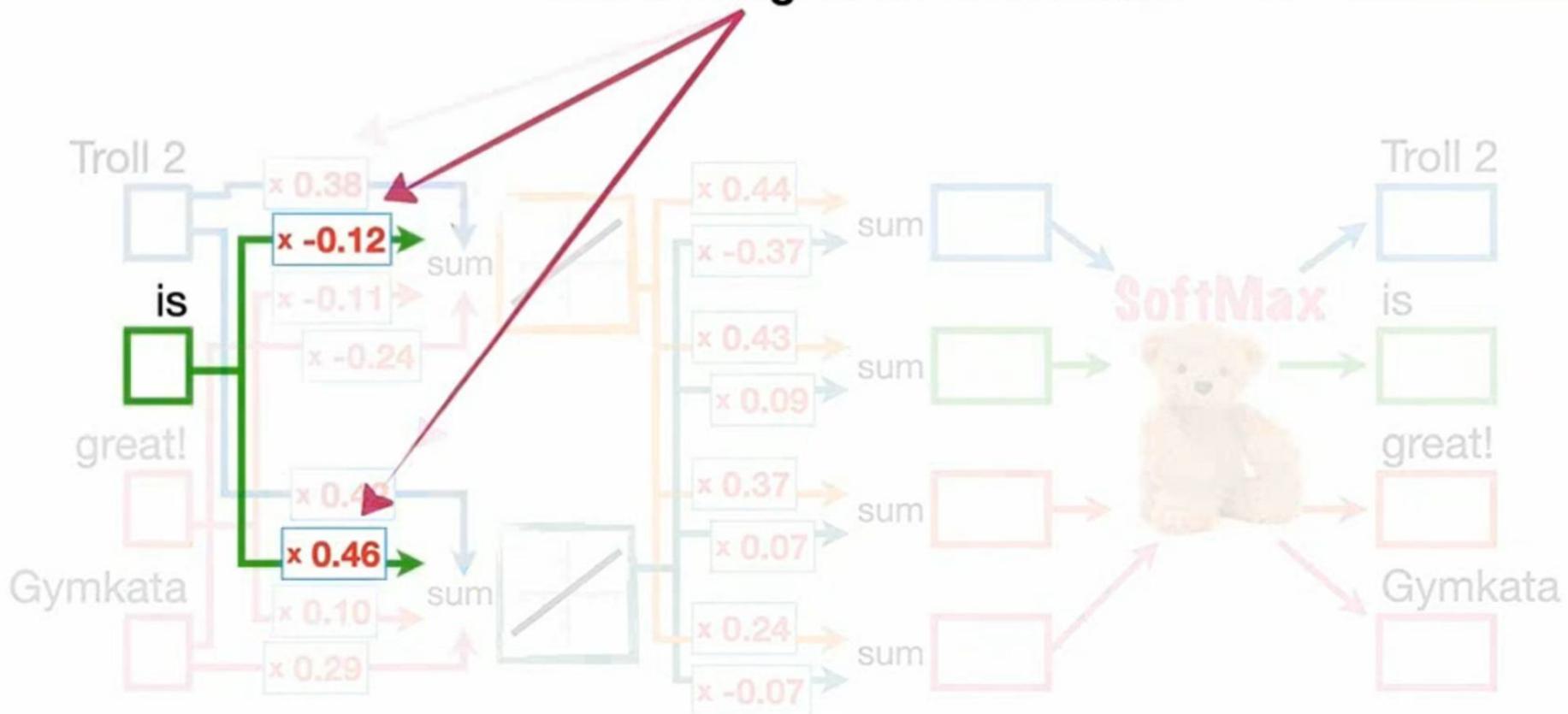
Word Embeddings: Word2Vec



Training Data

Troll 2 is great!
Gymkata is great!

And since, in this example, we have **2 Weights** for each word...



Word Embedding and Word2Vec, Clearly Explained!!!

<https://www.youtube.com/watch?v=viZrOnJclY0>

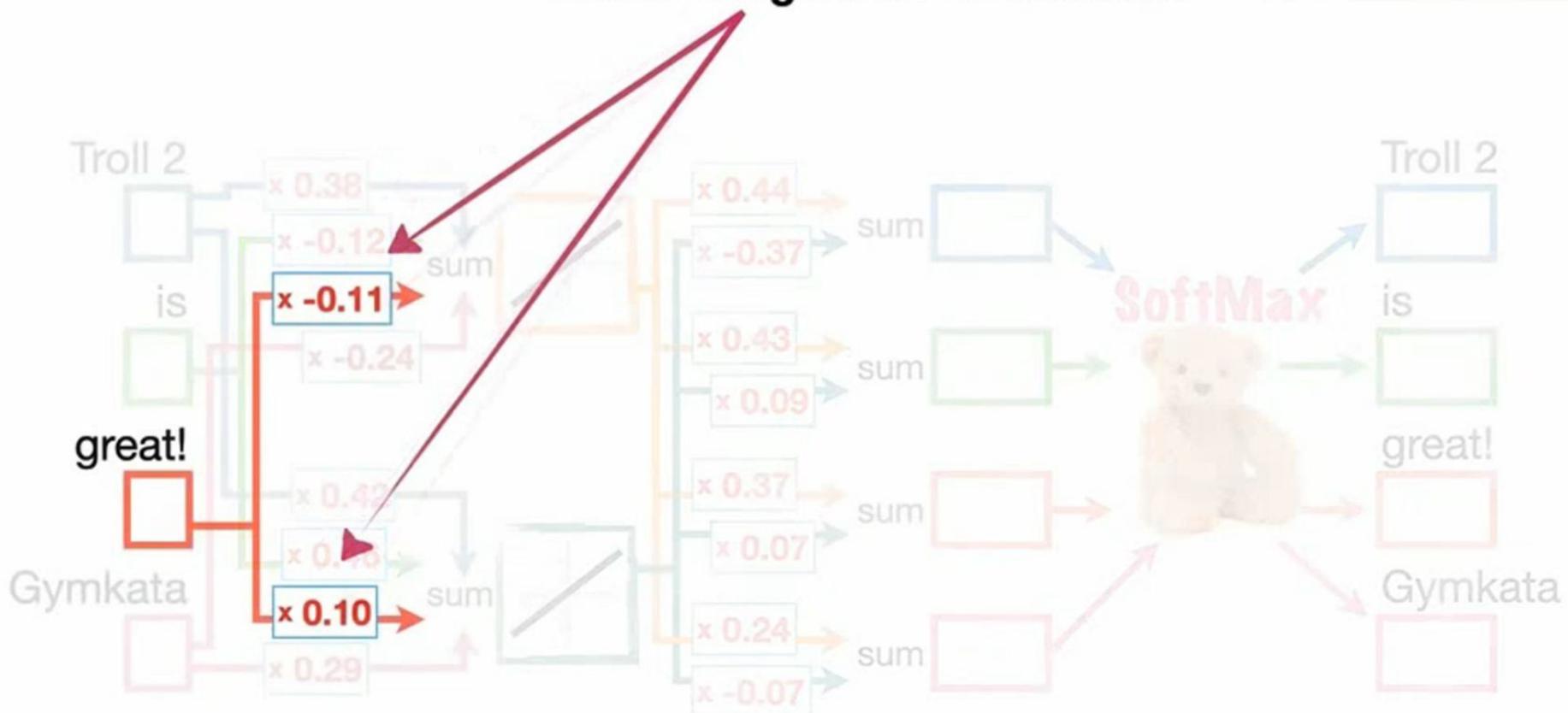
Word Embeddings: Word2Vec



Training Data

Troll 2 is great!
Gymkata is great!

And since, in this example, we have **2 Weights** for each word...



Word Embedding and Word2Vec, Clearly Explained!!!

<https://www.youtube.com/watch?v=viZrOnJclY0>

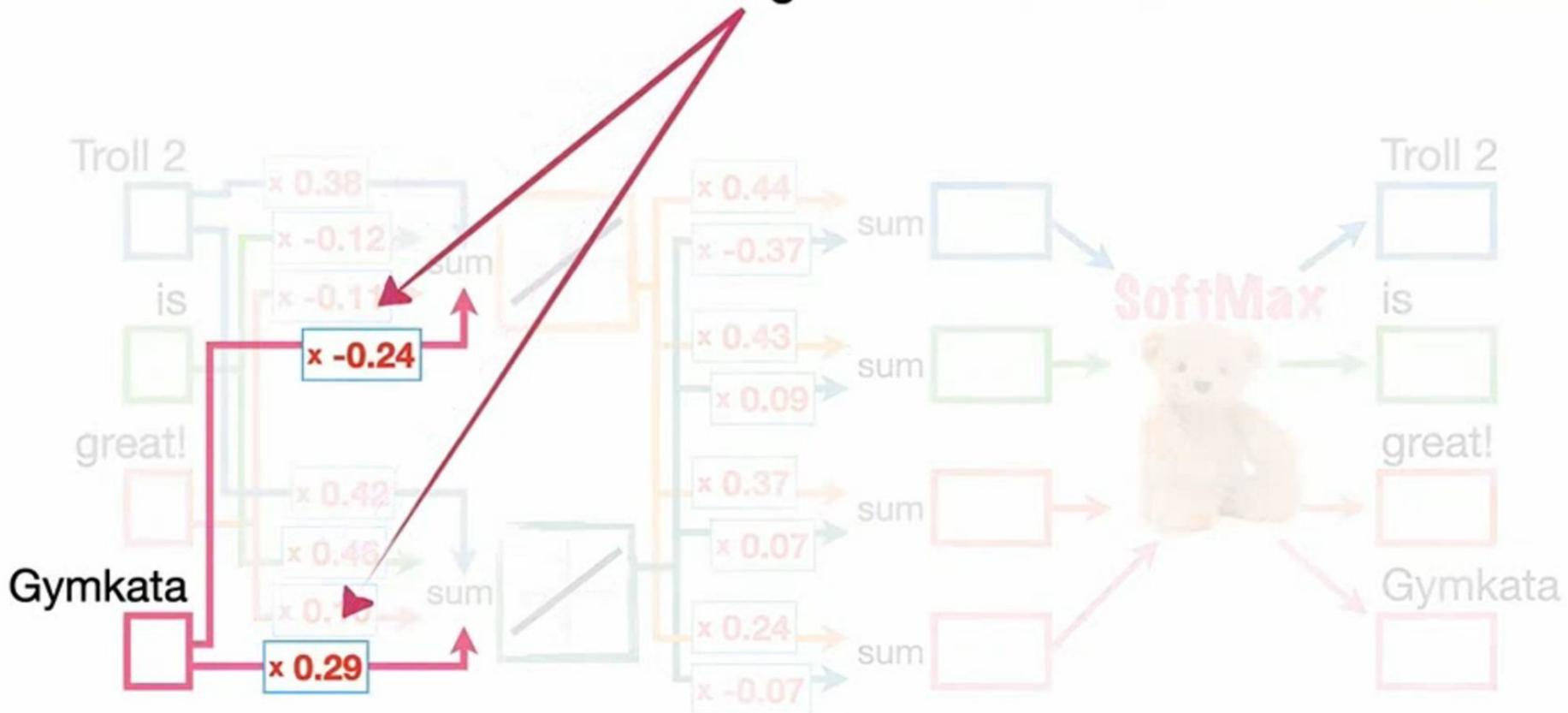
Word Embeddings: Word2Vec



Training Data

Troll 2 is great!
Gymkata is great!

And since, in this example, we have **2 Weights** for each word...



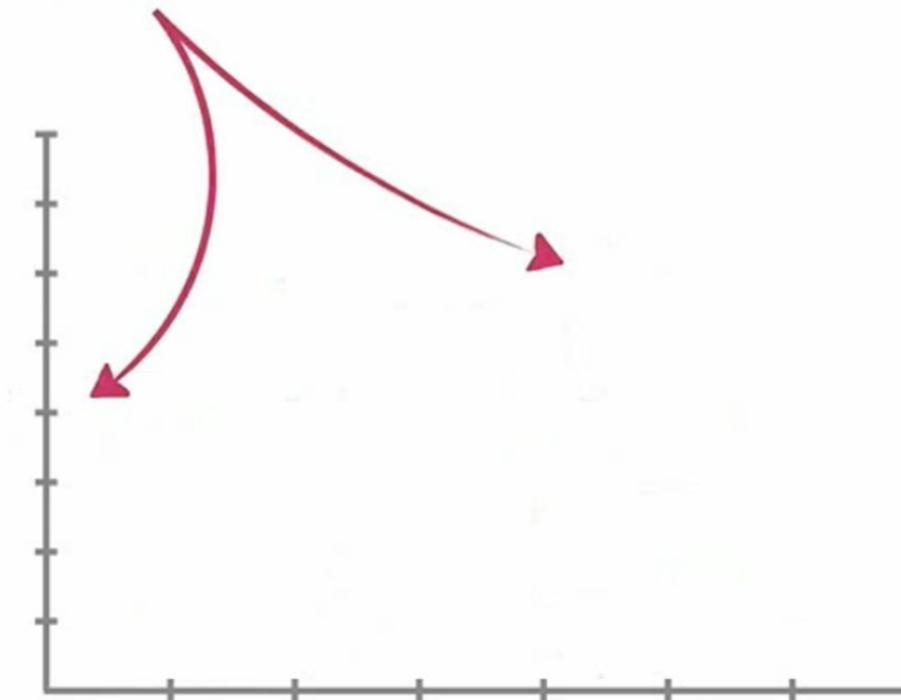
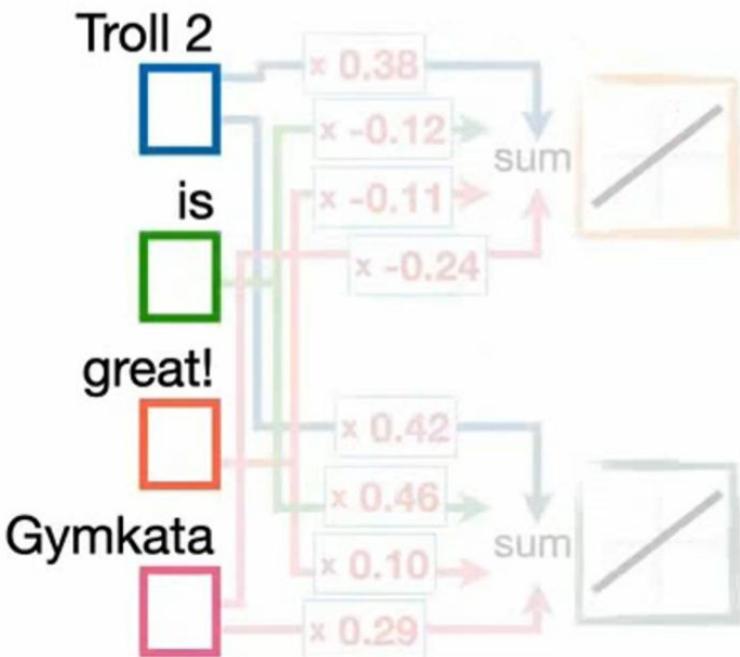
Word Embedding and Word2Vec, Clearly Explained!!!

<https://www.youtube.com/watch?v=viZrOnJclY0>

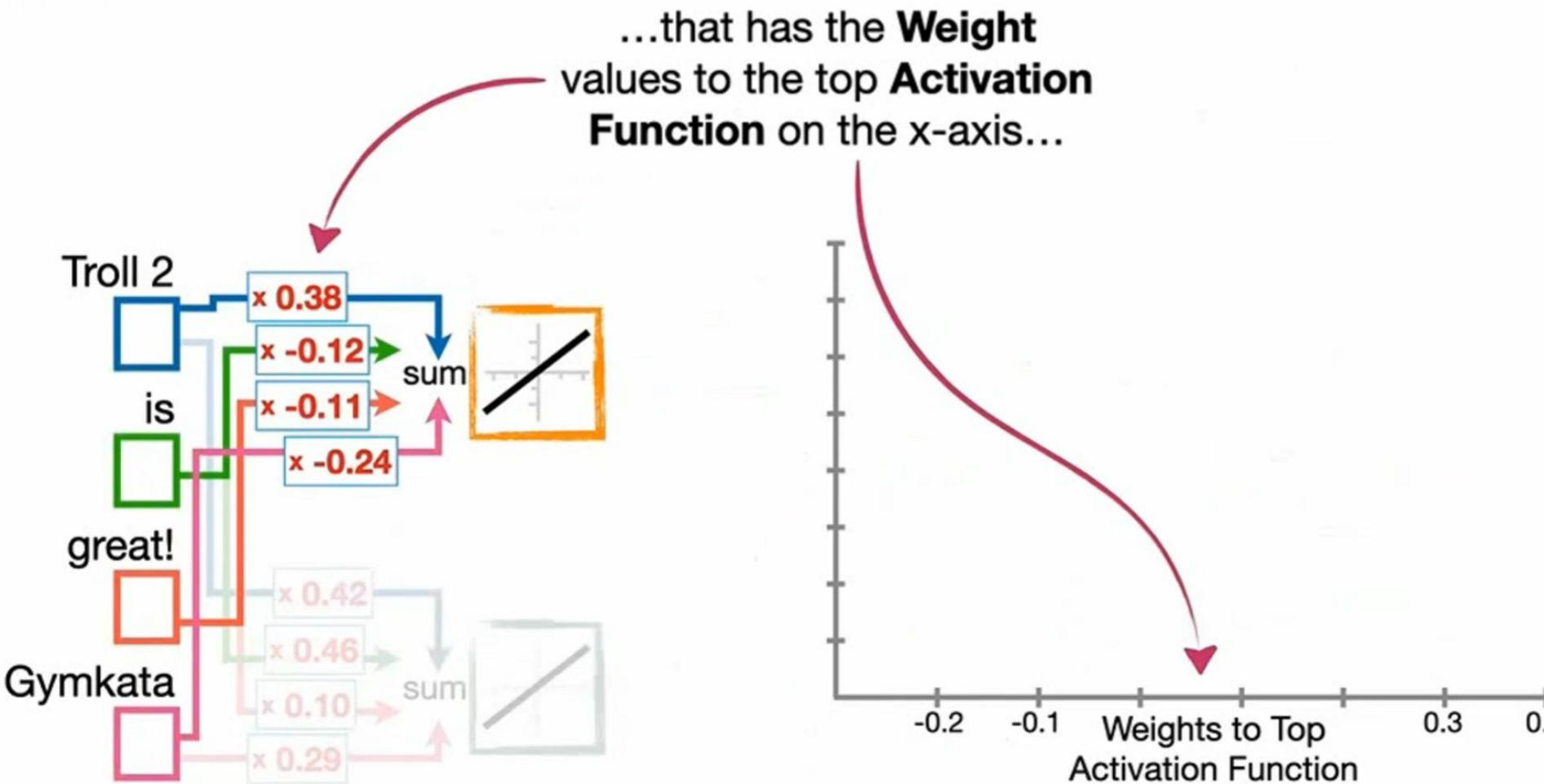
Word Embeddings: Word2Vec



...we can plot each word on a graph...



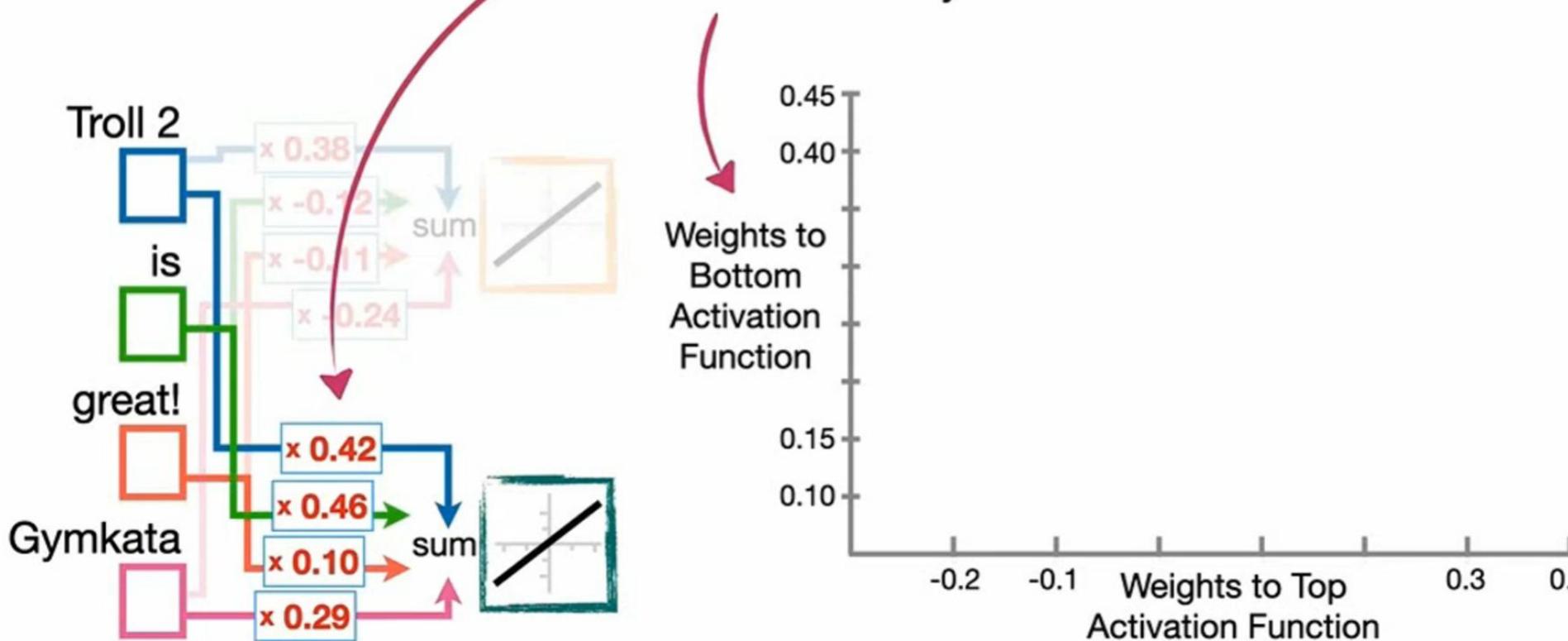
Word Embeddings: Word2Vec



Word Embeddings: Word2Vec



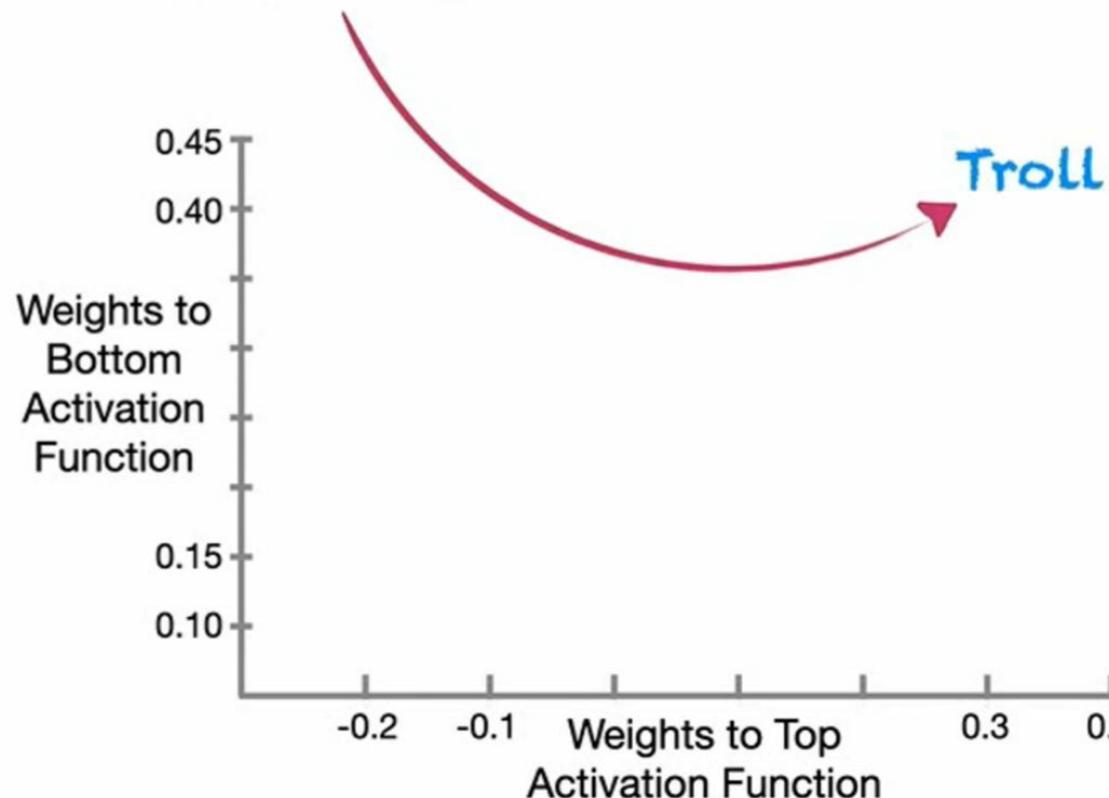
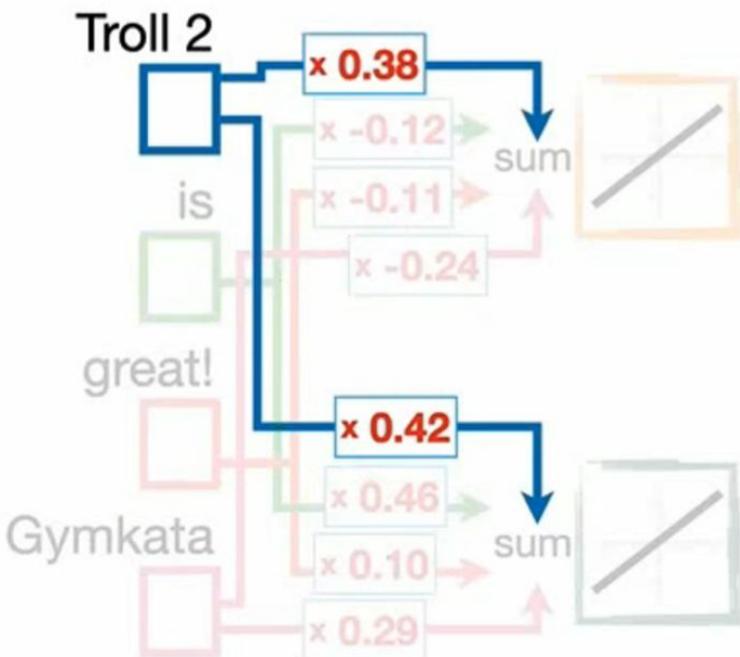
...and the **Weight** values to
the bottom **Activation
Function** on the y-axis.



