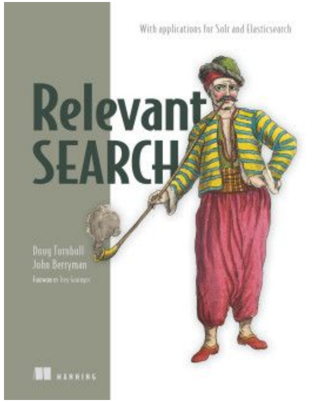
The Neural Search Frontier

How Deep Learning is poised to revolutionize the search relevance landscape

Adobe

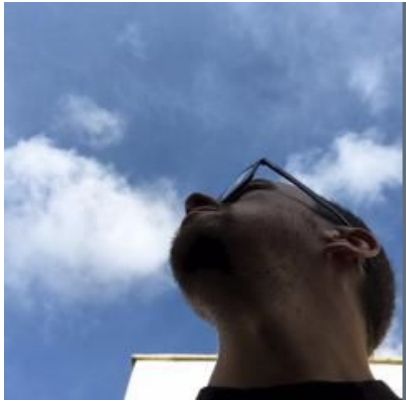
OpenSource Connections

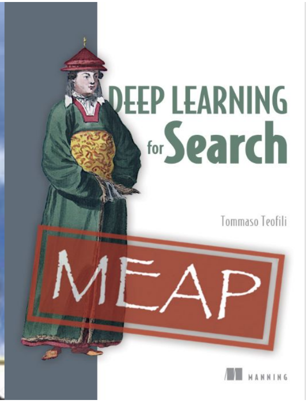
About us



Doug Turnbull, CTO - OpenSource Connections
(http://o19s.com)

Relevance raining July 10,11!
https://opensourceconnections.com/blog/2018/07/10/think-like-relevance-training/