Word Embeddings: Word2Vec



For example, the word
Troll 2 goes here...

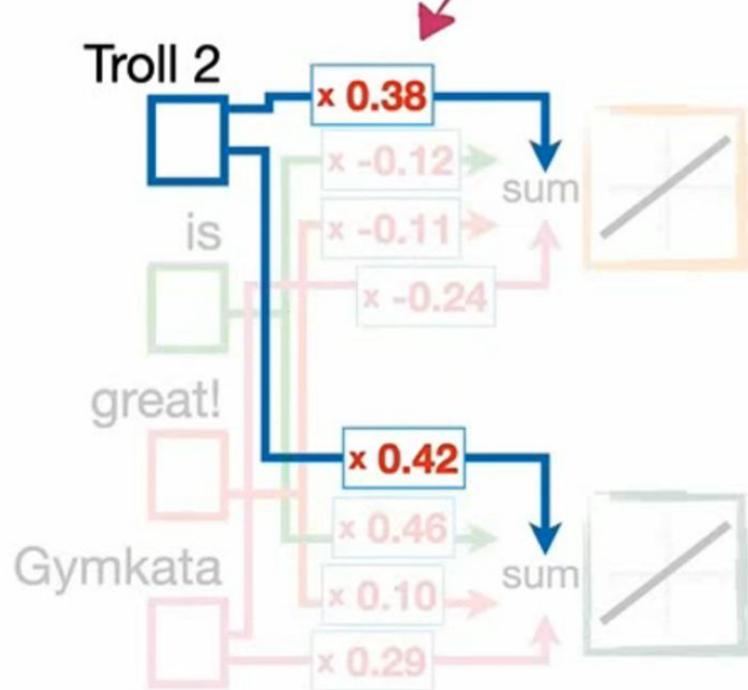


Word Embedding and Word2Vec, Clearly Explained!!!

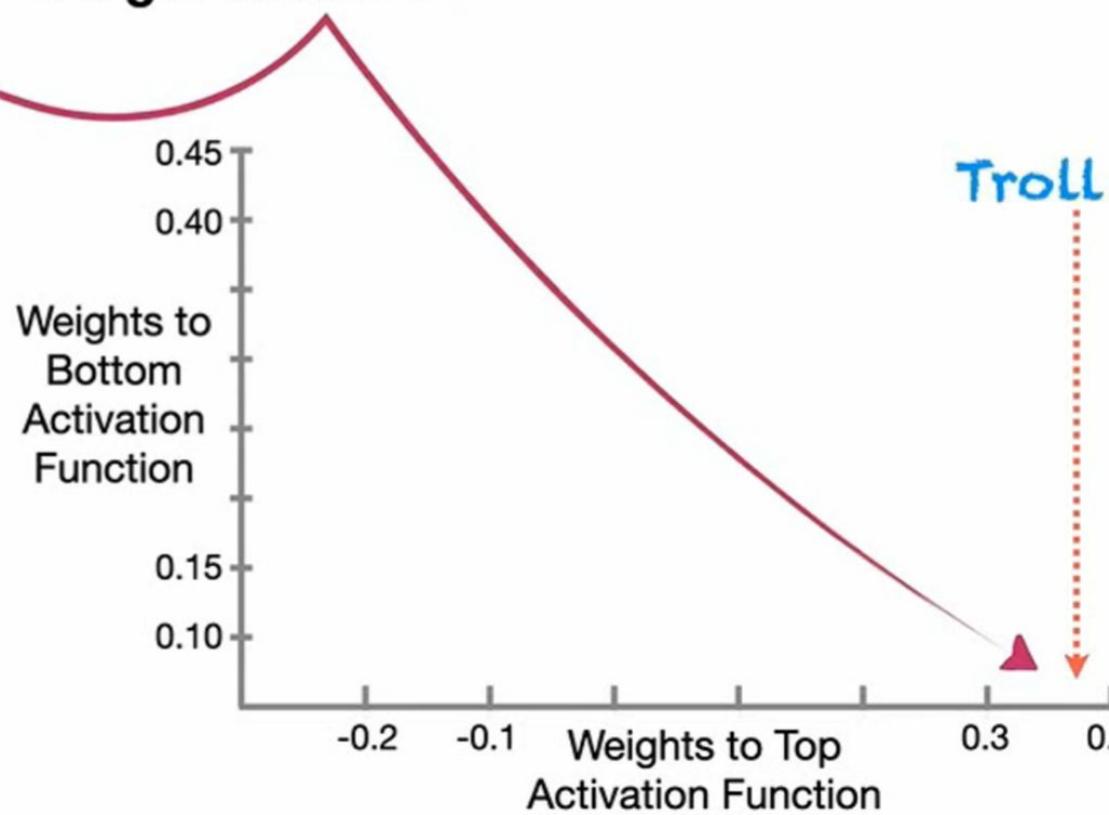
Word Embedding and Word2Vec, Clearly Explained!!!

<https://www.youtube.com/watch?v=viZrOnJclY0>

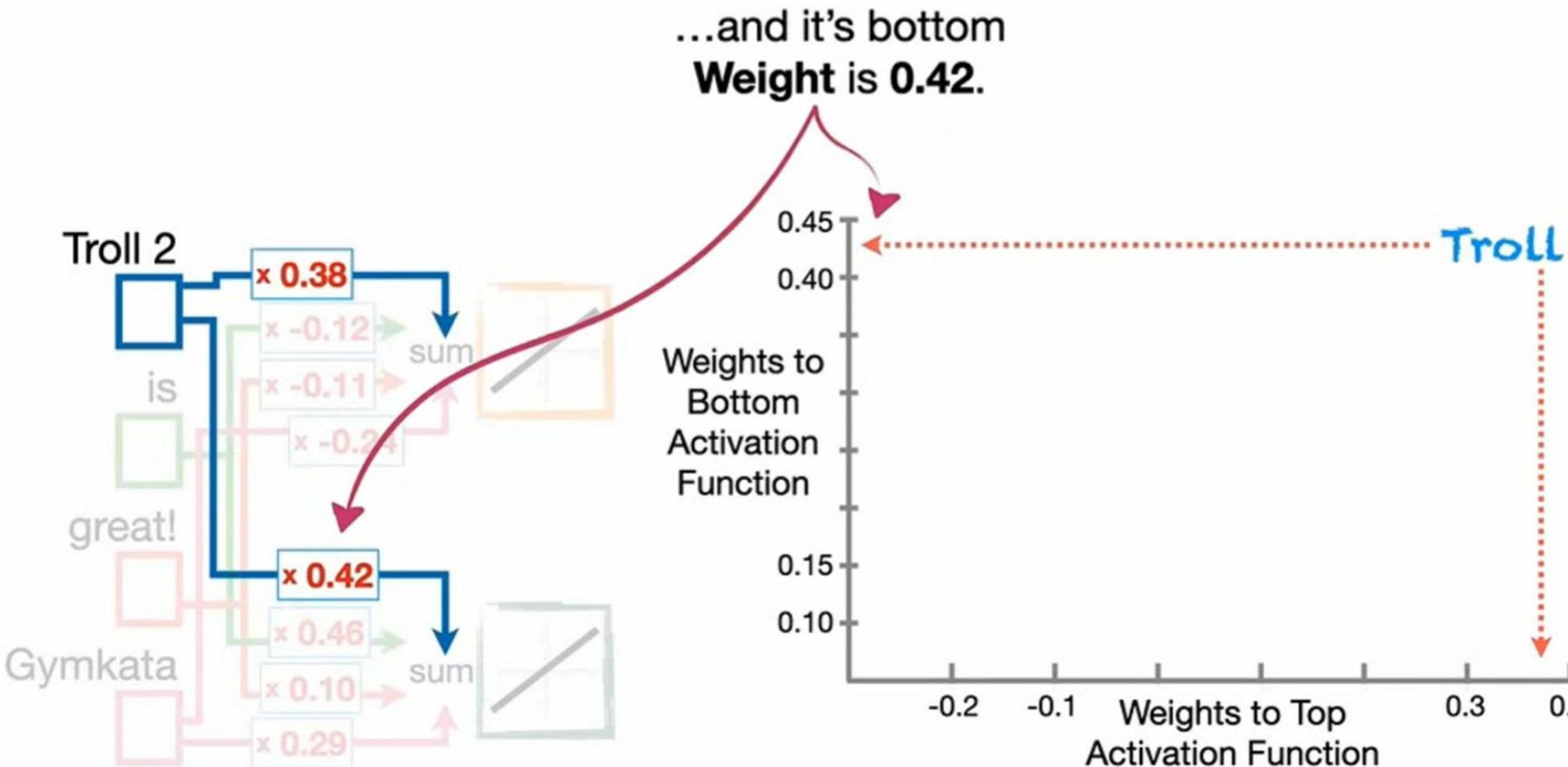
<https://www.youtube.com/watch?v=vj7rOnJclY0>



...because its top
Weight is 0.38...



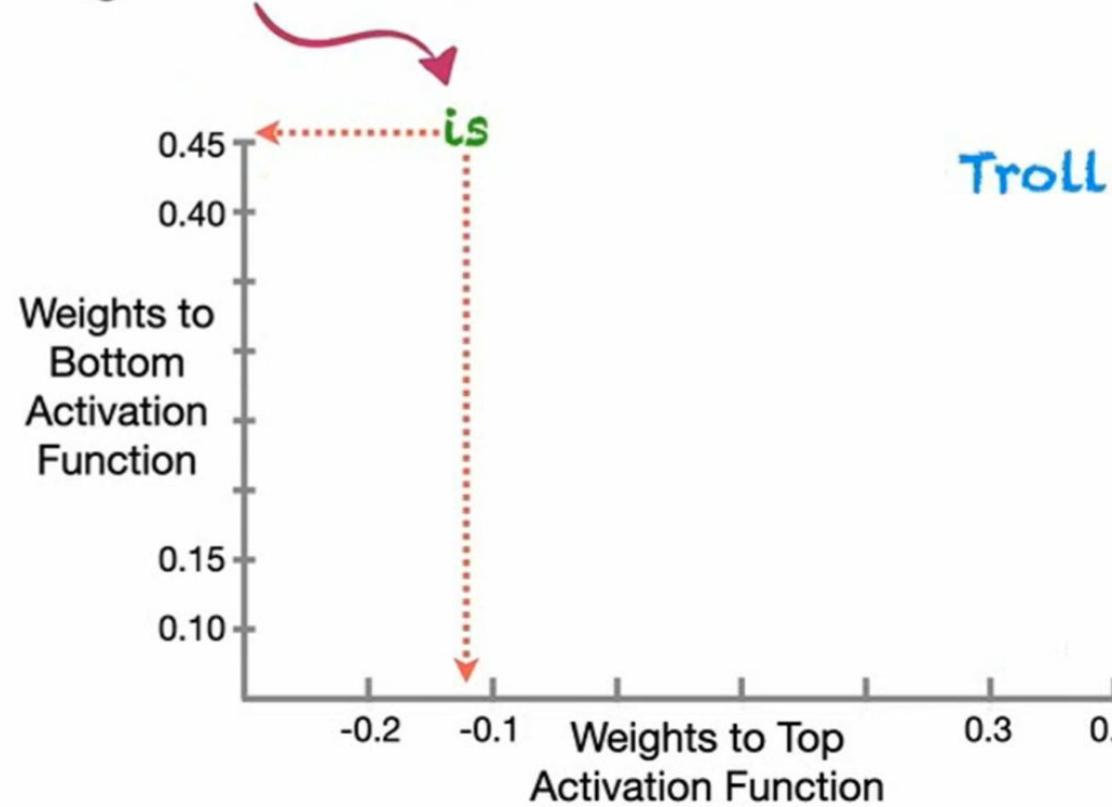
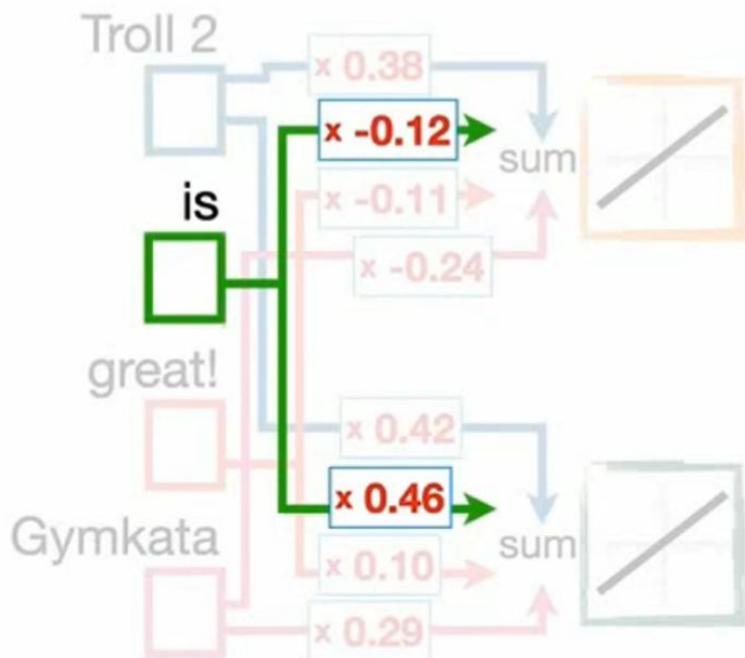
Word Embeddings: Word2Vec



Word Embedding and Word2Vec, Clearly Explained!!!
Word Embedding and Word2Vec, Clearly Explained!!!
<https://www.youtube.com/watch?v=viZrOnJcIY0>
<https://www.youtube.com/watch?v=vj7rOnIcIY0>



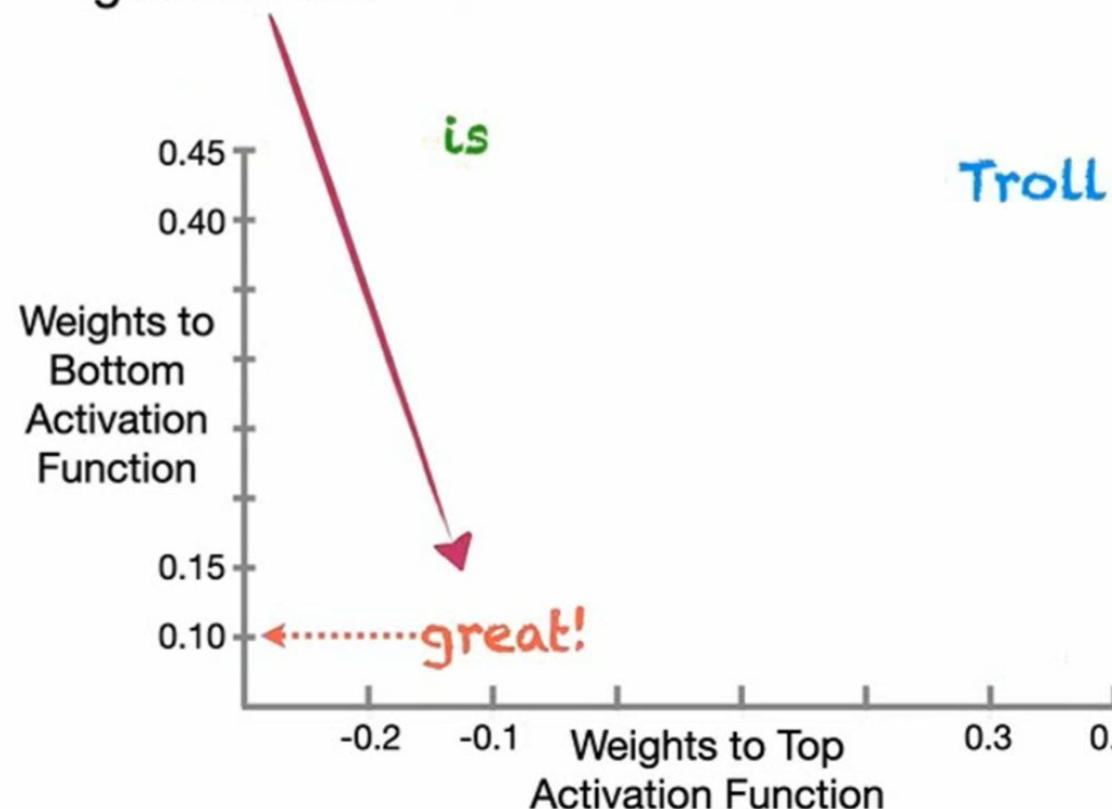
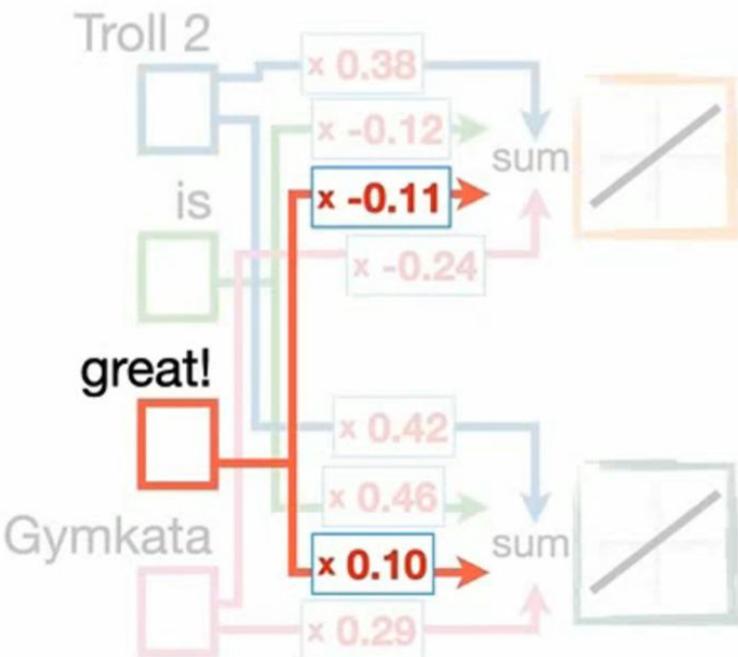
Likewise, the word **is** goes here...



Word Embeddings: Word2Vec

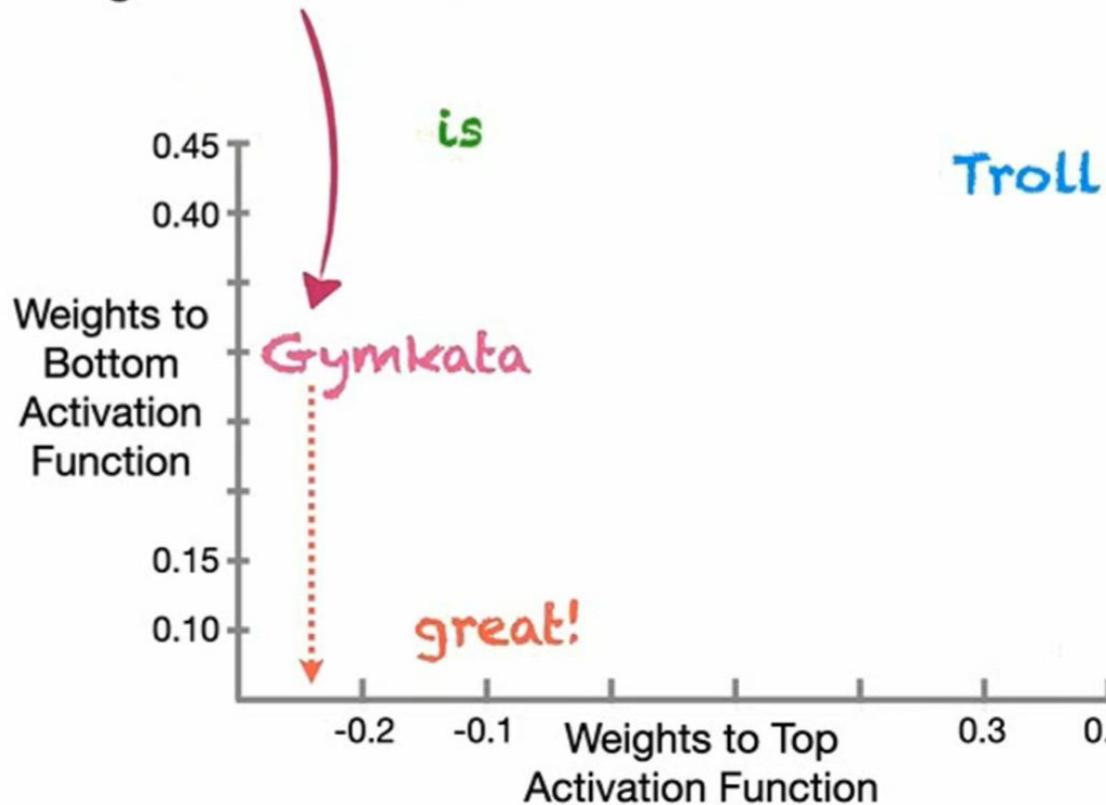
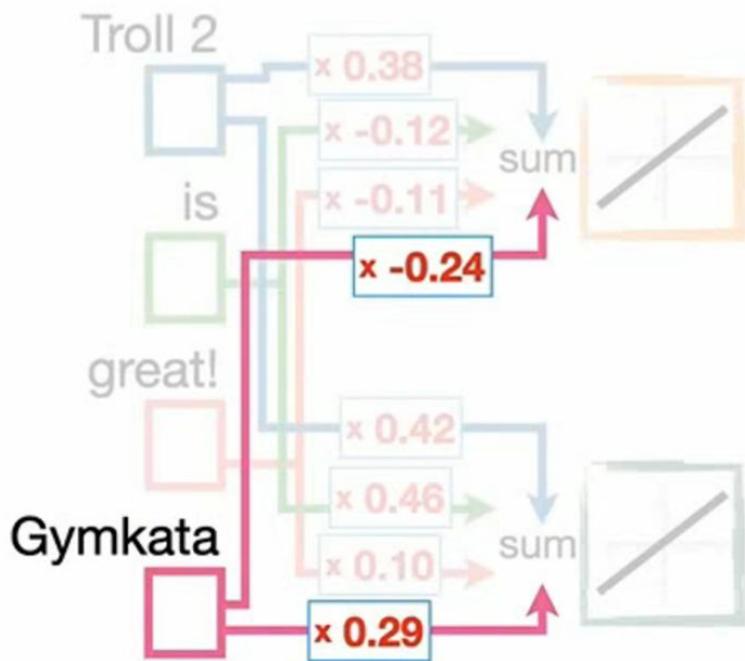


...and the word **great!**
goes here...