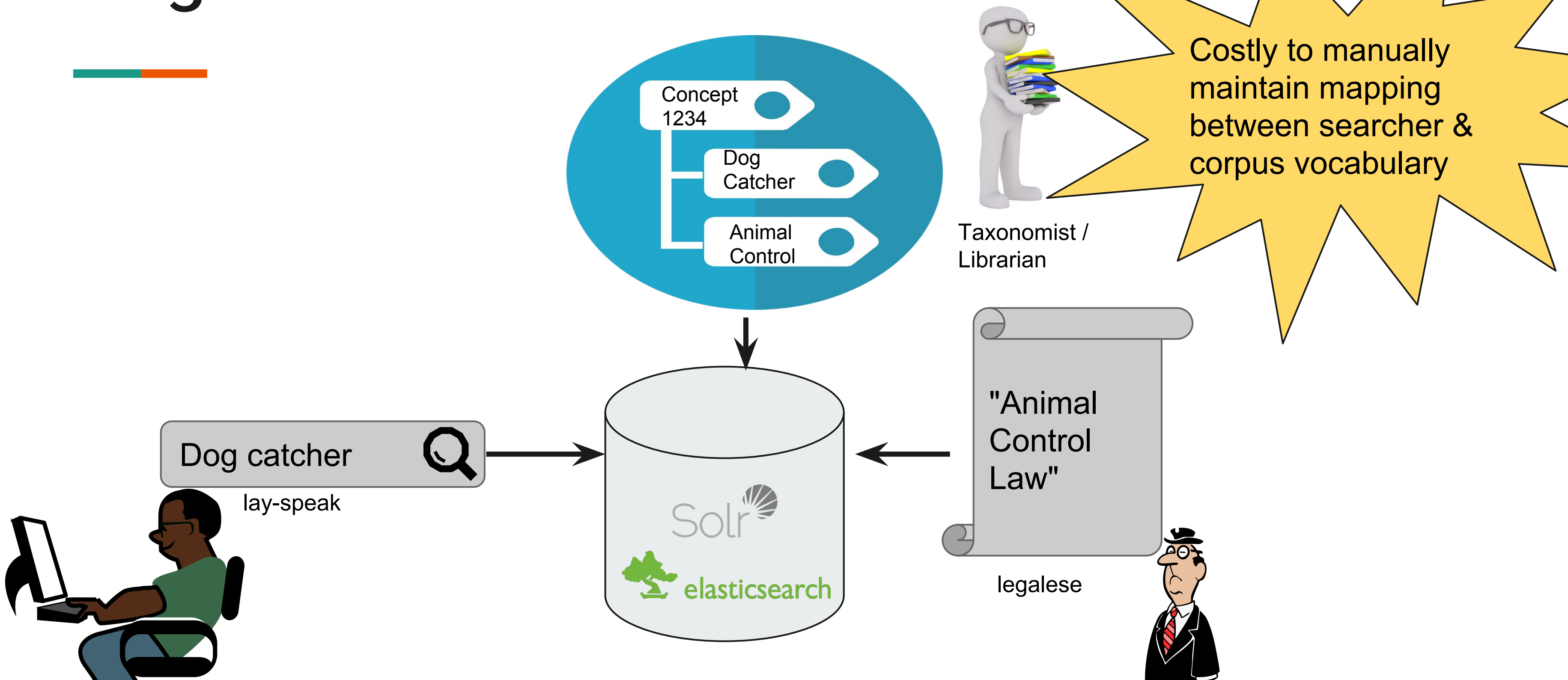


Tommaso Teofili

Computer Scientist, Adobe
ASF member

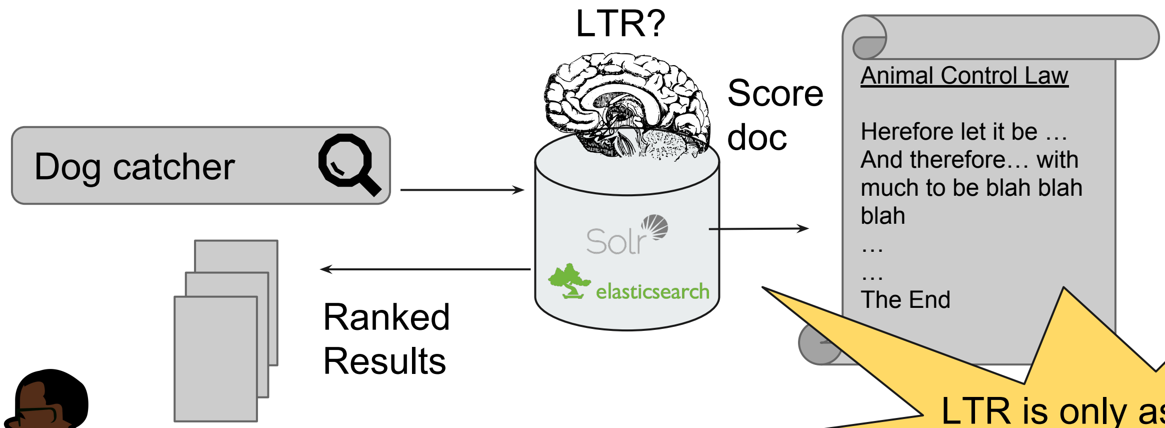
Search Relevance Challenges

Matching: vocab mismatch!



No Match, despite relevant results!

Ranking optimization hard!



Ranking can be hard optimization problem: Fine tuning heuristics: TF*IDF, Solr/ES queries, analyzers, etc...

LTR is only as good as your features: Solr/ES queries based on your skill with Solr/ES queries

Adobe OpenSource Connections

Embedding-Based Ranking

Embeddings

Word representations that ‘remember’ something about the <u>context</u> they tend appear in

Context: usually defined by the surrounding words

“<u>Cryptocurrency</u> is hyped?”

Appear in similar contexts

“Have you heard the hype about <u>bitcoin</u>?”

I would like to <u>cancel</u> my reservation

Appear in similar contexts

I would like to <u>confirm</u> my reservation

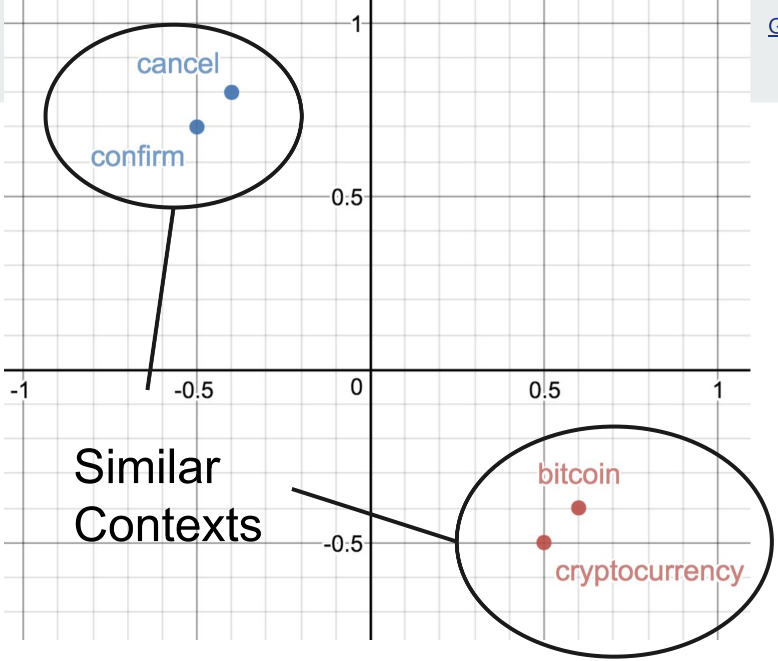
Embeddings are N-Dimensional Vectors that remember context

Graph from Desmos

2D Vector:

```
[0.5,-0.5] cryptocurrency
[0.6,-0.4] bitcoin

[-0.5,0.7] cancel
[-0.4,0.8] confirm
```



"Cryptocurrency is hyped?"