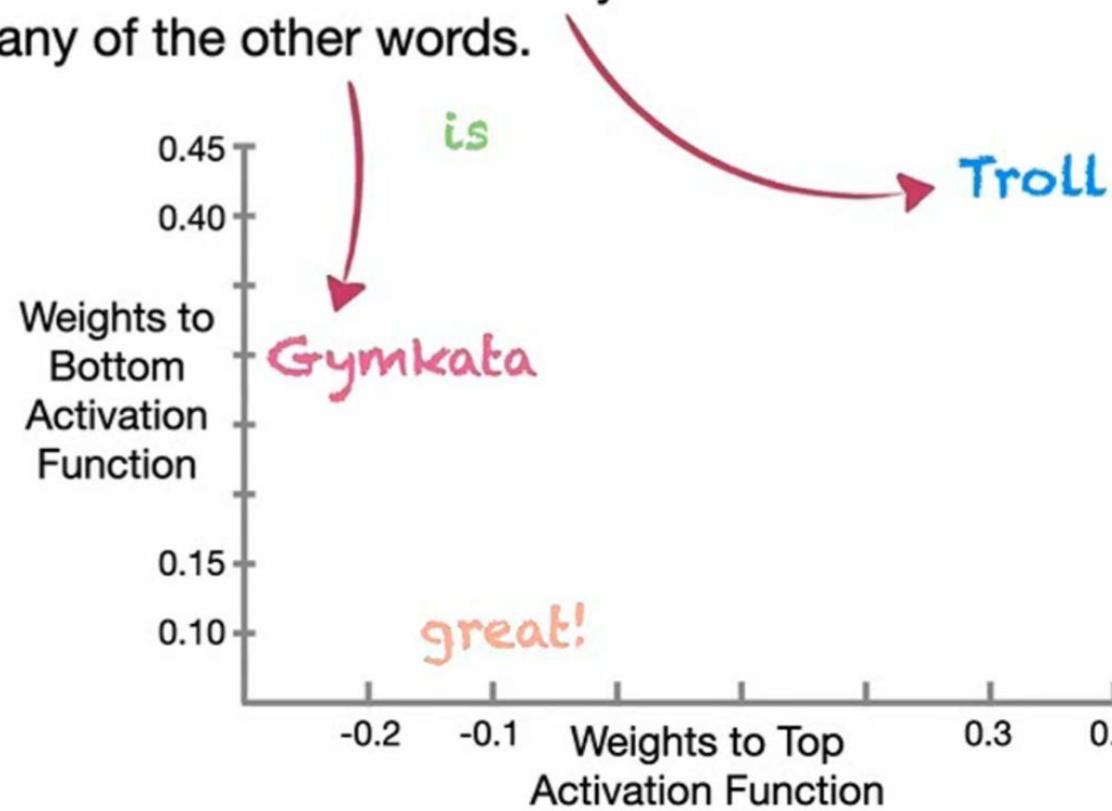
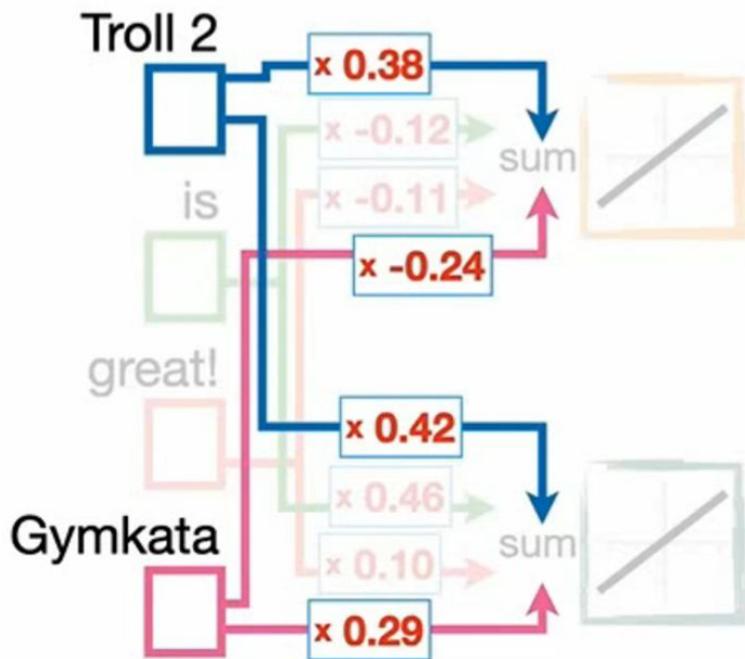


...and **Gymkata**
goes here.





Now, with this graph, we see that the words **Troll 2** and **Gymkata** are currently no more similar to each other as they are to any of the other words.



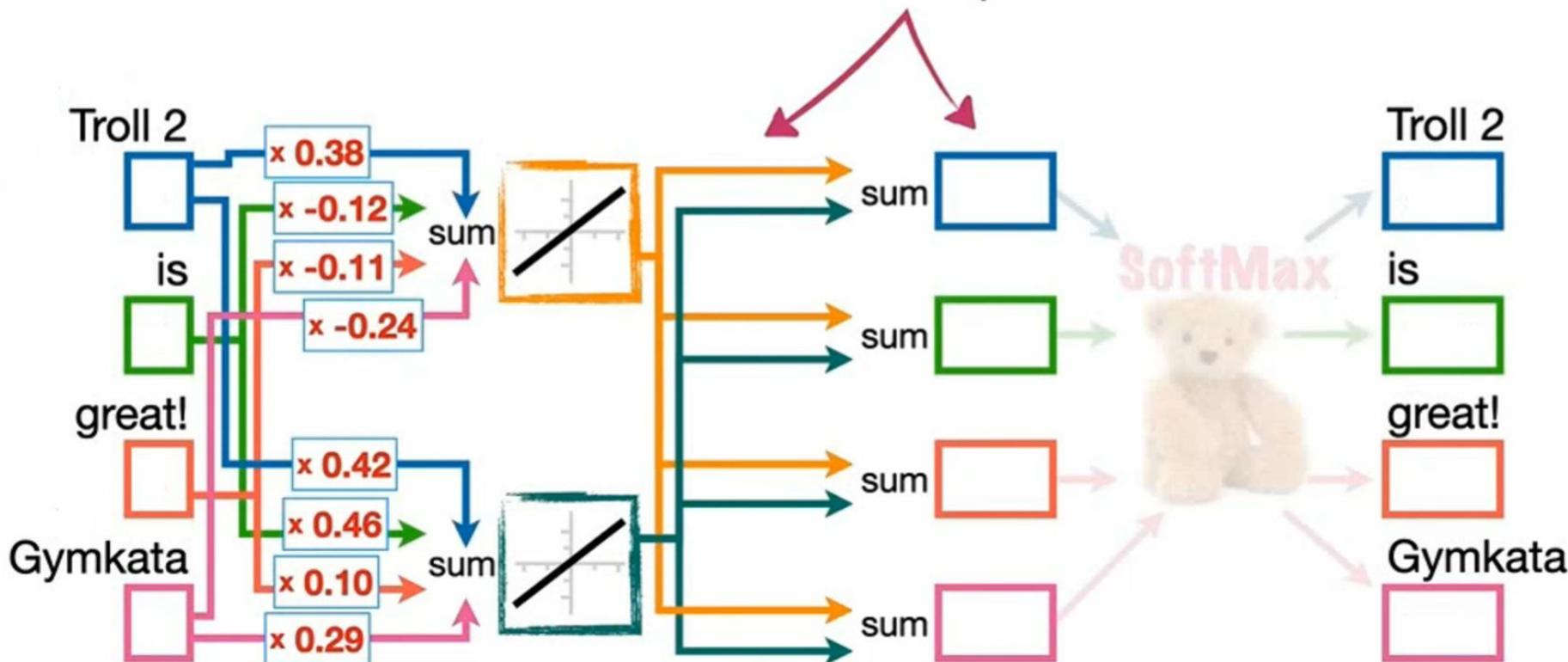
Word Embeddings: Word2Vec



In order to make these predictions,
we connect the **Activation Functions** to outputs...

Training Data

Troll 2 is great!
Gymkata is great!



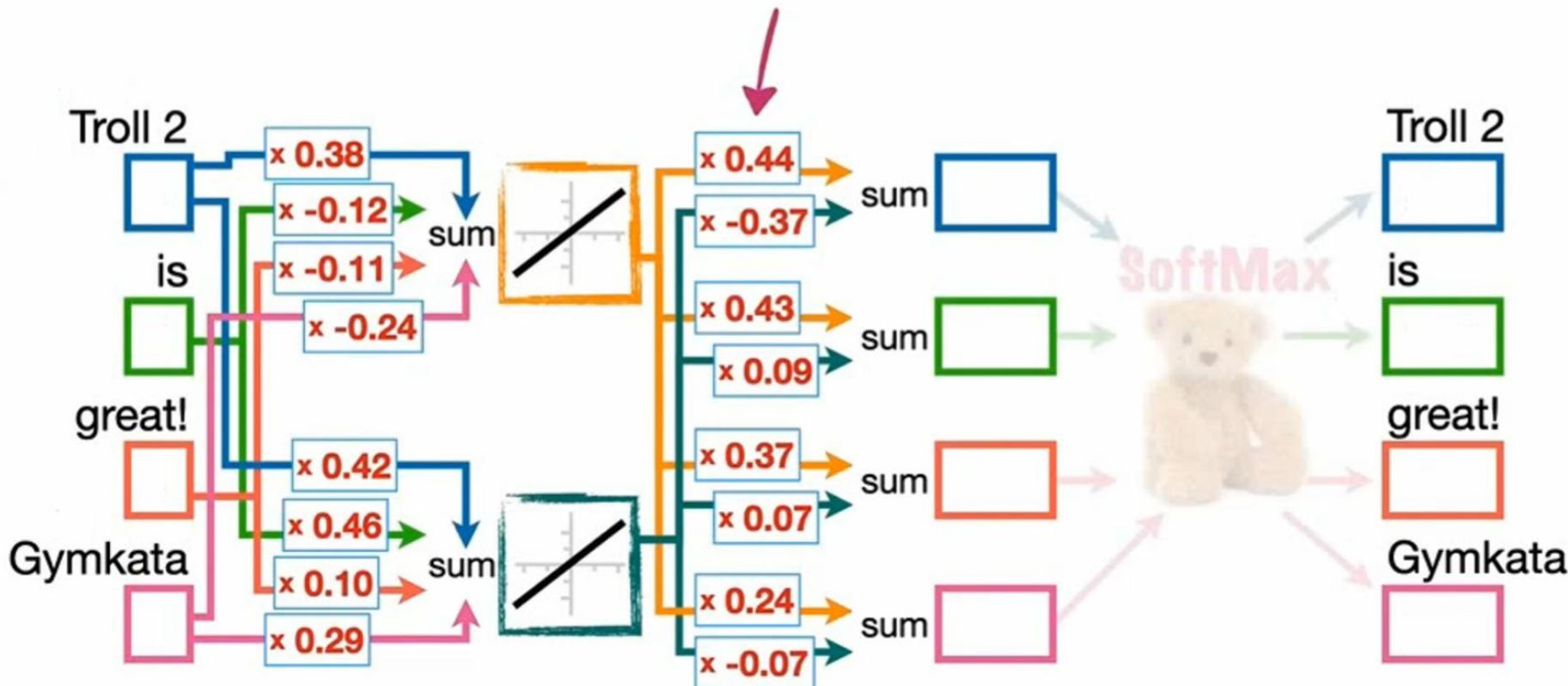
Word Embeddings: Word2Vec



...and we add **Weights** to those connections with random initialization values...

Training Data

Troll 2 is great!
Gymkata is great!



Word Embedding and Word2Vec, Clearly Explained!!!

<https://www.youtube.com/watch?v=viZrOnJclY0>

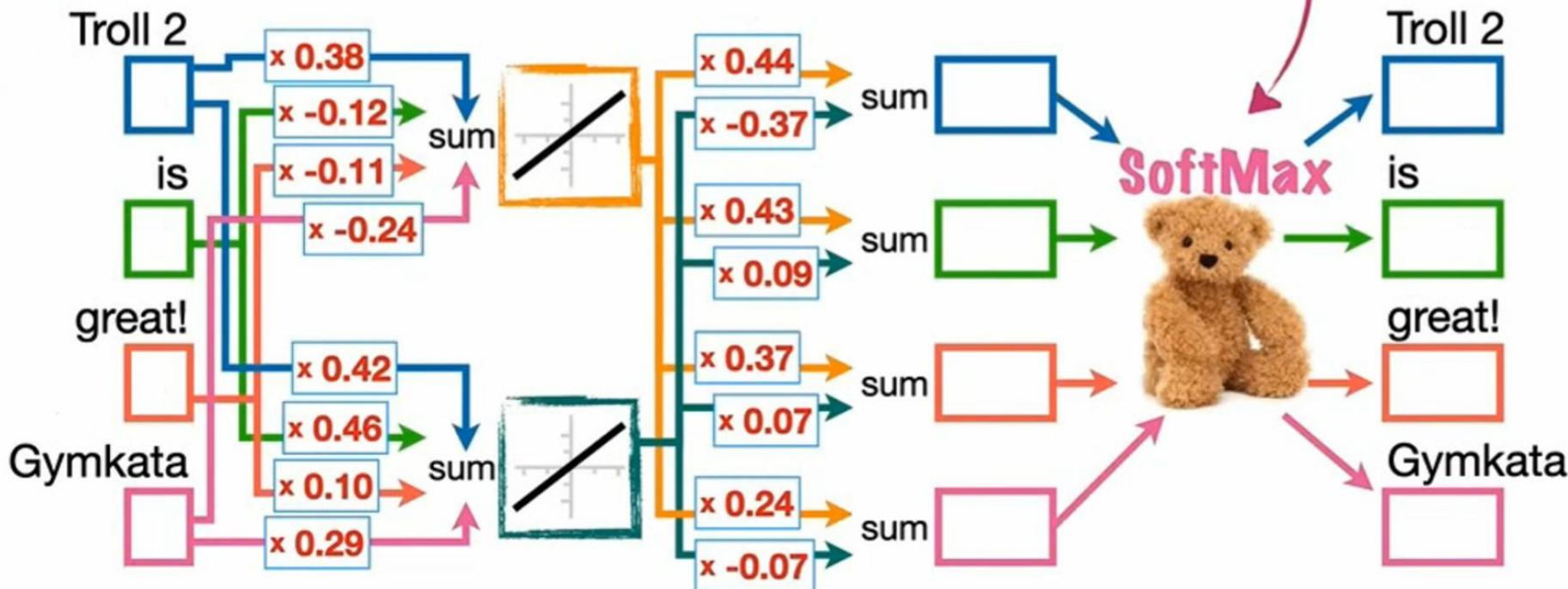
Word Embeddings: Word2Vec



...and then we run the outputs through the **SoftMax** function because we have multiple outputs for classification...

Training Data

Troll 2 is great!
Gymkata is great!



Word Embedding and Word2Vec, Clearly Explained!!!

<https://www.youtube.com/watch?v=viZrOnJclY0>

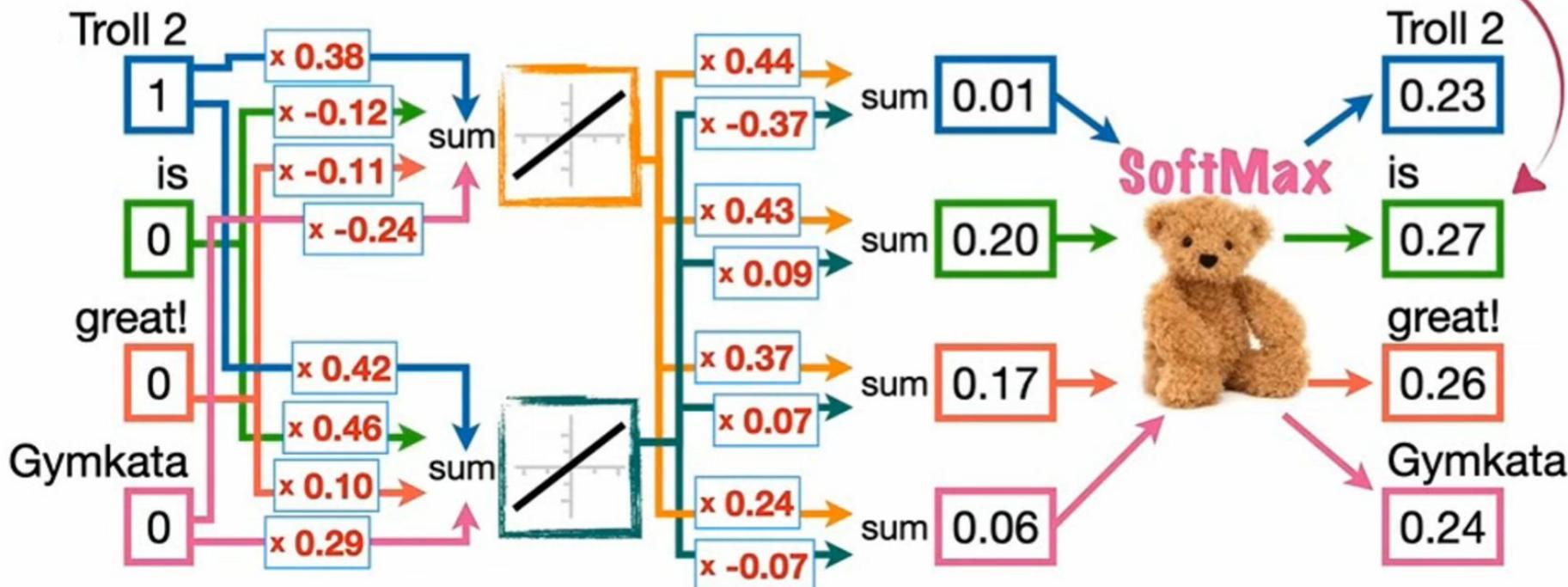
Word Embeddings: Word2Vec



...we get lucky and correctly predict the next word, **is**, but just barely.

Training Data

Troll 2 is great!
Gymkata is great!

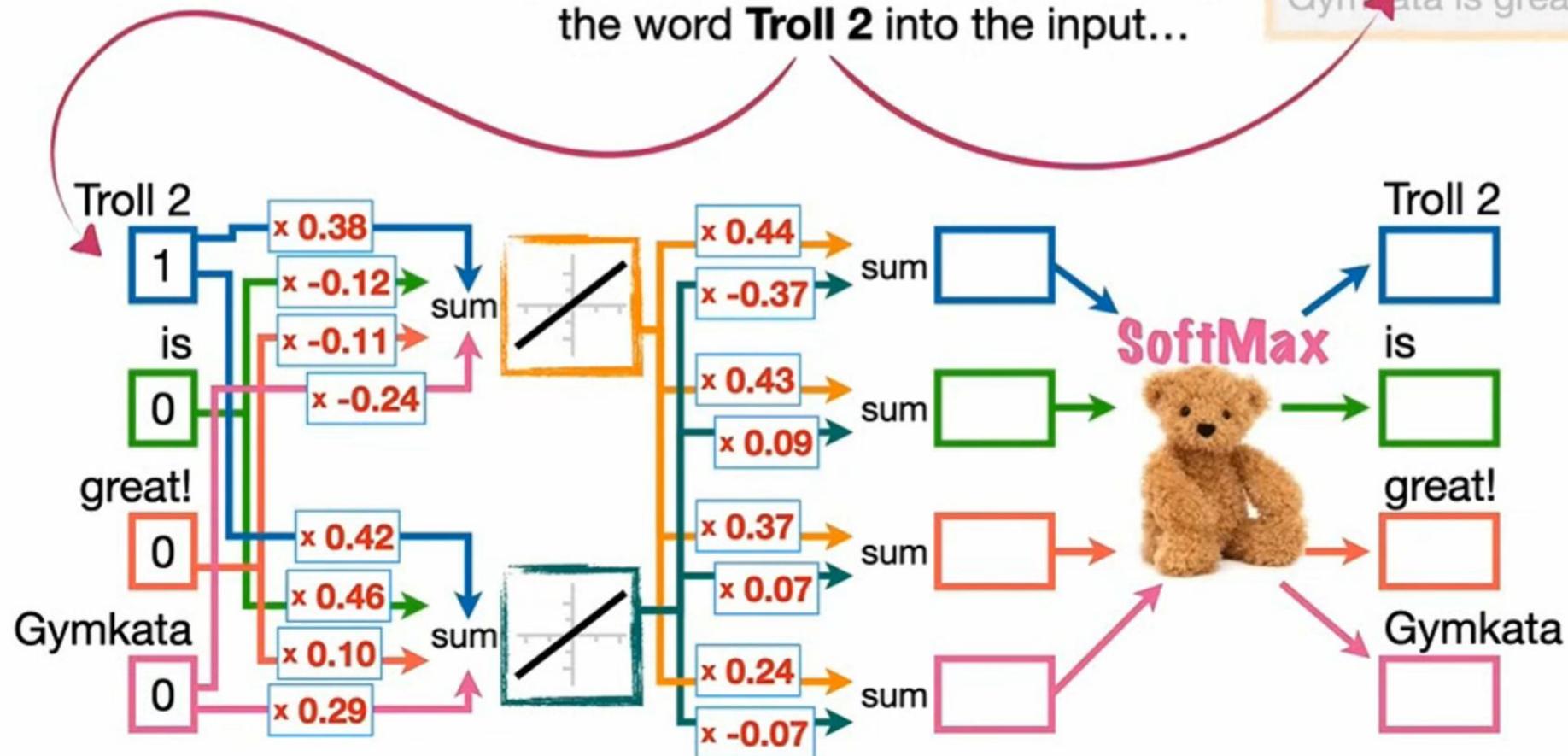




And now, before training, if we plug
the word **Troll 2** into the input...

Training Data

Troll 2 is great!
Gymkata is great!



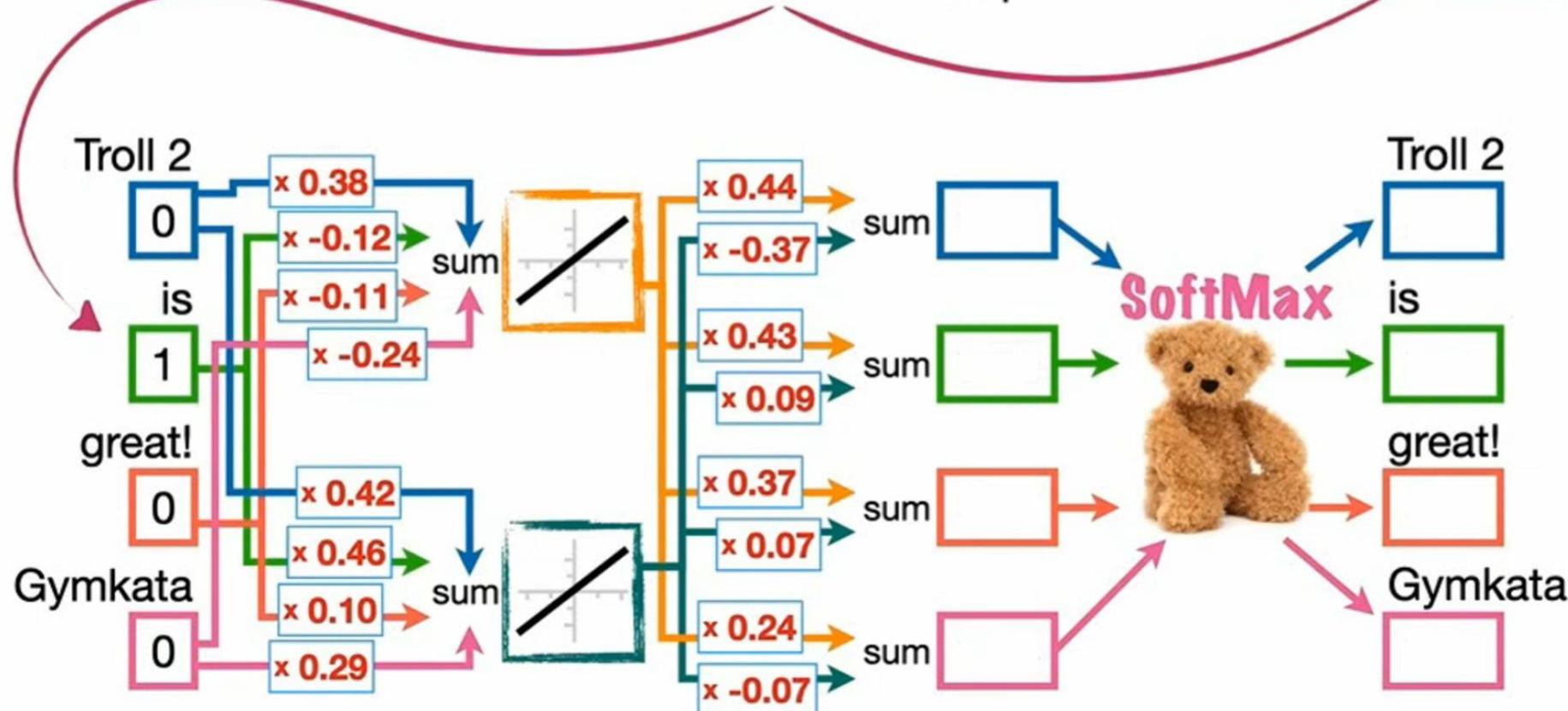
Word Embeddings: Word2Vec



However, when we plug the word **is** into the input...

Training Data

Troll 2 **is** great!
Gymkata **is** great!



Word Embedding and Word2Vec, Clearly Explained!!!

<https://www.youtube.com/watch?v=viZrOnJclY0>

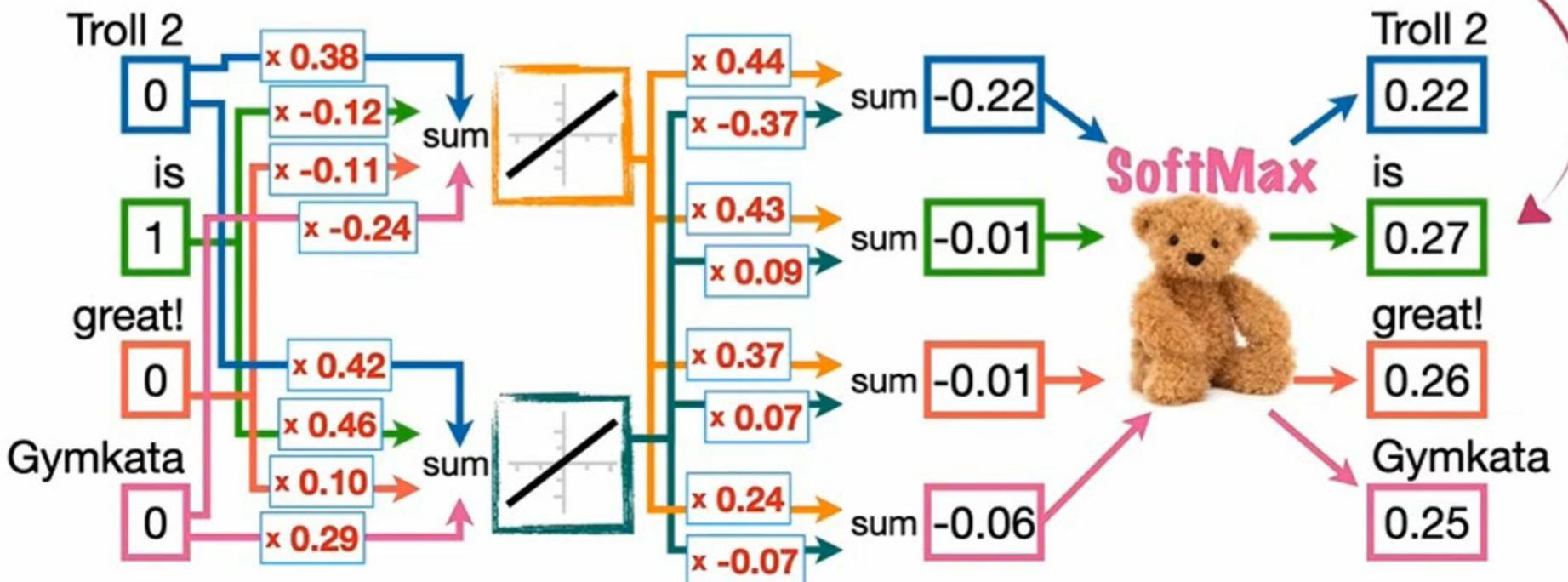
Word Embeddings: Word2Vec



...then we fail to correctly predict the next word, **great!**, and instead predict **is**.

Training Data

Troll 2 is great!
Gymkata is great!



Word Embedding and Word2Vec, Clearly Explained!!!

Word Embedding and Word2Vec, Clearly Explained!!!

<https://www.youtube.com/watch?v=viZrOnJcIY0>

<https://www.youtube.com/watch?v=vj7rOnIcIY0>

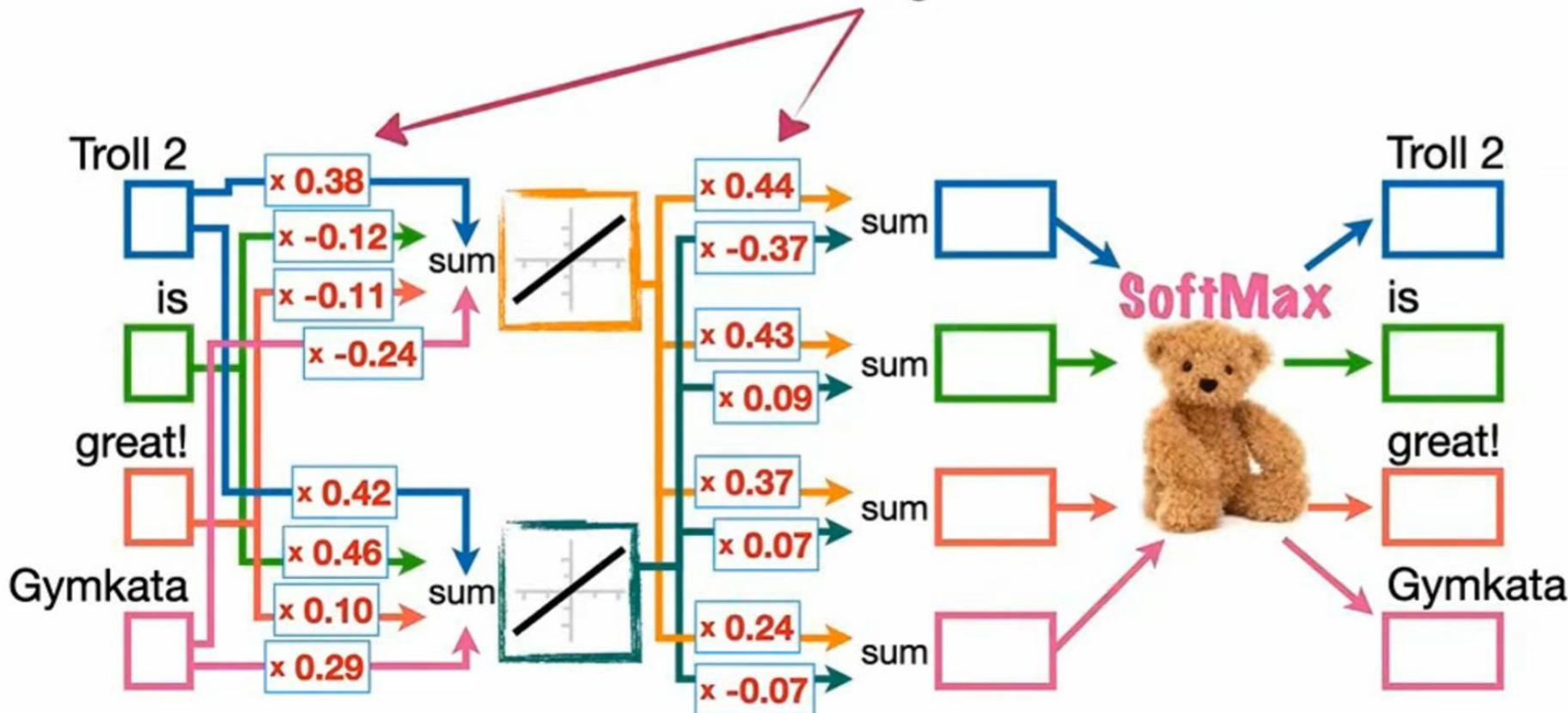
Word Embeddings: Word2Vec



So we start with
these **Weights**...

Training Data

Troll 2 is great!
Gymkata is great!



Word Embedding and Word2Vec, Clearly Explained!!!

<https://www.youtube.com/watch?v=viZrOnJclY0>

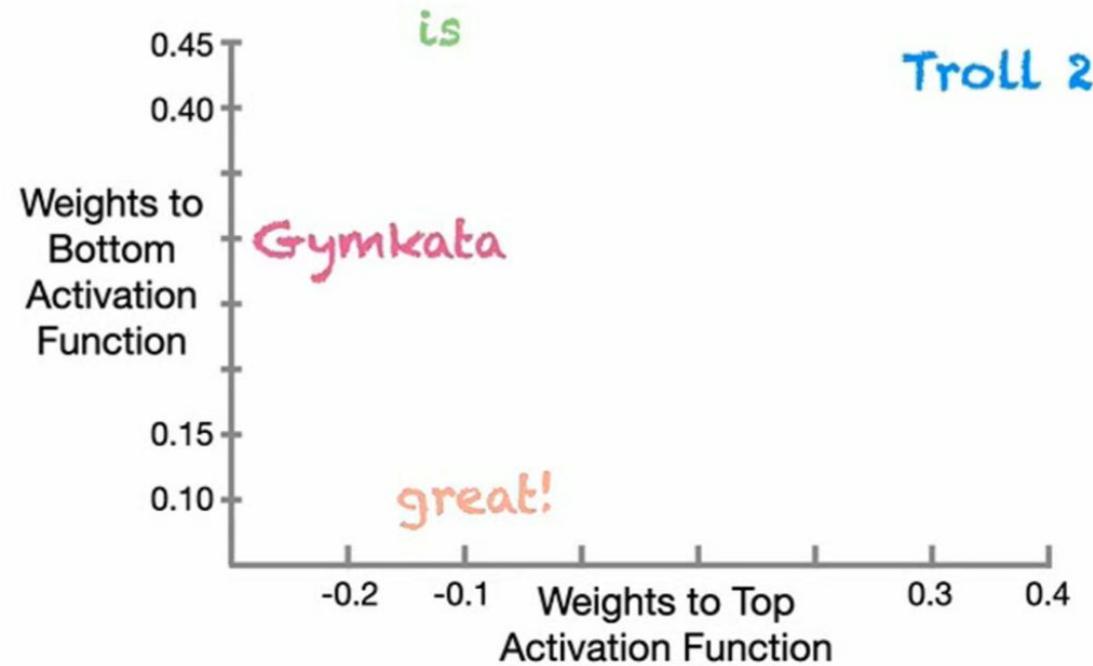
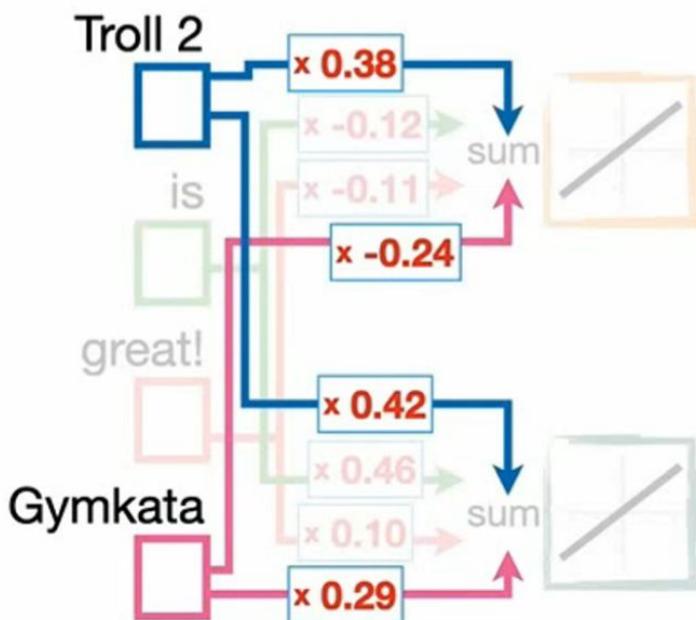


...we hope that

Backpropagation will make
their **Weights** more similar.

Training Data

Troll 2 is great!
Gymkata is great!



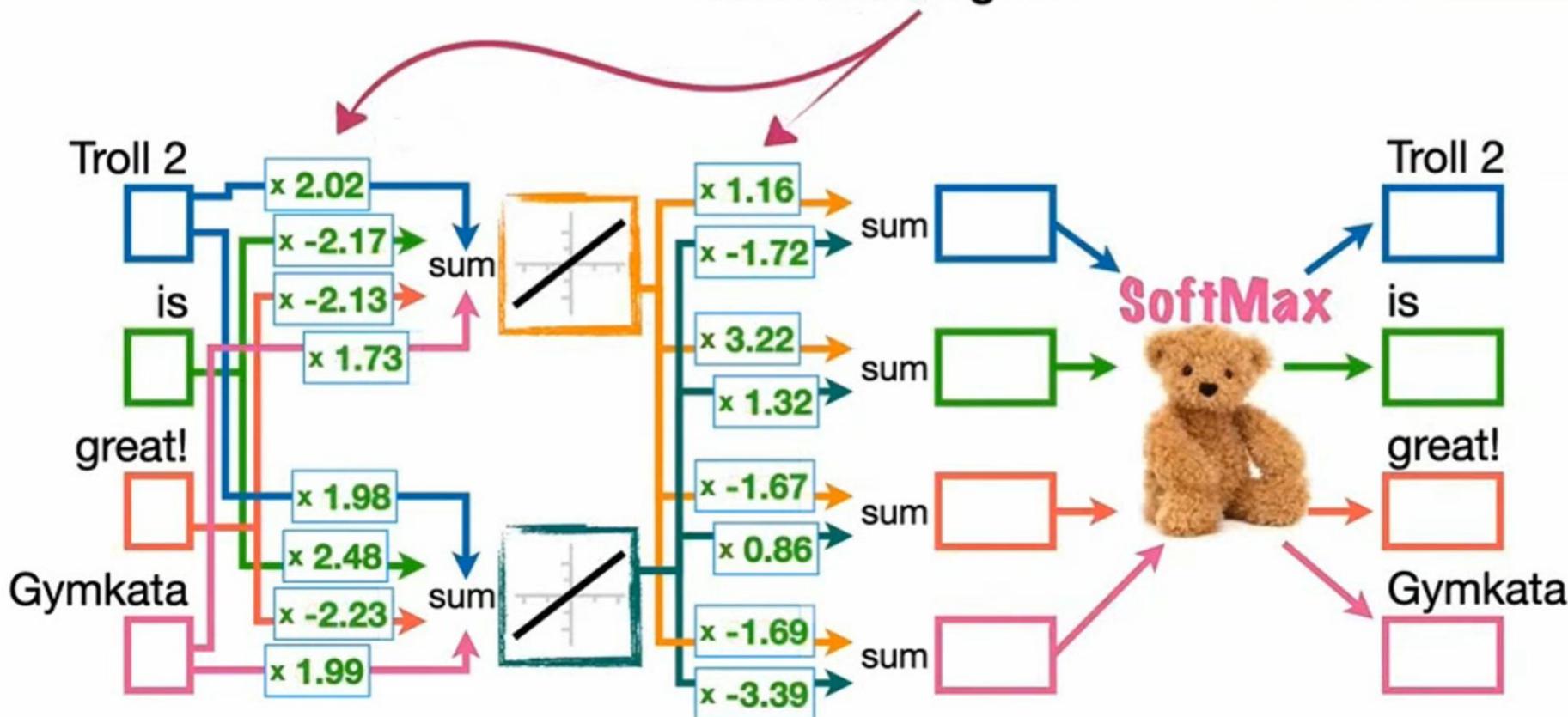
Word Embeddings: Word2Vec



...and we end up with
these new **Weights**.

Training Data

Troll 2 is great!
Gymkata is great!



Word Embedding and Word2Vec, Clearly Explained!!!

<https://www.youtube.com/watch?v=viZrOnJclY0>

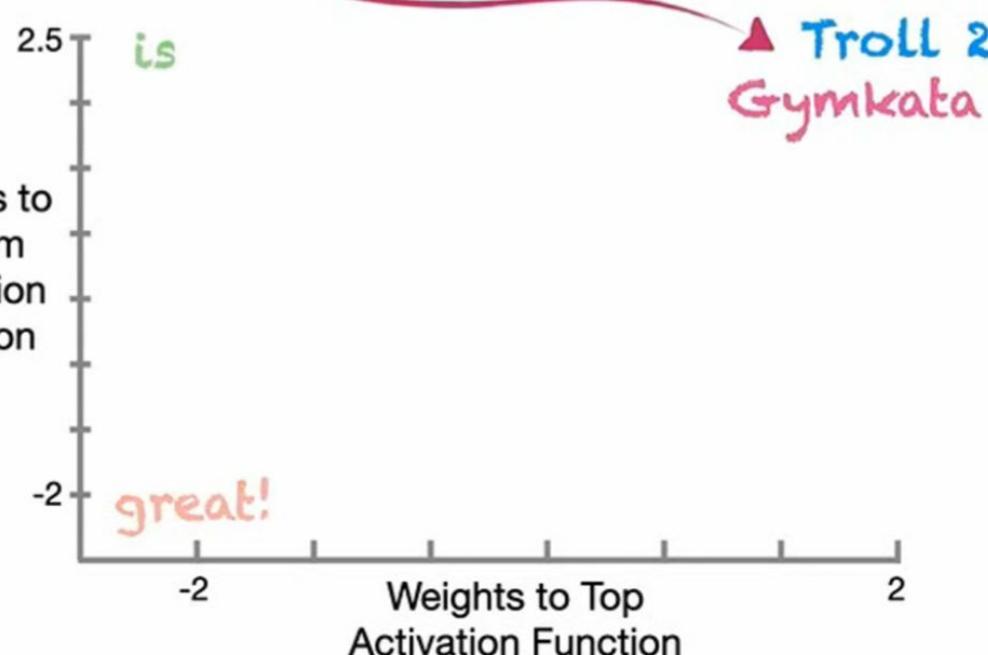
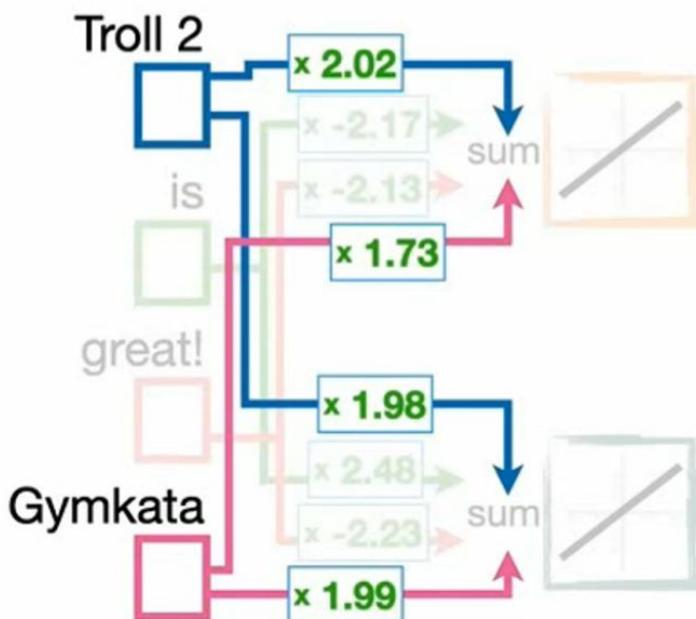
Word Embeddings: Word2Vec



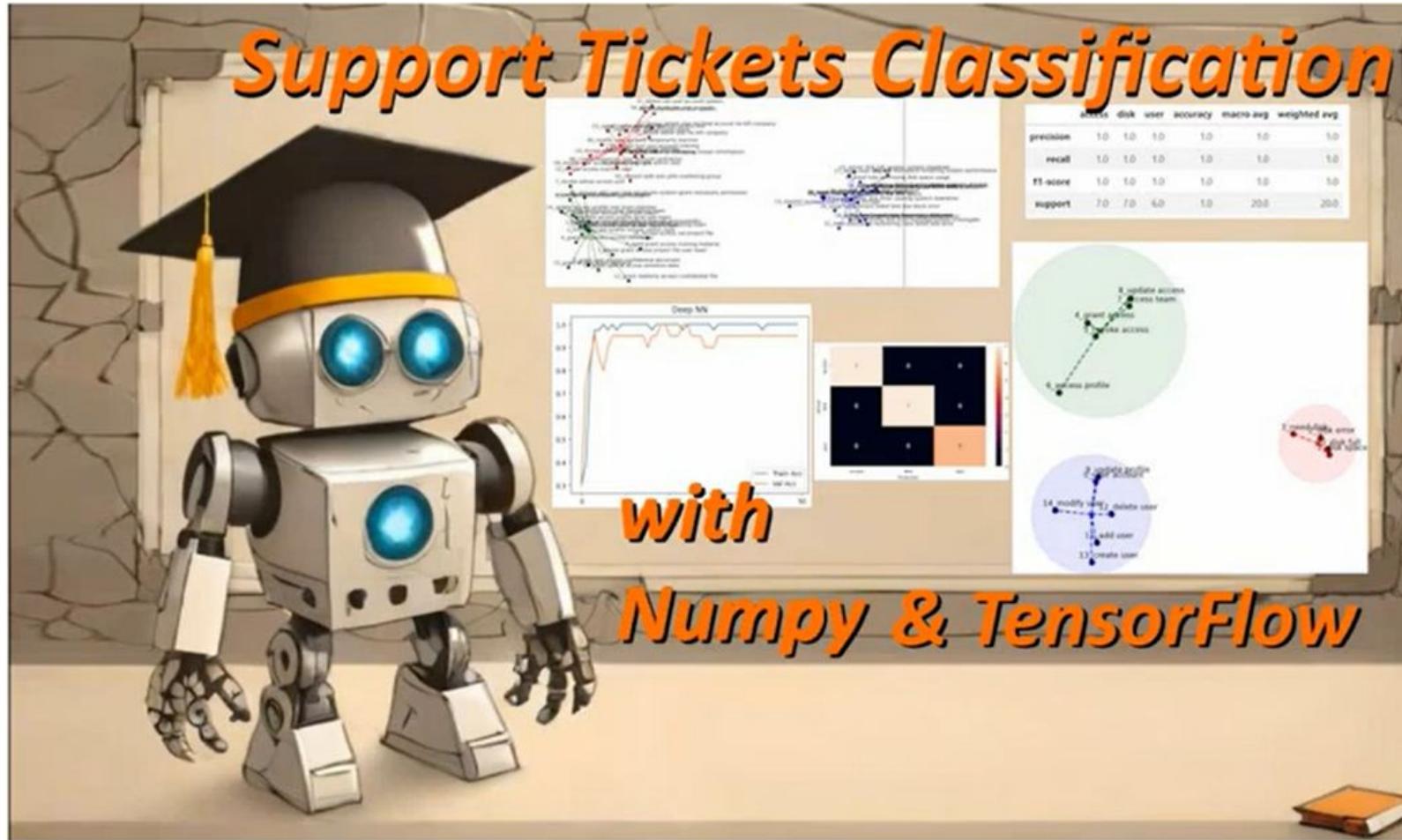
...we see that **Troll 2** and **Gymkata** are now relatively close to each other compared to the other words in the training data.

Training Data

Troll 2 is great!
Gymkata is great!



Word Embeddings with Sentence Transformers



Support Ticket Classification

Github: https://github.com/enoten/support_ticket_analysis/blob/main/support_tickets_classification.ipynb

Youtube: <https://www.youtube.com/watch?v=NIJ1yS0F03M>

Word Embeddings with Sentence Transformers

3. Transform text: embedding, dim reduction

3.0 Import LLM library

In [33]:

```
import warnings
from sentence_transformers import SentenceTransformer
warnings.filterwarnings("ignore")
```

3.1 Create embeddings for preprocessed description

In [34]:

```
phrase_model = SentenceTransformer("all-mpnet-base-v2")
```

In [35]:

```
#convert df column to list
preproc_desc = tickets_df['preproc_description'].to_list()
```

In [36]:

```
embeds = phrase_model.encode(preproc_desc)
```

In [37]:

```
embeds.shape
```

Out[37]: (60, 768)

Support Ticket Classification

https://github.com/enoten/support_ticket_analysis/blob/main/support_tickets_classification.ipynb

Word Embeddings with Sentence Transformers

	subject	description	key_phrase	label	preproc_description	preproc_subject
0	Grant Access to Confidential Files	Grant user access to confidential documents.	grant access	0	grant user access confidential document	grant access confidential file
1	Revoke Access to Sensitive Data	Revoke user access to sensitive information.	revoke access	0	revoke user access sensitive information	revoke access sensitive data
2	Update Access Profile	Modify my access profile to include admin rights	access profile	0	modify access profile include admin right	update access profile
3	Grant Access Request for New Project	Please grant access to project files for users...	grant access	0	please grant access project file user team	grant access request new project
4	Access Removal for Departing Employee	Revoke access for departing employee.	revoke access	0	revoke access departing employee	access removal departing employee

Support Ticket Classification

Github: https://github.com/enoten/support_ticket_analysis/blob/main/support_tickets_classification.ipynb

Youtube: <https://www.youtube.com/watch?v=NlJ1yS0F03M>

Word Embeddings with Sentence Transformers

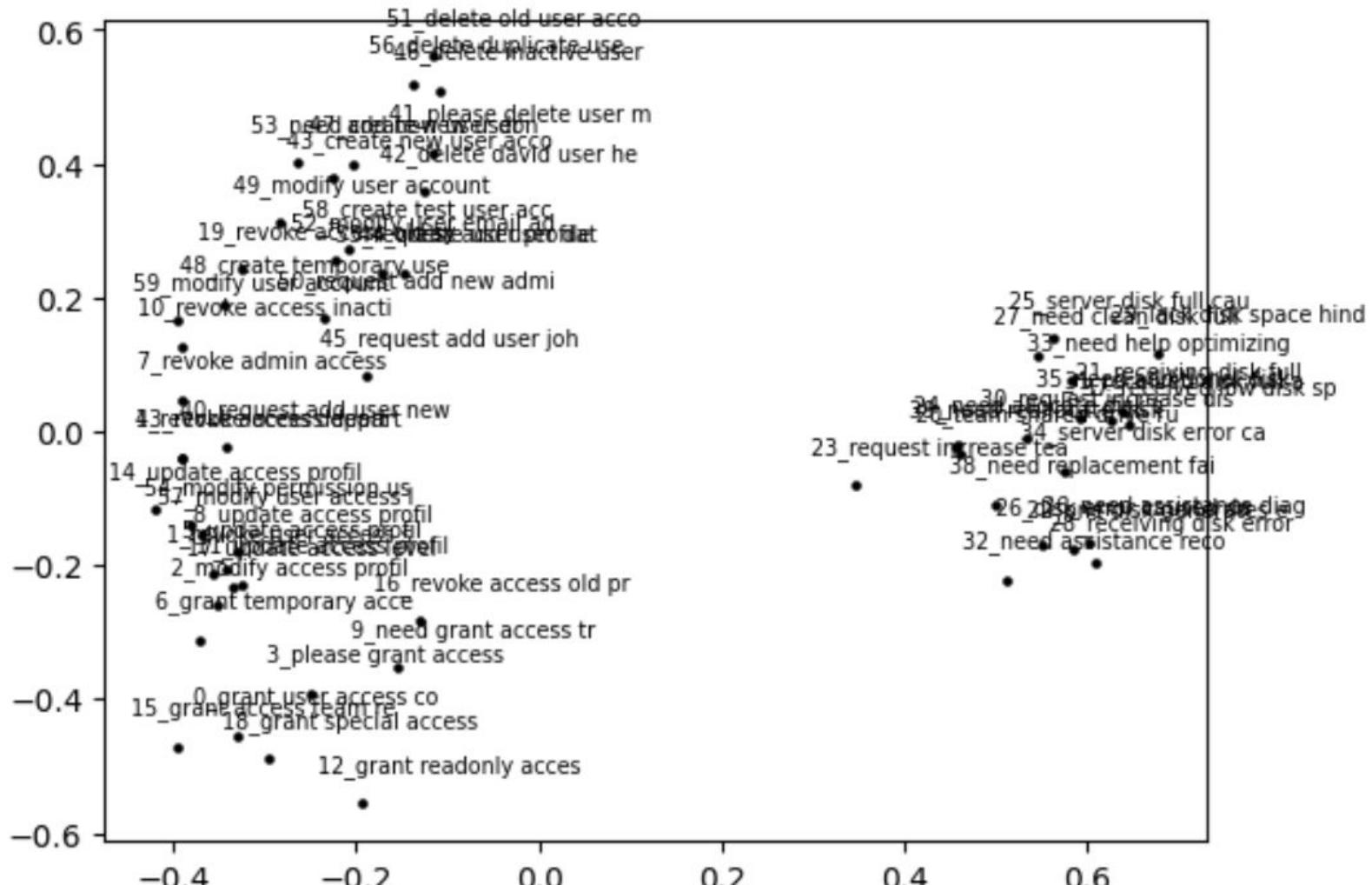
3.2 Do dim reduction for embeddings

```
import numpy as np
from sklearn.decomposition import PCA

pca = PCA(n_components=2, svd_solver='full')
x_embeds = embeds.copy()
X_transformed = pca.fit_transform(embeds)
```

```
X_transformed.shape
```

```
(60, 2)
```

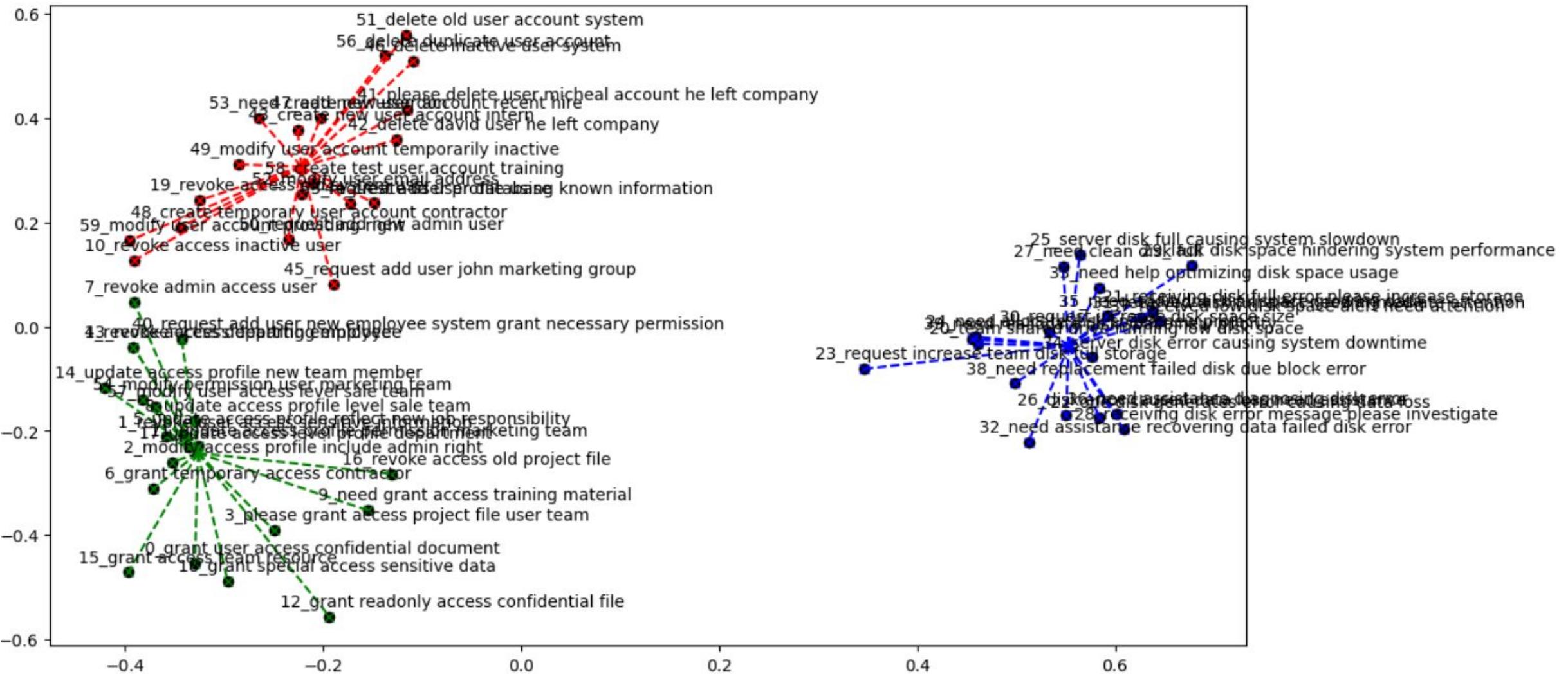


Support Ticket Classification

Github: https://github.com/enoten/support_ticket_analysis/blob/main/support_tickets_classification.ipynb

Youtube: <https://www.youtube.com/watch?v=NlJ1ySOF03M>

Clustering Embeddings



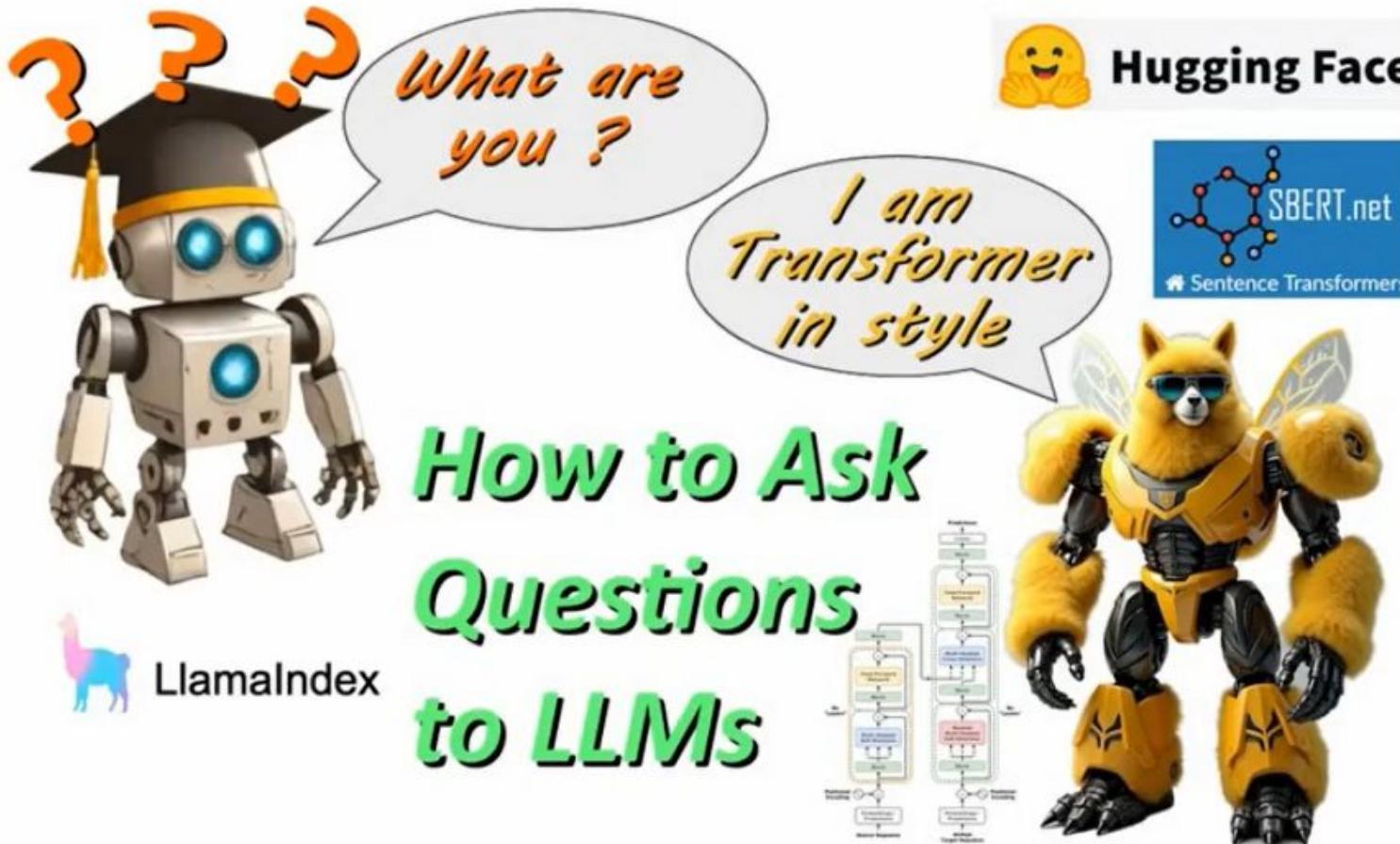
Support Ticket Classification

Github: https://github.com/enoten/support_ticket_analysis/blob/main/support_tickets_classification.ipynb

Youtube: <https://www.youtube.com/watch?v=NlJ1ySOF03M>

Semantic Search

Semantic Search



How to ask Questions to LLMs

Github:

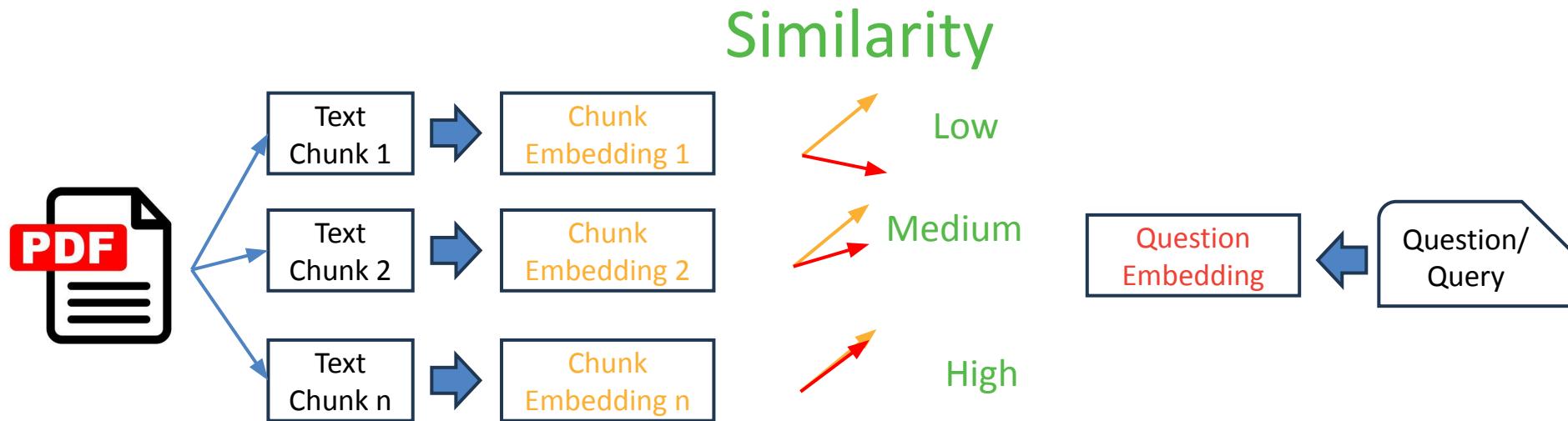
https://github.com/enoten/huggingface_llms_apps/blob/main/ask_questions_about_research_papers%20to%20LLMs.ipynb

Youtube: <https://www.youtube.com/watch?v=E4k1qGFemSE>

Origins of GenAI: Embeddings

Embeddings can be used to represent text/pdf documents in the numeric format.

On the other hand, we can convert Question/Query about text/pdf document into numeric vector as well.



Now We can Easily Find Most
Relevant Piece of the Text

Semantic Search

3.2.3 Applications of Attention in our Model The Transformer uses multi-head attention in three different ways:

- In "encoder-decoder attention" layers, the queries come from the previous decoder layer, and the memory keys and values come from the output of the encoder **Score:0.50** With a single attention head, averaging inhibits this **Score:0.46** In this work we employ $h = 8$ parallel attention layers, or **heads Score:0.46** Due to the reduced dimension of each head, the total computational cost is similar to that of single-head attention with full dimensionality **Score:0.43** Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions **Score:0.43** In a self-attention layer all of the keys, values and queries come from the same place, in this case, the output of the previous layer in the encoder **Score:0.37** • The encoder contains self-attention layers **Score:0.37** This mimics the typical encoder-decoder attention mechanisms in sequence-to-sequence models such as [38, 2, 9] **Score:0.36** • Similarly, self-attention layers in the decoder allow each position in the decoder to attend to all positions in the decoder up to and including that position **Score:0.33** 3.3 Position-wise Feed-Forward Networks In addition to attention sub-layers, each of the layers in our encoder and decoder contains a fully connected feed-forward network, which is applied to each position separately and identically **Score:0.33** We implement this inside of scaled dot-product attention by masking out (setting to $-\infty$) all values in the input of the softmax which correspond to illegal connections **Score:0.26** $\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W_O$ where $\text{head}_i = \text{Attention}(QW_Q^T, KW_K^T, WV_V^T)$ Where the projections are parameter matrices $WQ \in R^{d\text{model} \times dk}$, $WK \in R^{d\text{model} \times dk}$, $WV \in R^{d\text{model} \times dv}$ and $WO \in R^{d\text{model} \times d\text{model}}$ **Score:0.24** This allows every position in the decoder to attend over all positions in the input sequence **Score:0.20** Each position in the encoder can attend to all positions in the previous layer of the encoder **Score:0.15** This consists of two linear transformations with a ReLU activation in between **Score:0.15** 3.4 Embeddings and Softmax Similarly to other sequence transduction models, we use learned embeddings to convert the input tokens and output tokens to vectors of dimension d_{model} **Score:0.14** See Figure 2 **Score:0.14** We also use the usual learned linear transformation and softmax function to convert the decoder output to predicted next-token probabilities **Score:0.13** **5** **Score:0.13** Another way of describing this is as two convolutions with kernel size 1 **Score:0.12** The dimensionality of input and output is $d_{\text{model}} = 512$, and the inner-layer has dimensionality $d_{\text{ff}} = 2048$ **Score:0.10** output values **Score:0.10** We need to prevent leftward information flow in the decoder to preserve the auto-regressive property **Score:0.08** For each of these we use $dk = dv = d_{\text{model}}/h = 64$ **Score:0.04** These are concatenated and once again projected, resulting in the final values, as depicted in Figure 2 **Score:0.03** In the embedding layers, we multiply those weights by $\sqrt{d_{\text{model}}}$ **Score:0.02** In our model, we share the same weight matrix between the two embedding layers and the pre-softmax linear transformation, similar to [30] **Score:0.02** $\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$ (2) While the linear transformations are the same across different positions, they use different parameters from layer to layer **Score: -0.01**

How to ask Questions to LLMs

Github:

https://github.com/enoten/huggingface_llms_apps/blob/main/ask_questions_about_research_papers%20to%20LLMs.ipynb

Youtube: <https://www.youtube.com/watch?v=E4k1qGFemSE>

RDB Search

Relational DB Search

id	username	city	balance
3543	marichka	london	873
5322	theo	london	213
5565	alex	atlanta	643
6432	julian	new york	645
6788	nigel	santiago	4350
7545	syd	shanghai	5145
7657	kee	portland	75444
8645	jasper	boston	342
9769	luke	new york	2091
9908	miriam	new york	12

SQL indexing best practices | How to make your database FASTER!
<https://www.youtube.com/watch?v=BIIFTFrEFOI>

Relational DB Search

id	username	city	balance
3543	marichka	london	873
5322	theo	london	213
5565	alex	atlanta	643
6432	julian	new york	645
6788	nigel	santiago	4350
7545	syd	shanghai	5145
7657	kee	portland	75444
8645	jasper	boston	342
9769	luke	new york	2091
9908	miriam	new york	12

SQL indexing best practices | How to make your database FASTER!
<https://www.youtube.com/watch?v=BIIFTFrEFOI>

Relational DB Search

id	username	city	balance
3543	marichka	london	873
5322	theo	london	213
5565	alex	atlanta	643
6432	julian	new york	645
6788	nigel	santiago	4350
7545	syd	shanghai	5145
7657	kee	portland	75444
8645	jasper	boston	342
9769	luke	new york	2091
9908	miriam	new york	12

SQL indexing best practices | How to make your database FASTER!
<https://www.youtube.com/watch?v=BIIFTFrEFOI>

Relational DB Search

id	username	city	balance
3543	marichka	london	873
5322	theo	london	213
5565	alex	atlanta	643
6432	julian	new york	645
6788	nigel	santiago	4350
7545	syd	shanghai	5145
7657	kee	portland	75444
8645	jasper	boston	342
9769	luke	new york	2091
9908	miriam	new york	12

SQL indexing best practices | How to make your database FASTER!

<https://www.youtube.com/watch?v=BIIFTFrEFOI>

Relational DB Search

id	username	city	balance
3543	marichka	london	873
5322	theo	london	213
5565	alex	atlanta	643
6432	julian	new york	645
6788	nigel	santiago	4350
7545	syd	shanghai	5145
7657	kee	portland	75444
8645	jasper	boston	342
9769	luke	new york	2091
9908	miriam	new york	12

SQL indexing best practices | How to make your database FASTER!
<https://www.youtube.com/watch?v=BIIFTFrEFOI>

Relational DB Search

```
SELECT * FROM users  
WHERE city = 'new york'  
ORDER BY balance DESC;
```

Relational DB Search

id	username	city	balance
3543	marichka	london	873
5322	theo	london	213
5565	alex	atlanta	643
6432	julian	new york	645
6788	nigel	santiago	4350
7545	syd	shanghai	5145
7657	kee	portland	75444
8645	jasper	boston	342
9769	luke	new york	2091
9908	miriam	new york	12

SQL indexing best practices | How to make your database FASTER!
<https://www.youtube.com/watch?v=BIIFTFrEFOI>

Relational DB Search

id	username	city	balance
3543	marichka	london	873
5322	theo	london	213
5565	alex	atlanta	643
6432	julian	new york	645
6788	nigel	santiago	4350
7545	syd	shanghai	5145
7657	kee	portland	75444
8645	jasper	boston	342
9769	luke	new york	2091
9908	miriam	new york	12

SQL indexing best practices | How to make your database FASTER!
<https://www.youtube.com/watch?v=BIIFTFrEFOI>

Relational DB Search

id	username	city	balance
3543	marichka	london	873
5322	theo	london	213
5565	alex	atlanta	643
6432	julian	new york	645
6788	nigel	santiago	4350
7545	syd	shanghai	5145
7657	kee	portland	75444
8645	jasper	boston	342
9769	luke	new york	2091
9908	miriam	new york	12

SQL indexing best practices | How to make your database FASTER!
<https://www.youtube.com/watch?v=BIIFTFrEFOI>

Relational DB Search

id	username	city	balance
3543	marichka	london	873
5322	theo	london	213
5565	alex	atlanta	643
6432	julian	new york	645
6788	nigel	santiago	4350
7545	syd	shanghai	5145
7657	kee	portland	75444
8645	jasper	boston	342
9769	luke	new york	2091
9908	miriam	new york	12

SQL indexing best practices | How to make your database FASTER!
<https://www.youtube.com/watch?v=BIIFTFrEFOI>

Relational DB Search

id	username	city	balance
3543	marichka	london	873
5322	theo	london	213
5565	alex	atlanta	643
6432	julian	new york	645
6788	nigel	santiago	4350
7545	syd	shanghai	5145
7657	kee	portland	75444
8645	jasper	boston	342
9769	luke	new york	2091
9908	miriam	new york	12

SQL indexing best practices | How to make your database FASTER!
<https://www.youtube.com/watch?v=BIIFTFrEFOI>

Relational DB Search

id	username	city	balance
3543	marichka	london	873
5322	theo	london	213
5565	alex	atlanta	643
6432	julian	new york	645
6788	nigel	santiago	4350
7545	syd	shanghai	5145
7657	kee	portland	75444
8645	jasper	boston	342
9769	luke	new york	2091
9908	miriam	new york	12

SQL indexing best practices | How to make your database FASTER!
<https://www.youtube.com/watch?v=BIIFTFrEFOI>

id	username	city	balance
3543	marichka	london	873
5322	theo	london	213
5565	alex	atlanta	643
6432	julian	new york	645
6788	nigel	santiago	4350
7545	syd	shanghai	5145
7657	kee	portland	75444
8645	jasper	boston	342
9769	luke	new york	2091
9908	miriam	new york	12

SQL indexing best practices | How to make your database FASTER!
<https://www.youtube.com/watch?v=BIIFTFrEFOI>

id	username	city	balance
3543	marichka	london	873
5322	theo	london	213
5565	alex	atlanta	643
6432	julian	new york	645
6788	nigel	santiago	4350
7545	syd	shanghai	5145
7657	kee	portland	75444
8645	jasper	boston	342
9769	luke	new york	2091
9908	miriam	new york	12

SQL indexing best practices | How to make your database FASTER!
<https://www.youtube.com/watch?v=BIIFTFrEFOI>

id	username	city	balance
3543	marichka	london	873
5322	theo	london	213
5565	alex	atlanta	643
6432	julian	new york	645
6788	nigel	santiago	4350
7545	syd	shanghai	5145
7657	kee	portland	75444
8645	jasper	boston	342
9769	luke	new york	2091
9908	miriam	new york	12

SQL indexing best practices | How to make your database FASTER!
<https://www.youtube.com/watch?v=BIIFTFrEFOI>

Relational DB Search

id	username	city	balance
3543	marichka	london	873
5322	theo	london	213
5565	alex	atlanta	643
6432	julian	new york	645
6788	nigel	santiago	4350
7545	syd	shanghai	5145
7657	kee	portland	75444
8645	jasper	boston	342
9769	luke	new york	2091
9908	miriam	new york	12

SQL indexing best practices | How to make your database FASTER!
<https://www.youtube.com/watch?v=BIIFTFrEFOI>

Relational DB Search

8645	jasper	boston	342
9769	luke	new york	2091
9908	miriam	new york	12
3543	marichka	london	873
5322	theo	london	213
5565	alex	atlanta	643
6432	julian	new york	645
6788	nigel	santiago	4350
7545	syd	shanghai	5145
7657	kee	portland	75444
8645	jasper	boston	342
9769	luke	new york	2091
9908	miriam	new york	12
3543	marichka	london	873
5322	theo	london	213
5565	alex	atlanta	643
6432	julian	new york	645
6788	nigel	santiago	4350
7545	syd	shanghai	5145
7657	kee	portland	75444
8645	jasper	boston	342
9769	luke	new york	2091
9908	miriam	new york	12
3543	marichka	london	873
5322	theo	london	213
5565	alex	atlanta	643
6432	julian	new york	645
6788	nigel	santiago	4350
7545	syd	shanghai	5145
7657	kee	portland	75444
8645	jasper	boston	342
9769	luke	new york	2091
9908	miriam	new york	12
3543	marichka	london	873
5322	theo	london	213
5565	alex	atlanta	643
6432	julian	new york	645
6788	nigel	santiago	4350
7545	syd	shanghai	5145
7657	kee	portland	75444
8645	jasper	boston	342
9769	luke	new york	2091
9908	miriam	new york	12
3543	marichka	london	873
5322	theo	london	213

SQL indexing best practices | How to make your database FASTER!

<https://www.youtube.com/watch?v=BIIFTFrEFOI>

Relational DB Search

```
planning time: 430µs
execution time: 2ms

info

EXPLAIN ANALYZE
SELECT * FROM users
WHERE city = 'new york'
ORDER BY balance DESC;

3
  • sort
    nodes: n2
    actual row count: 3
    estimated max memory allocated: 10 KiB
    estimated max sql temp disk usage: 0 B
    estimated row count: 3
    order: -balance

2
  • filter
    nodes: n2
    actual row count: 3
    estimated row count: 3
    filter: city = 'new york'

1
  • scan
    nodes: n2
    actual row count: 10
    KV time: 2ms
    KV contention time: 0µs
    KV rows read: 10
    KV bytes read: 532 B
    estimated max memory allocated: 20 KiB
    estimated row count: 10 (100% of the table; stats collected 57 seconds ago)
    table: users@users_pkey
    spans: FULL SCAN
```

SQL indexing best practices | How to make your database FASTER!
<https://www.youtube.com/watch?v=BIIFTFrEFOI>

Relational DB Search

```
CREATE INDEX city_idx ON users (city);
```

SQL indexing best practices | How to make your database FASTER!
<https://www.youtube.com/watch?v=BIIFTFrEFOI>

Relational DB Search

atlanta	_003
boston	_008
london	_001
london	_002
new york	_004
new york	_009
new york	_010
portland	_007
santiago	_005
shanghai	_006

SQL indexing best practices | How to make your database FASTER!
<https://www.youtube.com/watch?v=BIIFTFrEFOI>

Relational DB Search

atlanta	_003
boston	_008
london	_001
london	_002
new york	_004
new york	_009
new york	_010
portland	_007
santiago	_005
shanghai	_006

SQL indexing best practices | How to make your database FASTER!
<https://www.youtube.com/watch?v=BIIFTFrEFOI>

			id	username	city	balance
atlanta	_003		3543	marichka	london	873
boston	_008		5322	theo	london	213
london	_001		5565	alex	atlanta	643
london	_002		6432	julian	new york	645
new york	_004		6788	nigel	santiago	4350
new york	_009		7545	syd	shanghai	5145
new york	_010		7657	kee	portland	75444
portland	_007		8645	jasper	boston	342
santiago	_005		9769	luke	new york	2091
shanghai	_006		9908	miriam	new york	12

SQL indexing best practices | How to make your database FASTER!
<https://www.youtube.com/watch?v=BIIFTFrEFOI>

```
CREATE INDEX city_balance_idx  
ON users (city, balance);
```

Relational DB Search

```
EXPLAIN ANALYZE
SELECT * FROM users
WHERE city = 'new york'
ORDER BY balance DESC;
```

2

1

info

```
planning time: 901µs
execution time: 4ms
```

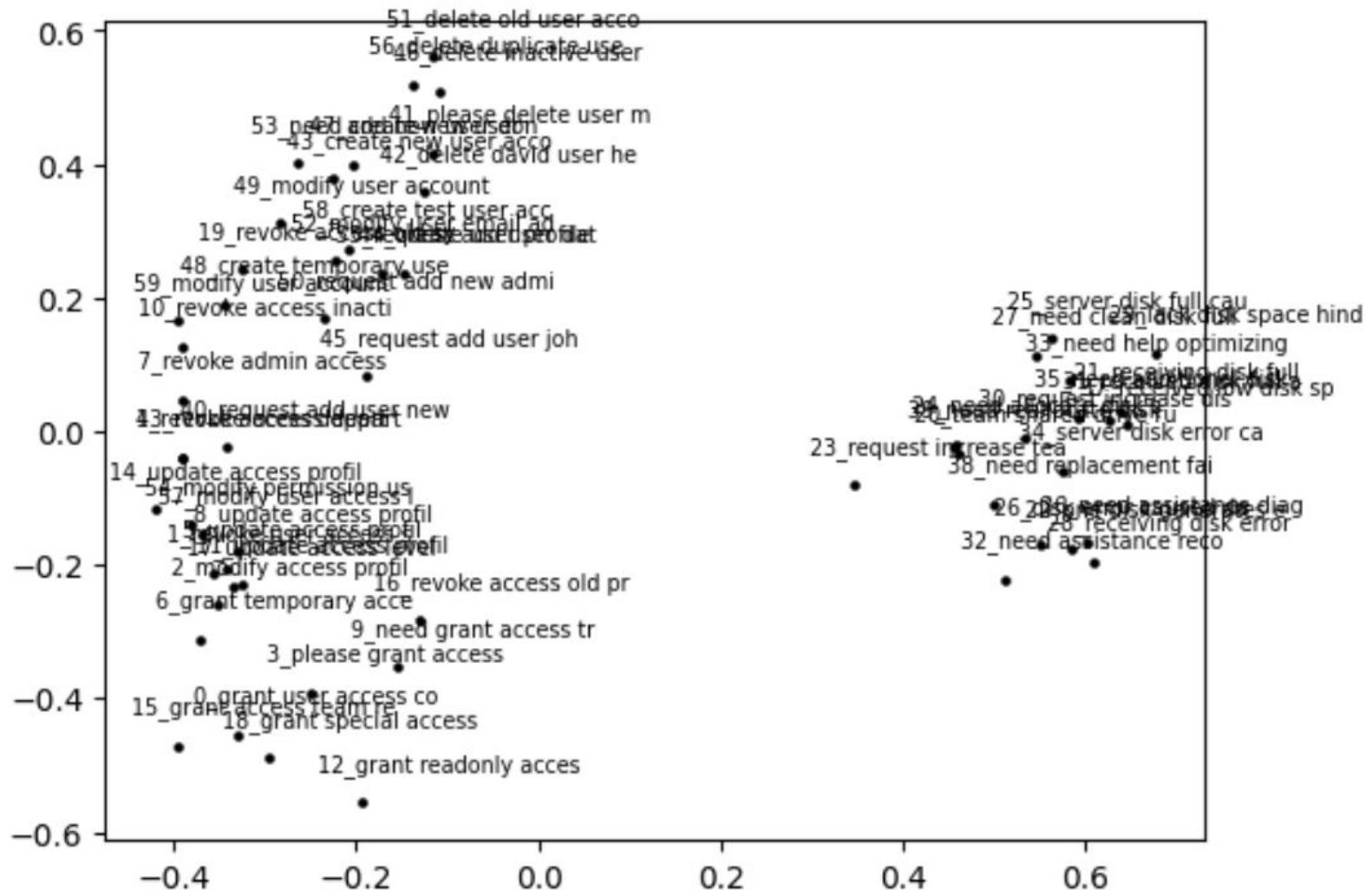
```
• index join
  nodes: n2
  actual row count: 3
  KV time: 1ms
  KV contention time: 0µs
  KV rows read: 3
  KV bytes read: 108 B
  estimated max memory allocated: 20 KiB
  estimated max sql temp disk usage: 0 B
  estimated row count: 2
  table: users@users_pkey

  • revscan
    nodes: n2
    actual row count: 3
    KV time: 1ms
    KV contention time: 0µs
    KV rows read: 3
    KV bytes read: 142 B
    estimated max memory allocated: 20 KiB
    estimated row count: 2 (16% of the table; stats collected 2 minutes ago)
    table: users@city_balance_idx
    spans: [/new york' - /new york']
```

SQL indexing best practices | How to make your database FASTER!
<https://www.youtube.com/watch?v=BIIFTFrEFOI>

Vector Space Search

Vector Space Search



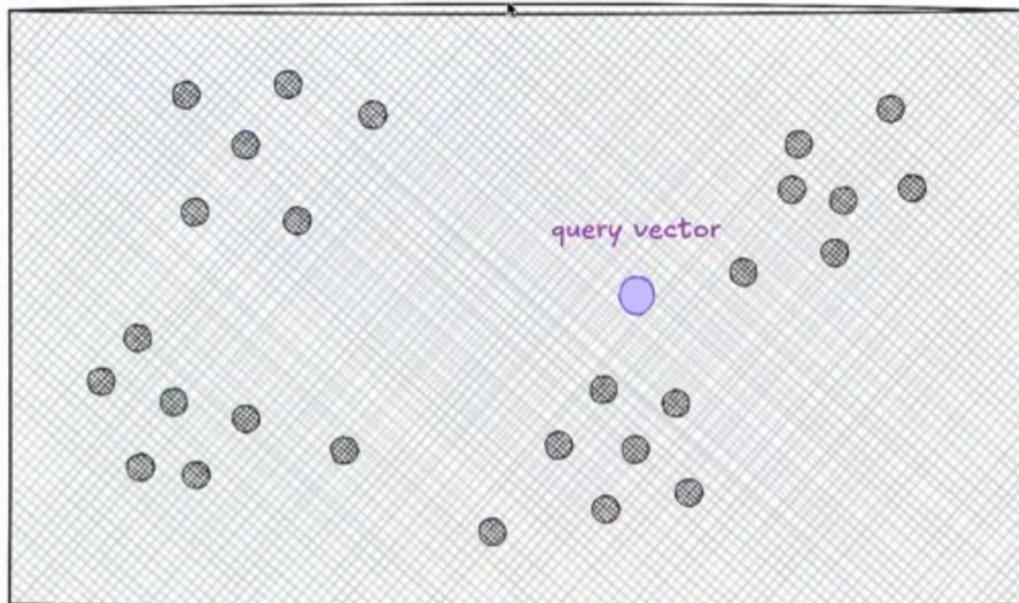
Support Ticket Classification

Github: https://github.com/enoten/support_ticket_analysis/blob/main/support_tickets_classification.ipynb

Youtube: <https://www.youtube.com/watch?v=NlJ1ySOF03M>

Vector Space Search: IVF indexing

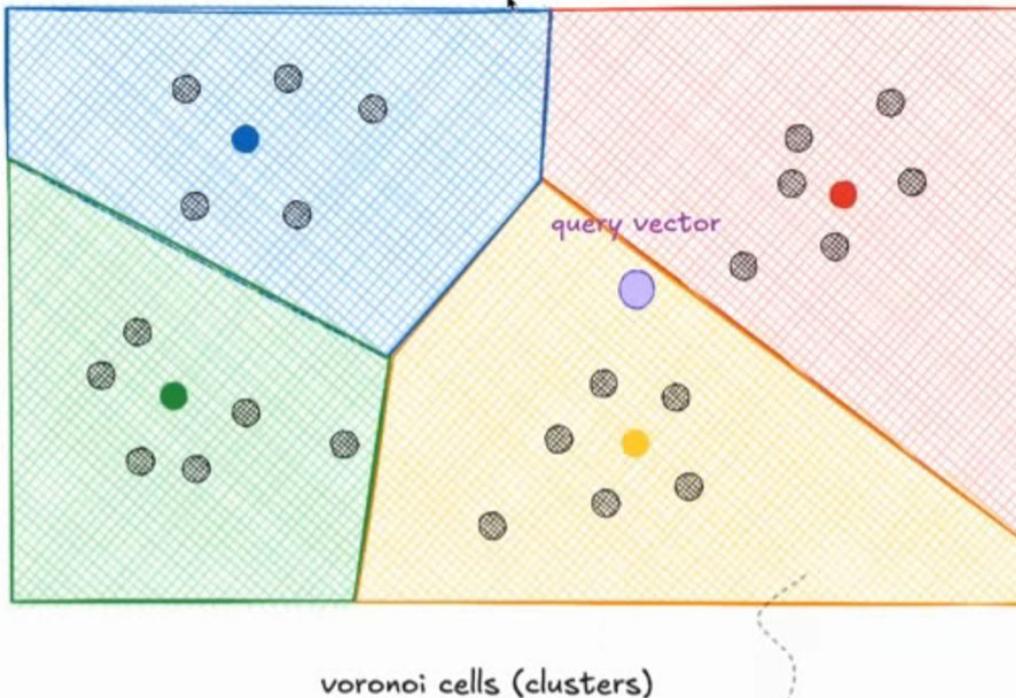
IVF (Inverted File Indexing)



Vector Search & Approximate Nearest Neighbors (ANN) | FAISS (HNSW & IVF)
<https://www.youtube.com/watch?v=chz74Mtd1AA>

Vector Space Search: IVF indexing

IVF (Inverted File Indexing)



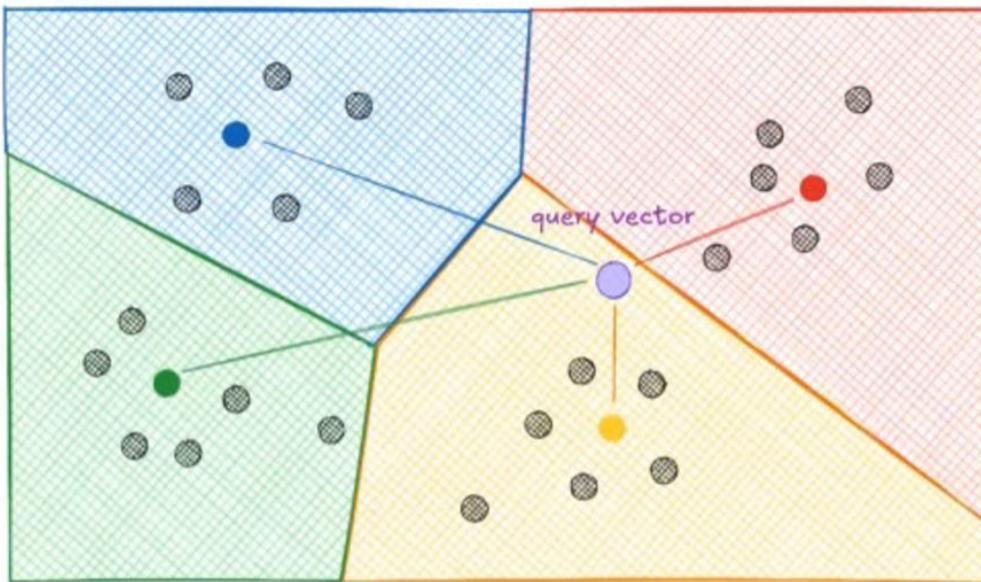
<https://sfoteini.github.io/blog/use-ivfflat-index-on-azure-cosmos-db-for-postgresql-for-similarity-search/>

Vector Search & Approximate Nearest Neighbors (ANN) | FAISS (HNSW & IVF)

<https://www.youtube.com/watch?v=chz74Mtd1AA>

Vector Space Search: IVF indexing

IVF (Inverted File Indexing)



Step 1

1. Identify the cluster by choosing the closest centroid

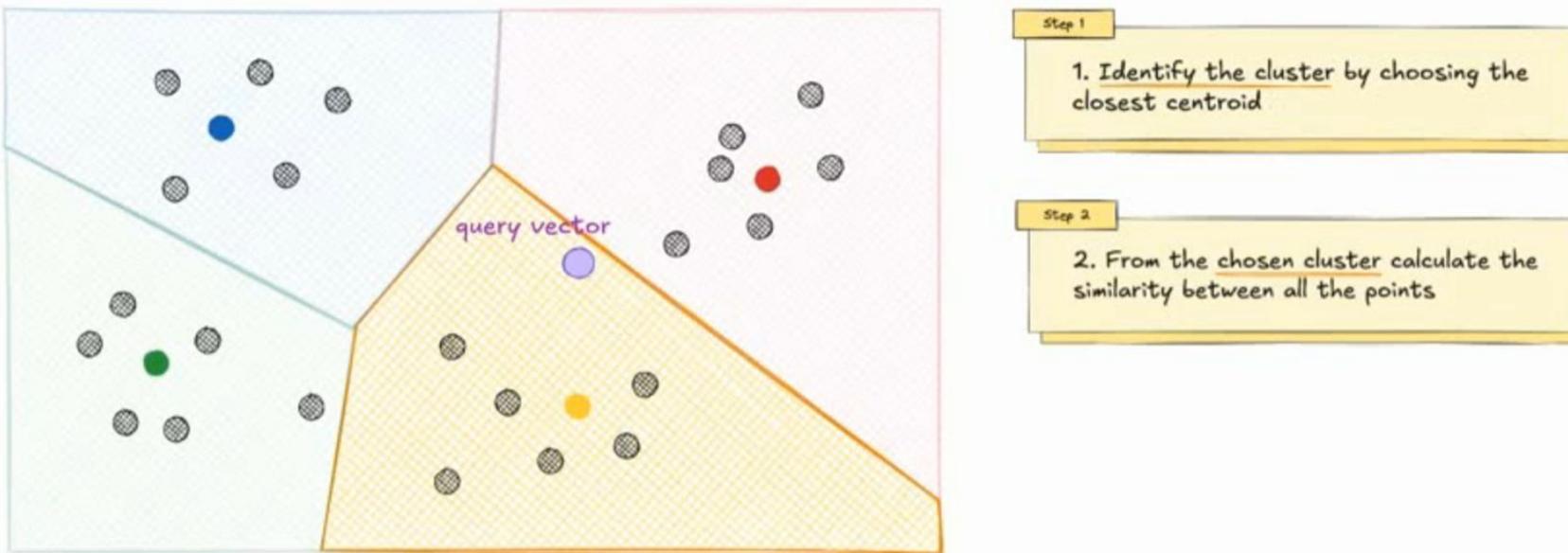
<https://sfoteini.github.io/blog/use-ivfflat-index-on-azure-cosmos-db-for-postgresql-for-similarity-search/>

Vector Search & Approximate Nearest Neighbors (ANN) | FAISS (HNSW & IVF)

<https://www.youtube.com/watch?v=chz74Mtd1AA>

Vector Space Search: IVF indexing

IVF (Inverted File Indexing)



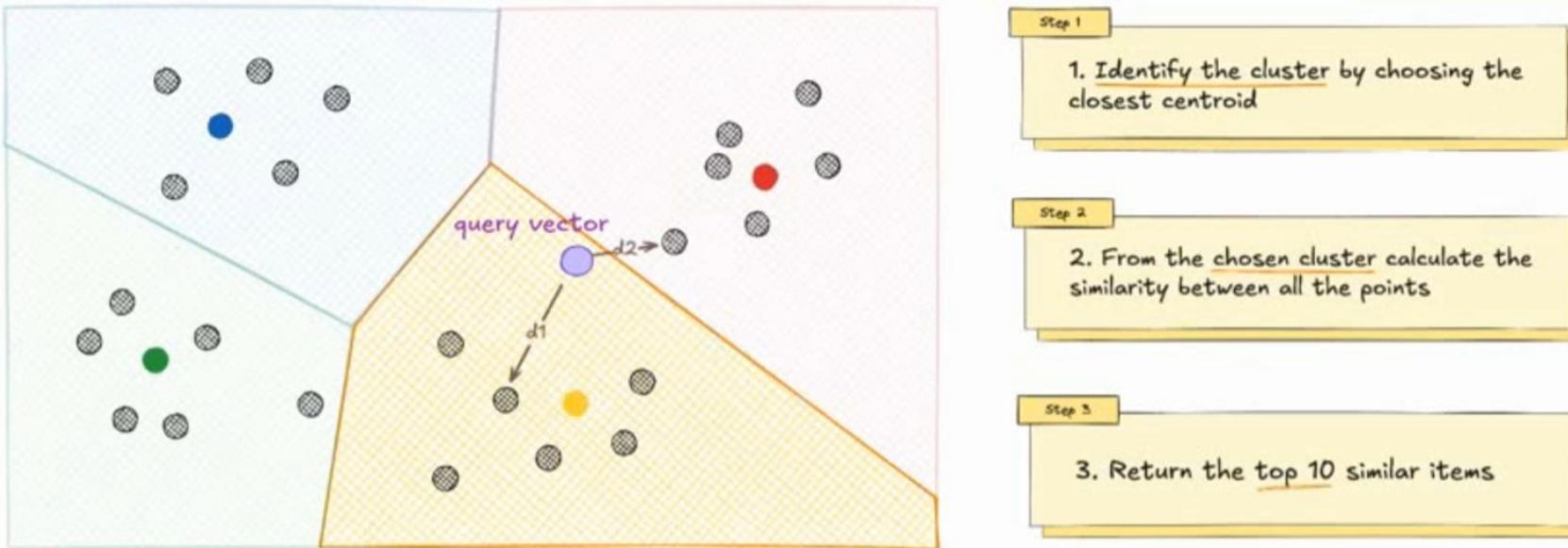
<https://sfoteini.github.io/blog/use-ivfflat-index-on-azure-cosmos-db-for-postgresql-for-similarity-search/>

Vector Search & Approximate Nearest Neighbors (ANN) | FAISS (HNSW & IVF)

<https://www.youtube.com/watch?v=chz74Mtd1AA>

Vector Space Search: IVF indexing

IVF (Inverted File Indexing)



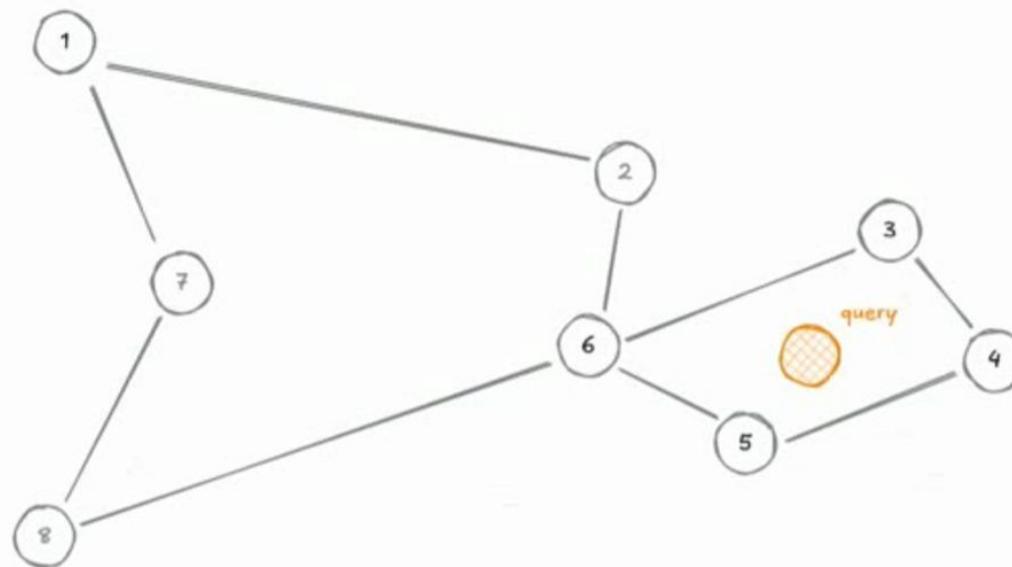
<https://sfoteini.github.io/blog/use-ivfflat-index-on-azure-cosmos-db-for-postgresql-for-similarity-search/>

Vector Search & Approximate Nearest Neighbors (ANN) | FAISS (HNSW & IVF)

<https://www.youtube.com/watch?v=chz74Mtd1AA>

Vector Space Search: Hierarchical Navigable Small Worlds

Navigable Small World (NSW)



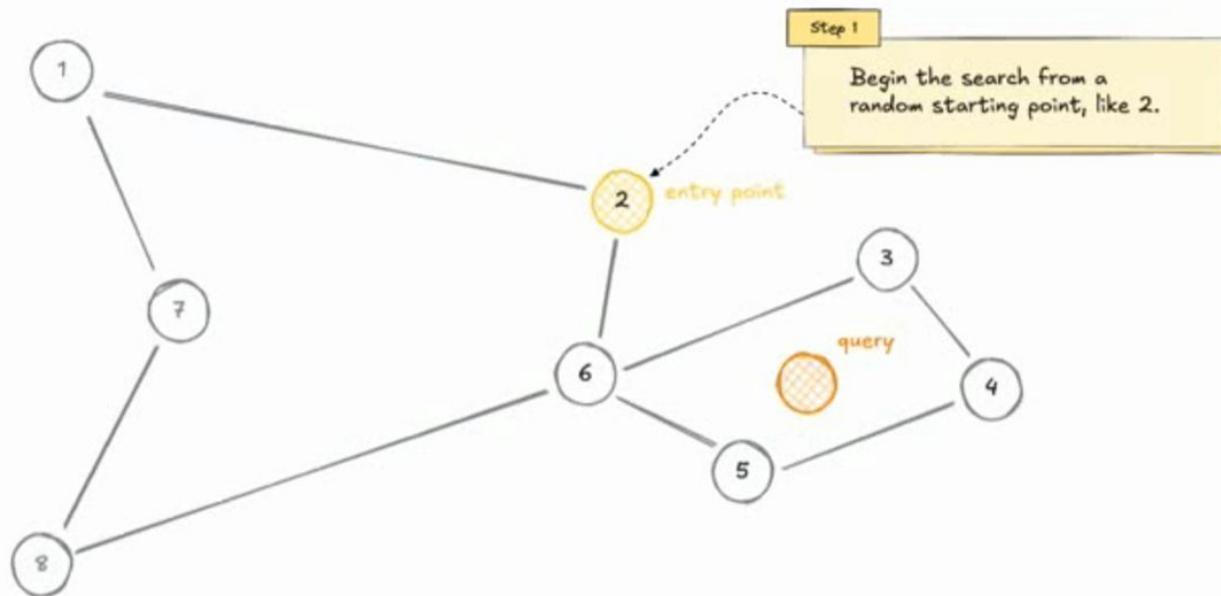
<https://towardsdatascience.com/similarity-search-part-4-hierarchical-navigable-small-world-hnsw-2aad4fe87d37>

Vector Search & Approximate Nearest Neighbors (ANN) | FAISS (HNSW & IVF)

<https://www.youtube.com/watch?v=chz74Mtd1AA>

Vector Space Search: Hierarchical Navigable Small Worlds

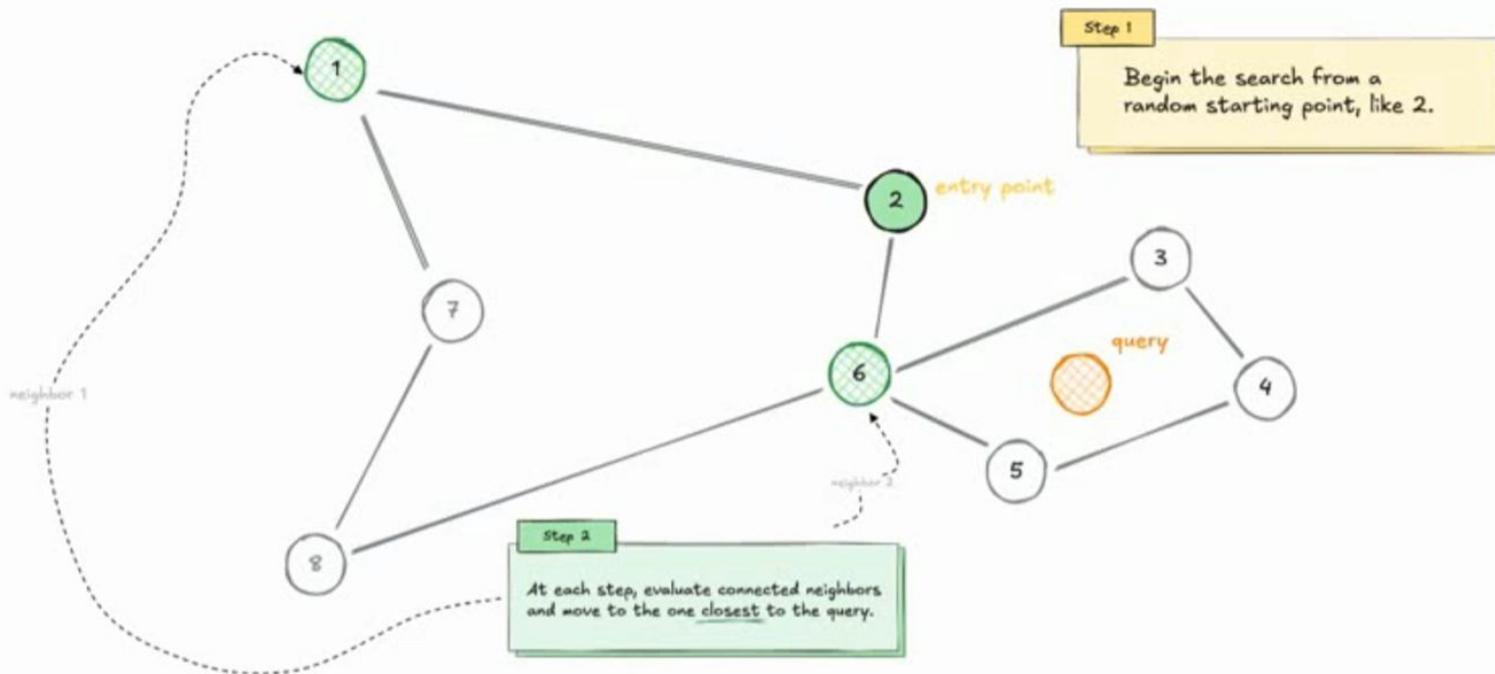
Navigable Small World (NSW)



Vector Search & Approximate Nearest Neighbors (ANN) | FAISS (HNSW & IVF)
<https://www.youtube.com/watch?v=chz74Mtd1AA>

Vector Space Search: Hierarchical Navigable Small Worlds

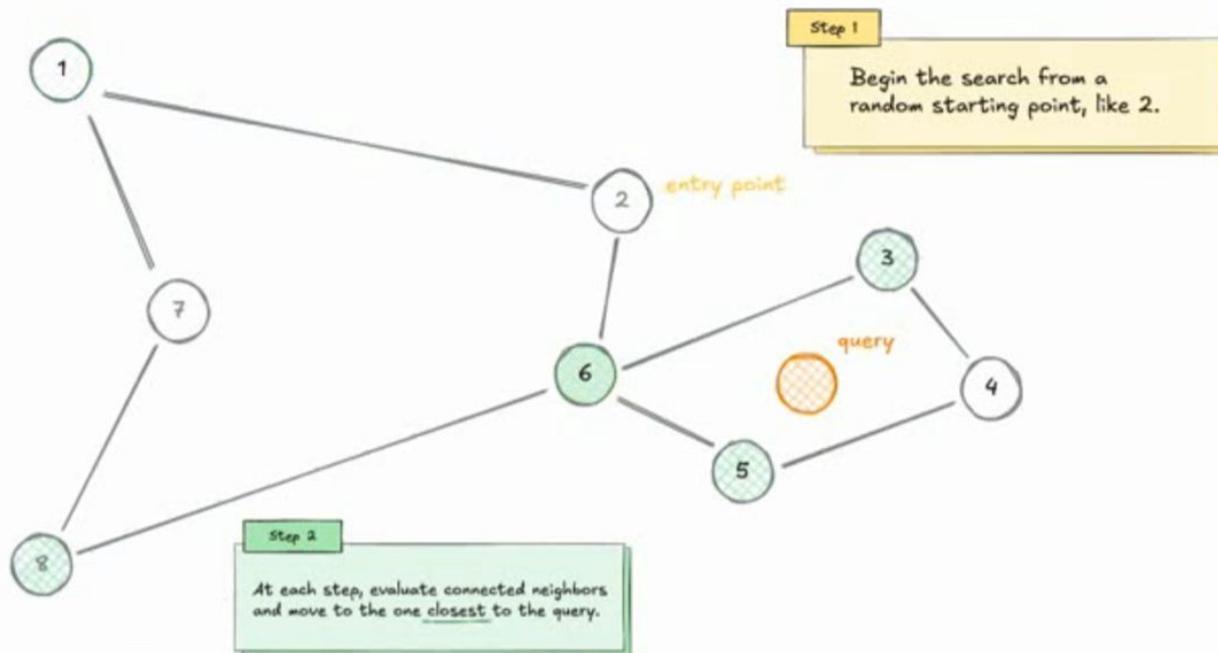
Navigable Small World (NSW)



Vector Search & Approximate Nearest Neighbors (ANN) | FAISS (HNSW & IVF)
<https://www.youtube.com/watch?v=chz74Mtd1AA>

Vector Space Search: Hierarchical Navigable Small Worlds

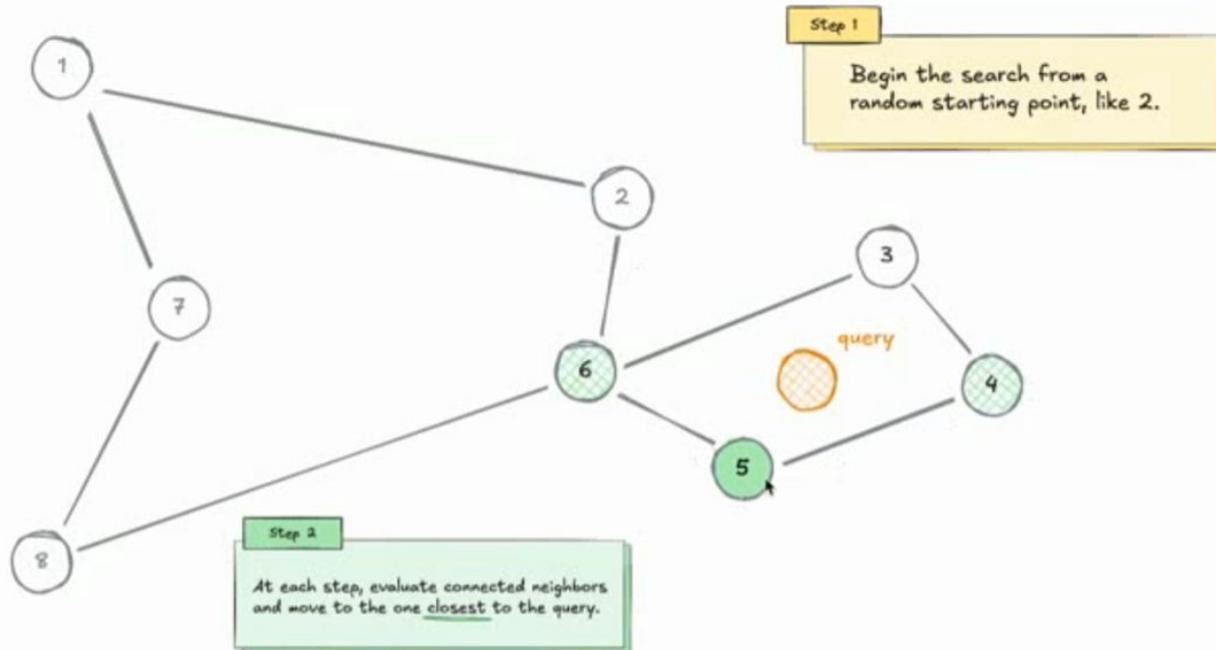
Navigable Small World (NSW)



Vector Search & Approximate Nearest Neighbors (ANN) | FAISS (HNSW & IVF)
<https://www.youtube.com/watch?v=chz74Mtd1AA>

Vector Space Search: Hierarchical Navigable Small Worlds

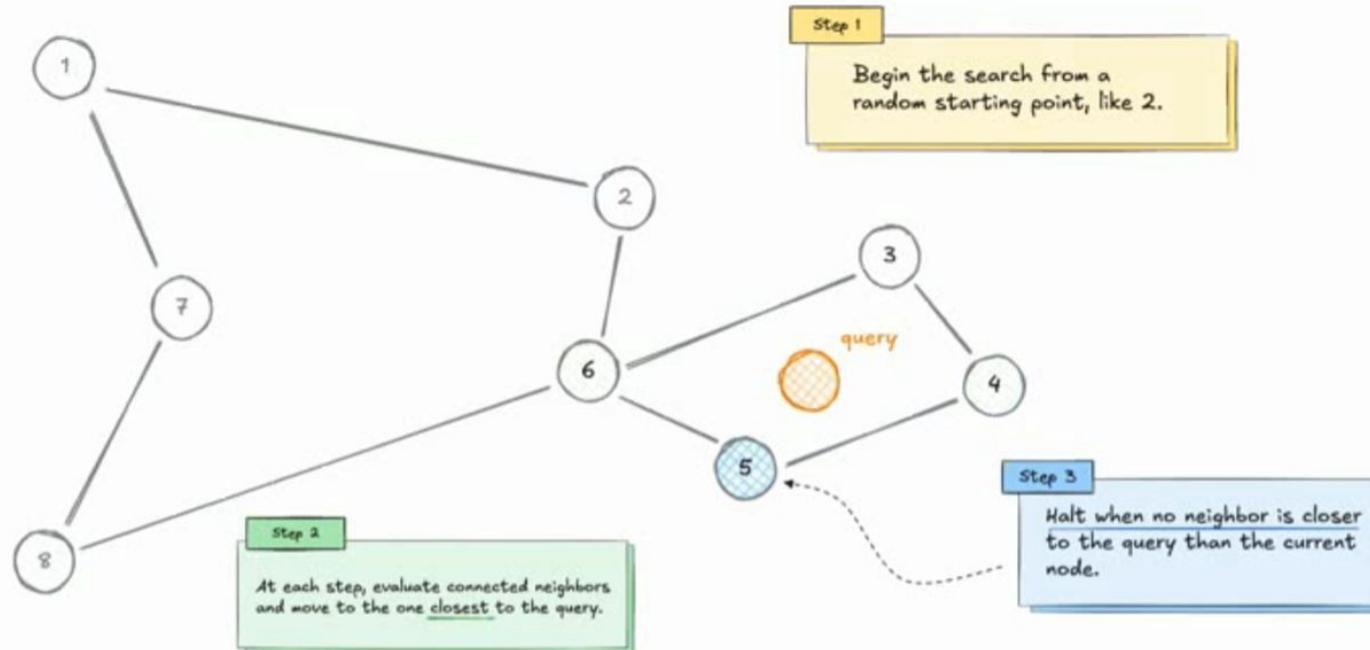
Navigable Small World (NSW)



Vector Search & Approximate Nearest Neighbors (ANN) | FAISS (HNSW & IVF)
<https://www.youtube.com/watch?v=chz74Mtd1AA>

Vector Space Search: Hierarchical Navigable Small Worlds

Navigable Small World (NSW)

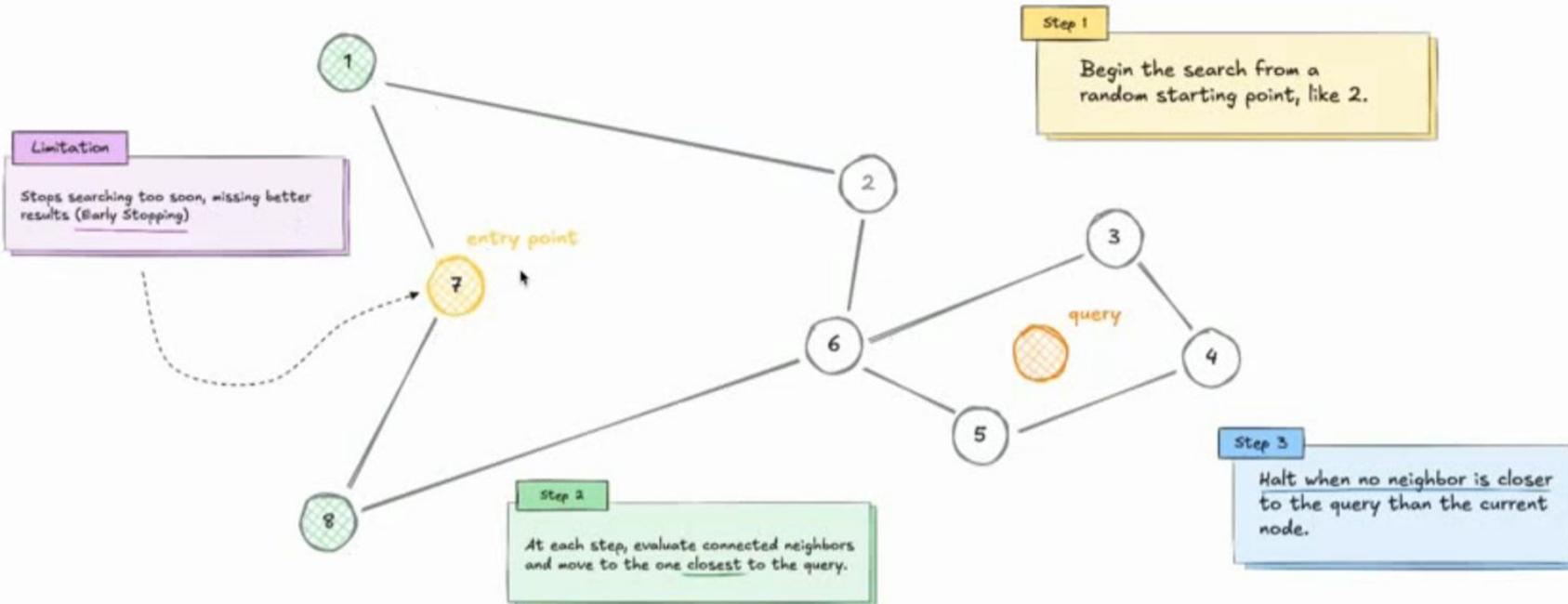


Vector Search & Approximate Nearest Neighbors (ANN) | FAISS (HNSW & IVF)

<https://www.youtube.com/watch?v=chz74Mtd1AA>

Vector Space Search: Hierarchical Navigable Small Worlds

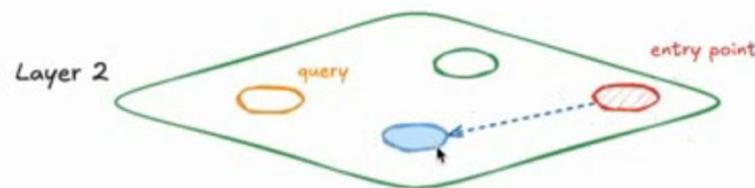
Navigable Small World (NSW)



Vector Search & Approximate Nearest Neighbors (ANN) | FAISS (HNSW & IVF)
<https://www.youtube.com/watch?v=chz74Mtd1AA>

Vector Space Search: Hierarchical Navigable Small Worlds

Hierarchical Navigable Small World (HNSW)

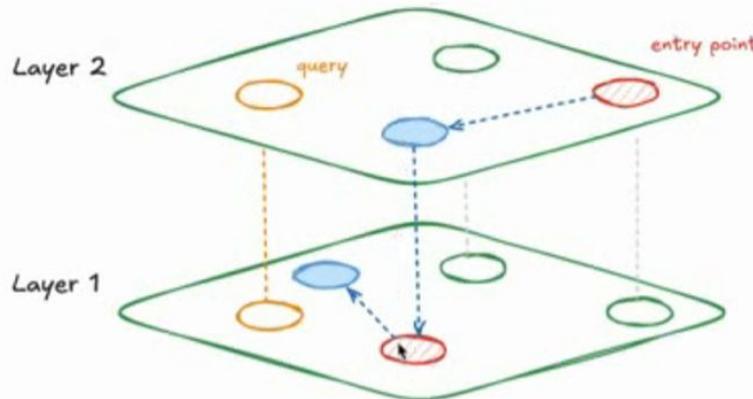


Step 1

The search begins at a random entry point in the topmost layer.

Vector Space Search: Hierarchical Navigable Small Worlds

Hierarchical Navigable Small World (HNSW)



Step 1

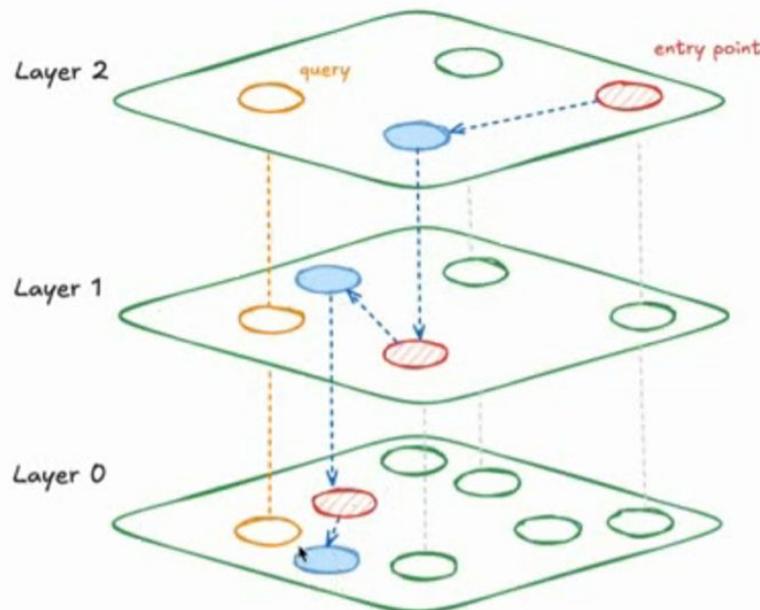
The search begins at a random entry point in the topmost layer.

Step 2

At each layer, the algorithm moves to the closest neighbor, narrowing down the search.

Vector Space Search: Hierarchical Navigable Small Worlds

Hierarchical Navigable Small World (HNSW)



Step 1

The search begins at a **random entry point** in the **topmost layer**.

Step 2

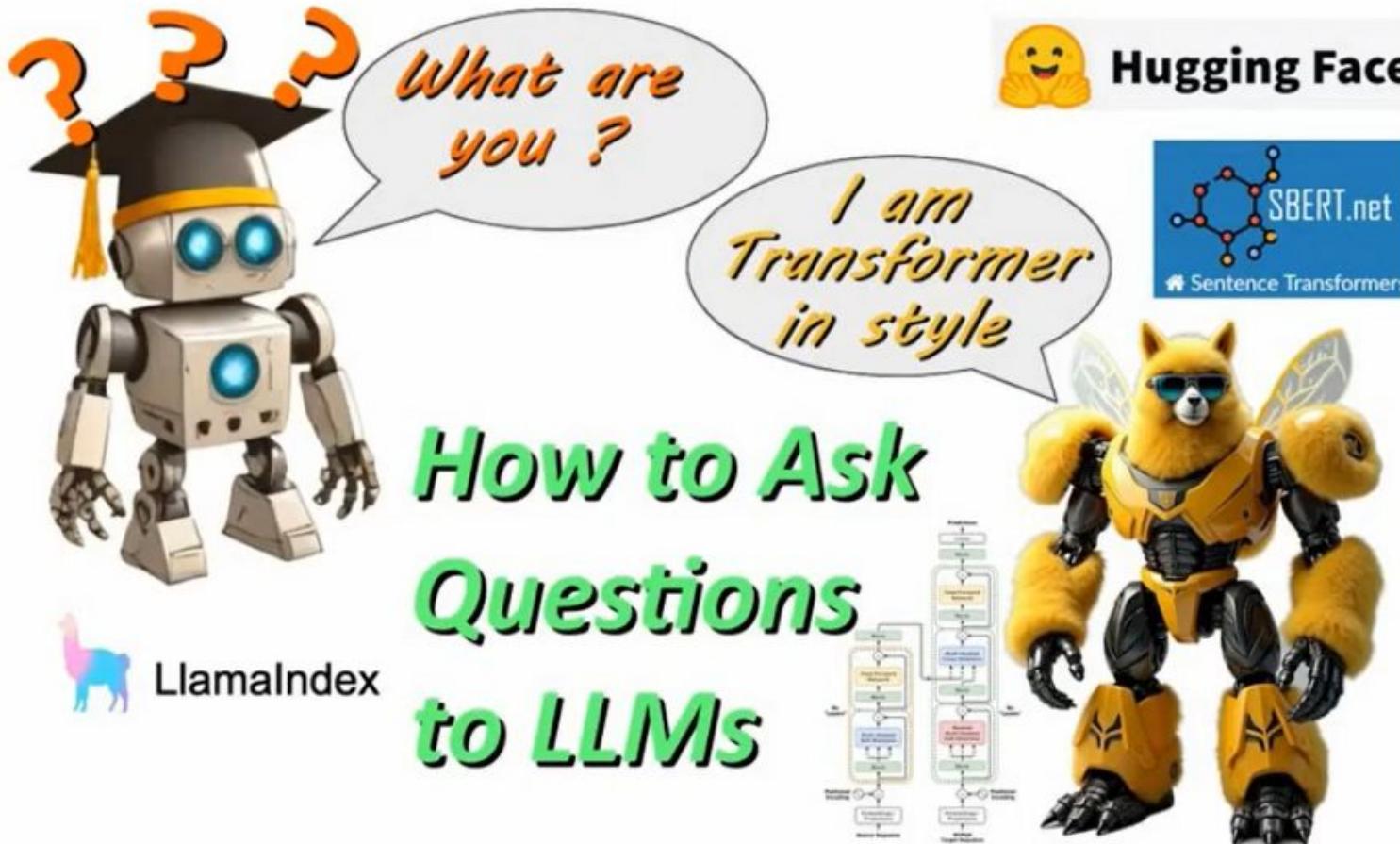
At each layer, the algorithm **moves to the closest neighbor**, narrowing down the search.

Step 3

In Layer 0, the **densest layer**, the nearest neighbor to the query is identified.

Document Chunking

Semantic Search



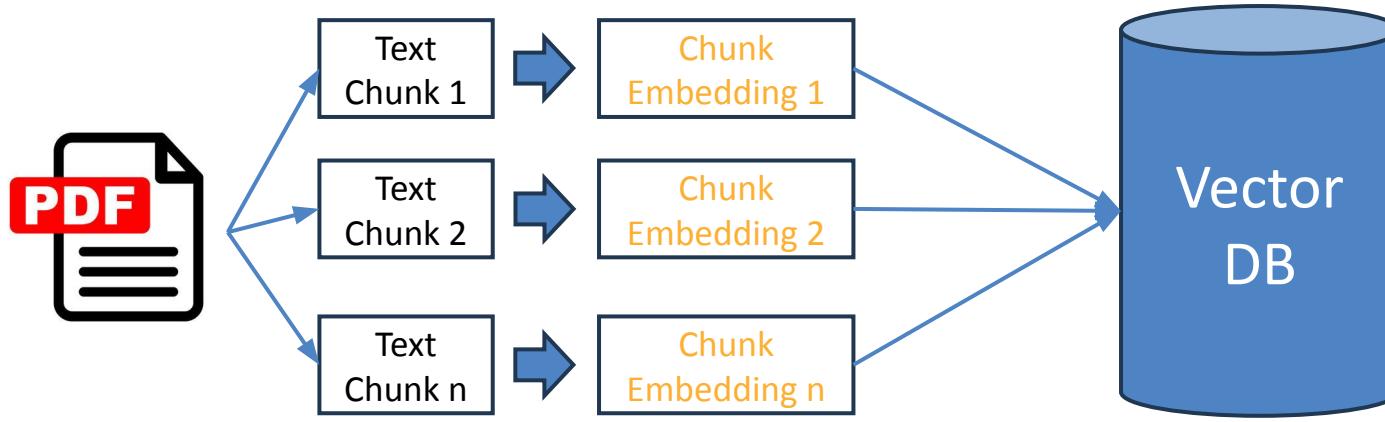
How to ask Questions to LLMs

Github:

https://github.com/enoten/huggingface_llms_apps/blob/main/ask_questions_about_research_papers%20to%20LLMs.ipynb

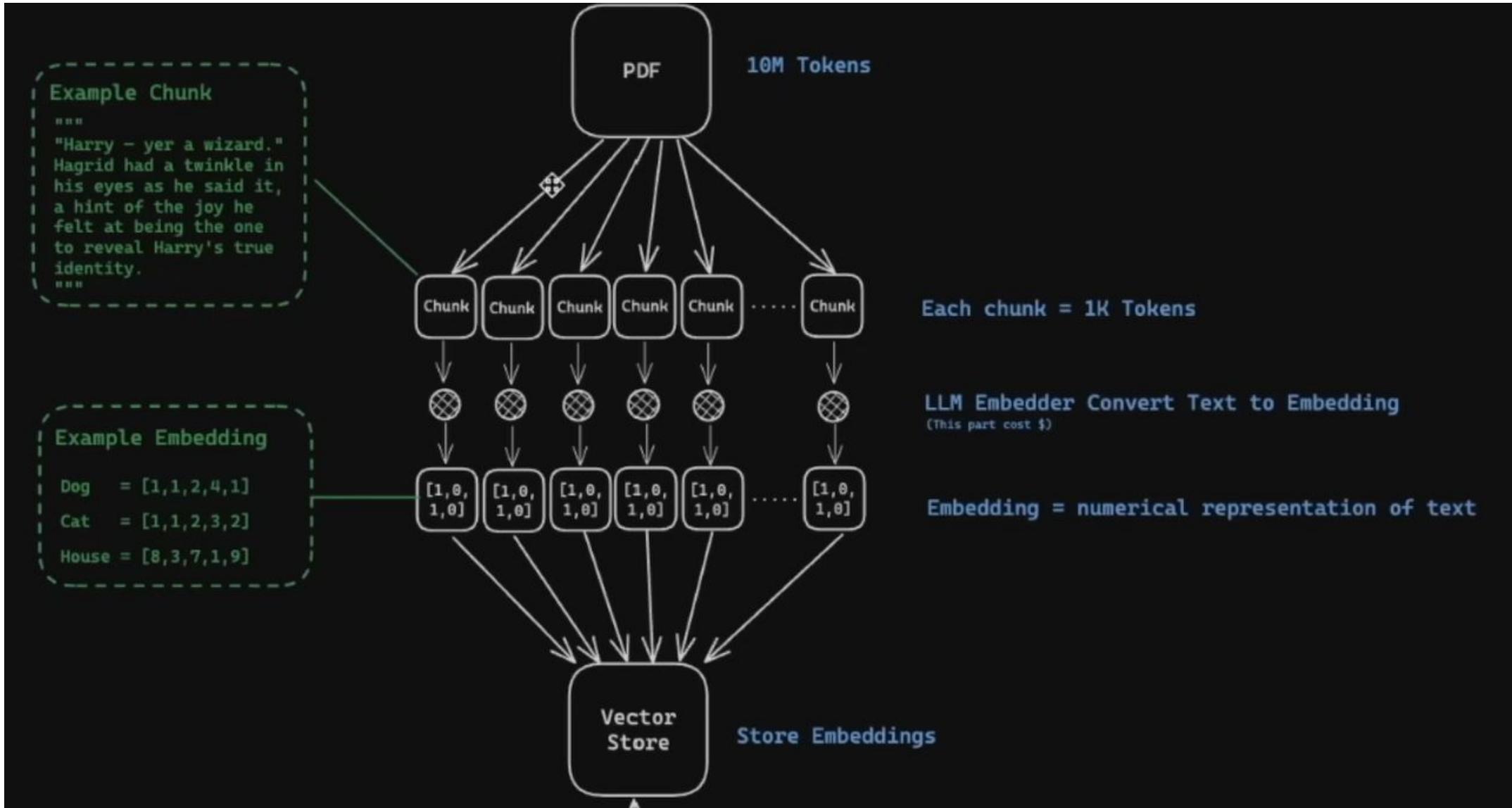
Youtube: <https://www.youtube.com/watch?v=E4k1qGFemSE>

Embeddings & Vector DB



Vector DB

Vector DB: Generic Stru

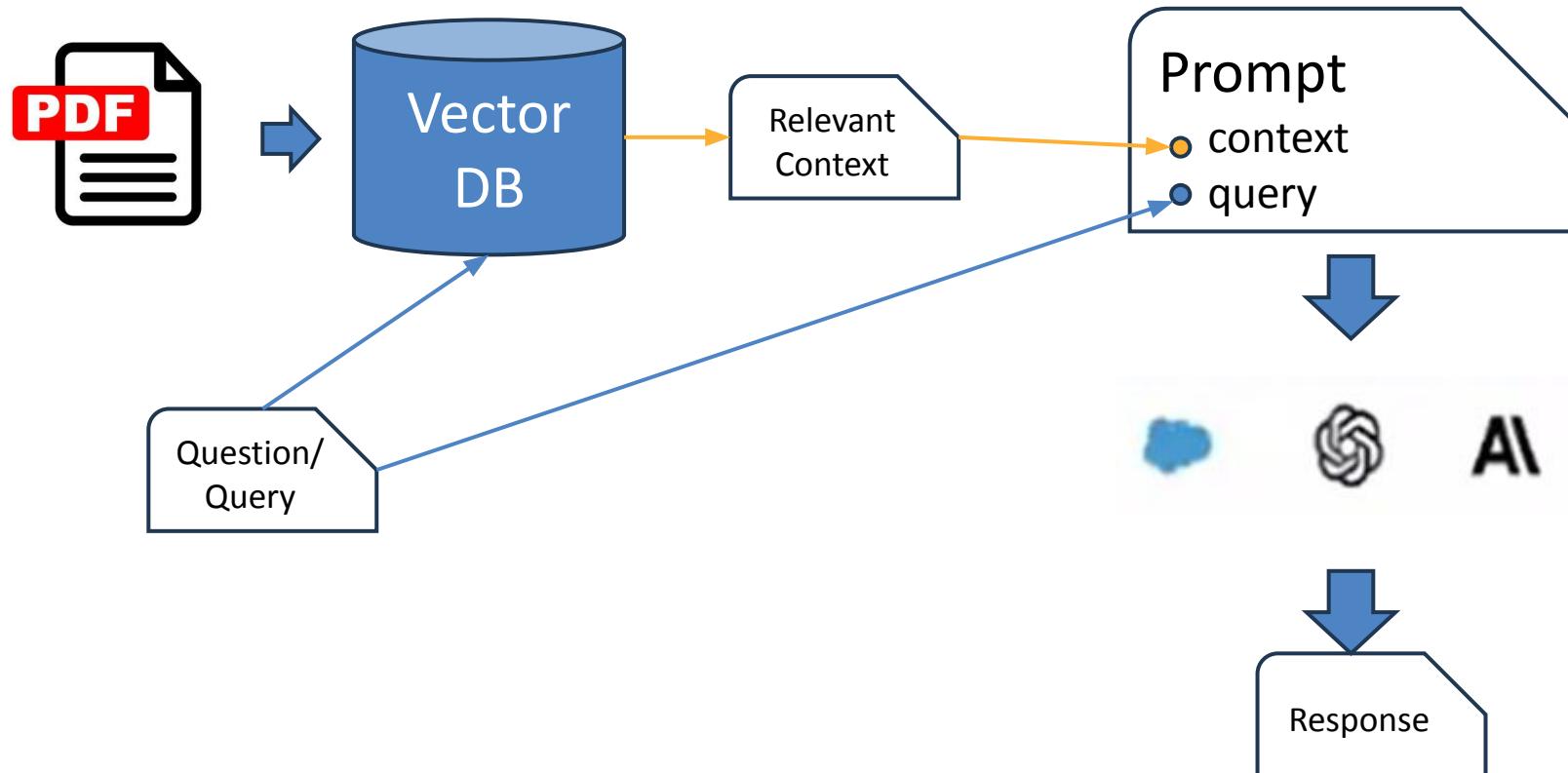


Vector DB: Examples

Database	Strengths	Use Cases	Type	Vendor
Pinecone	<ul style="list-style-type: none">- Fully managed- High scalability- Real-time updates- Advanced filtering	<ul style="list-style-type: none">- Production-grade RAG- Semantic & hybrid search- Personalization engines	Proprietary (SaaS, Cloud)	Pinecone, Inc.
Milvus	<ul style="list-style-type: none">- High throughput- Designed for billion-scale vectors- GPU acceleration- Rich indexing options	<ul style="list-style-type: none">- Image/video search- Large-scale AI pipelines- IoT sensor data retrieval	Open-source + Managed (Zilliz Cloud)	Zilliz
FAISS	<ul style="list-style-type: none">- Extremely fast- Fine-tuned indexing control- Good for offline/local search	<ul style="list-style-type: none">- Research and prototyping- On-device vector search- Embedding similarity testing	Library (C++/Python, Local)	Meta (Facebook AI)
Chroma DB	<ul style="list-style-type: none">- Simple developer UX- Built-in embedding storage- Tight integration with LangChain- Lightweight	<ul style="list-style-type: none">- Prototyping RAG apps- Local vector stores- Rapid LLM app development	Open-source (local-first)	Chroma (open community project)

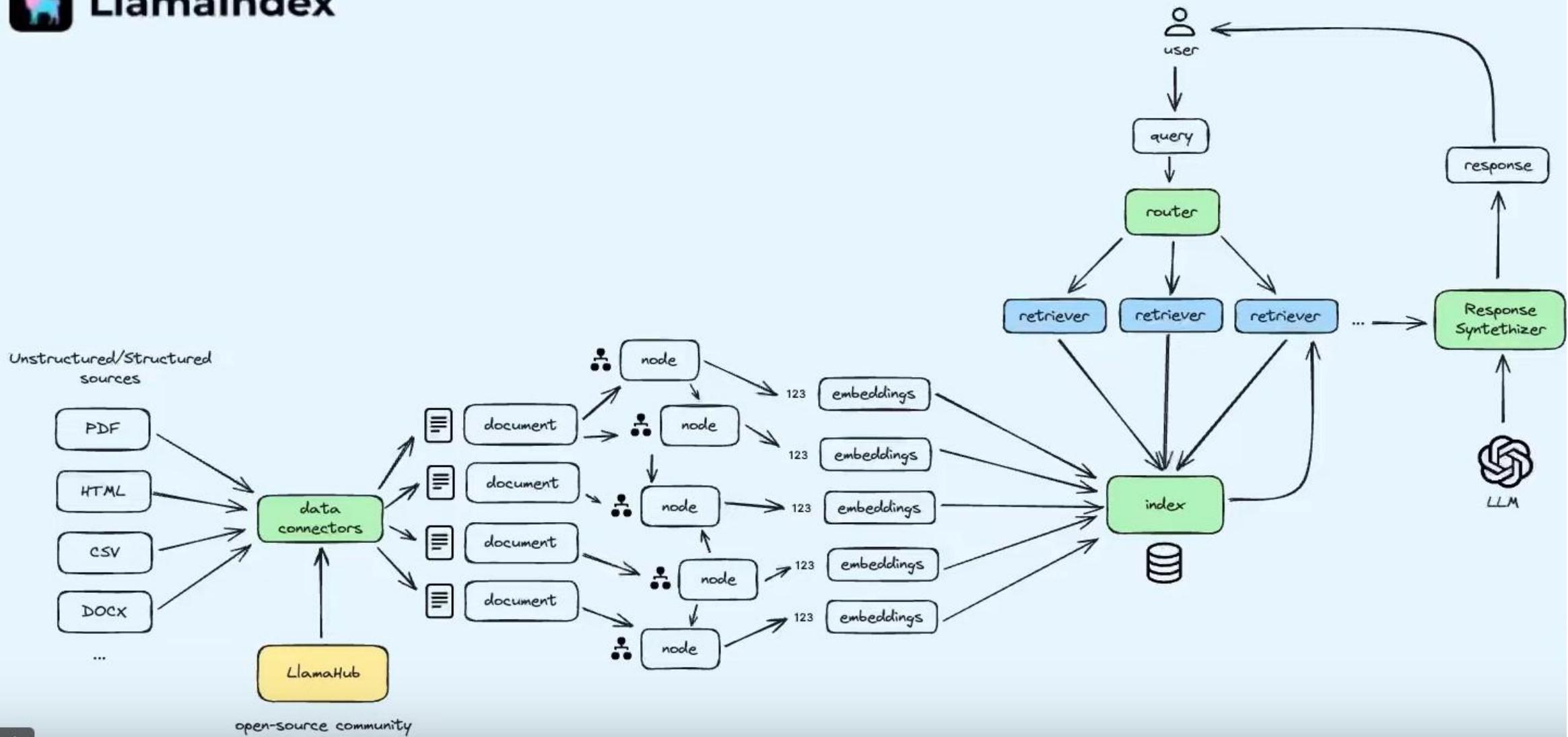
Vector DB and RAG

Retrieval Augmented Generation (RAG)



Retrieval Augmented Generation: LlamaIndex

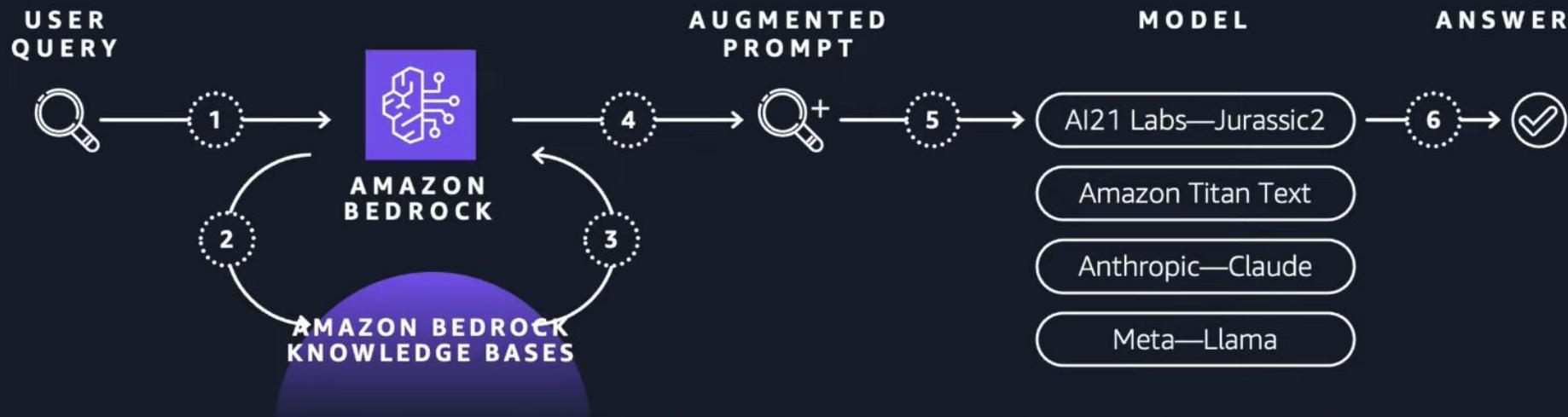
To move canvas, hold mouse wheel or spacebar while dragging, or use the hand tool



Retrieval Augmented Generation (RAG)

Amazon Bedrock Knowledge Bases

NATIVE SUPPORT FOR RAG



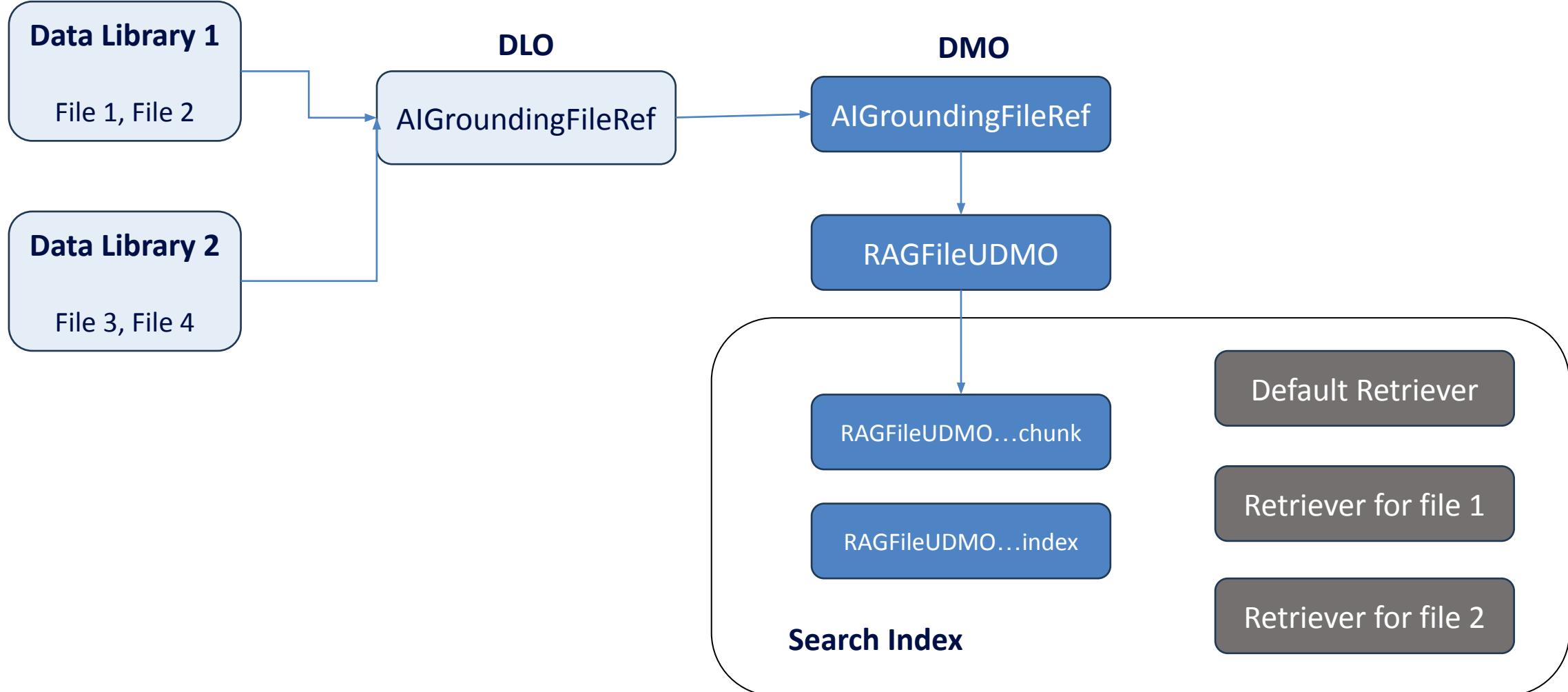
Securely connect FMs to data sources for RAG to deliver more relevant responses

Fully managed RAG workflow including ingestion, retrieval, and augmentation

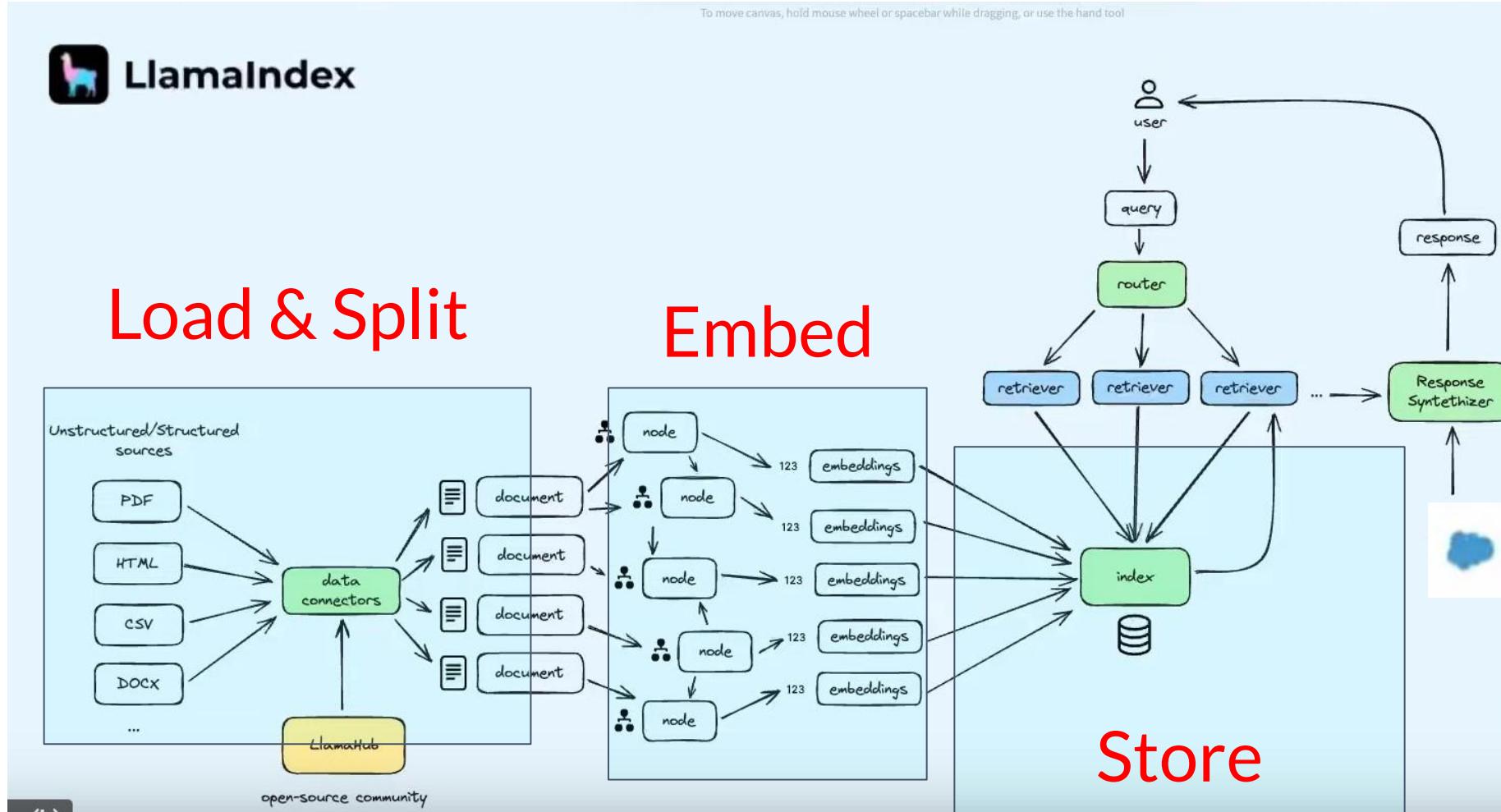
Built-in session context management for multturn conversations

Automatic citations with retrievals to improve transparency

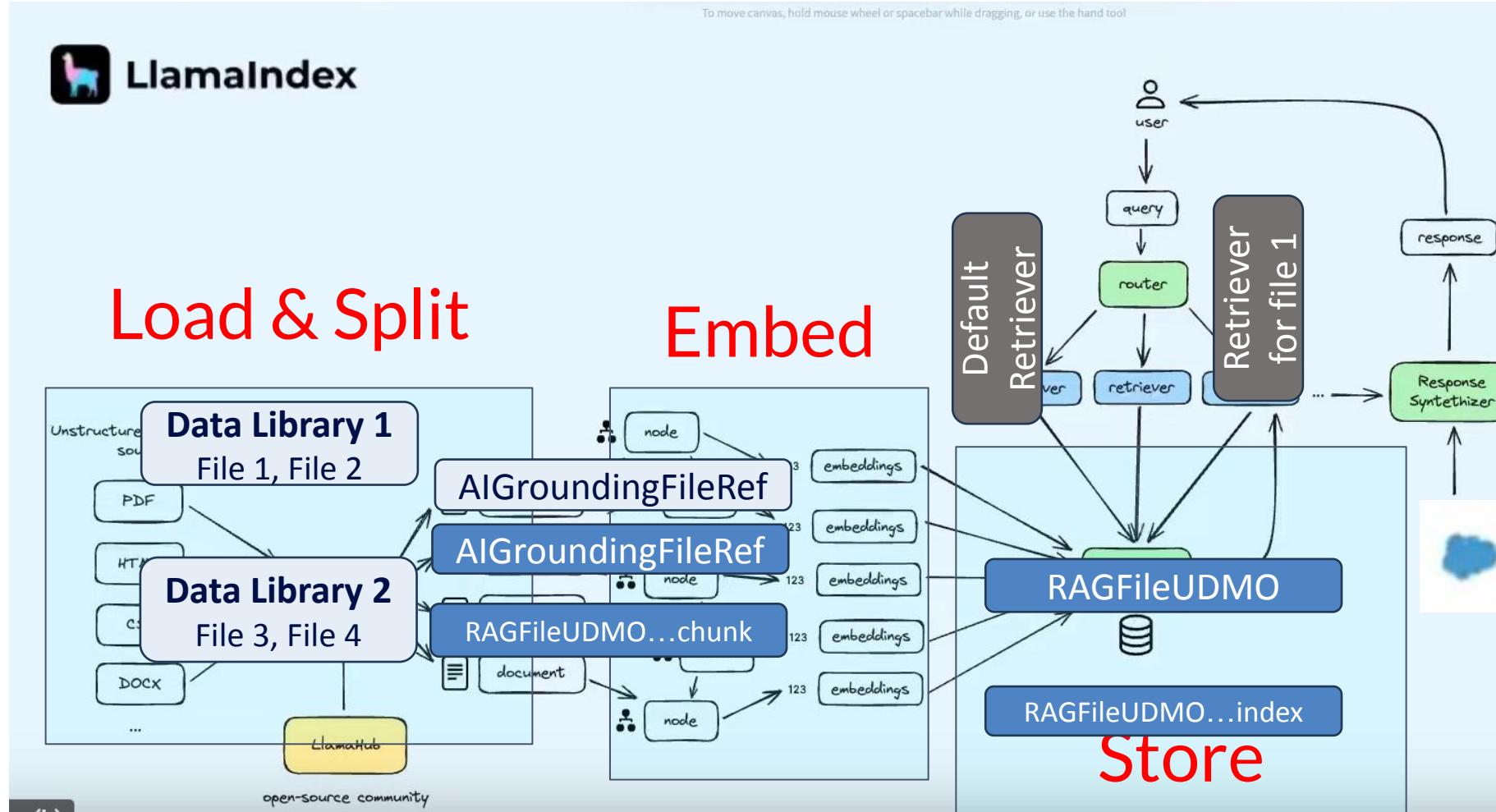
RAG in Agentforce Salesforce



Retrieval Augmented Generation (RAG)



Retrieval Augmented Generation



Useful Links



<https://www.youtube.com/@phiai1618>

Embedding & Support Ticket Classification

<https://www.youtube.com/watch?v=NIJ1yS0F03M>

https://github.com/enoten/support_ticket_analysis/blob/main/

How To Ask Questions to LLMs

<https://www.youtube.com/watch?v=E4k1qGFemSE>

https://github.com/enoten/huggingface_llms_apps/blob/main/