Appear in similar contexts

"Have you heard the hype about bitcoin?"

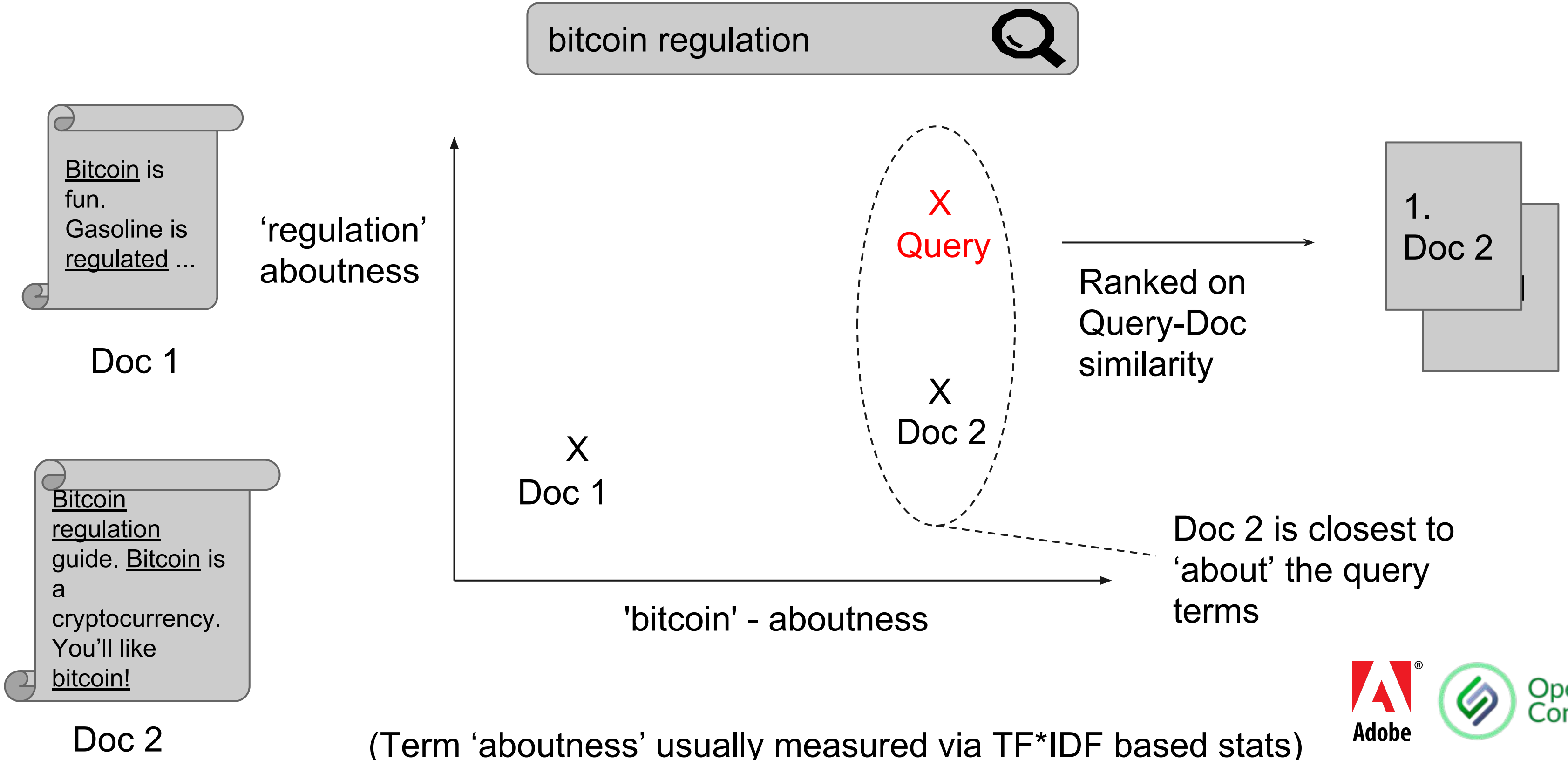
I would like to cancel my reservation

Appear in similar contexts

I would like to confirm my reservation

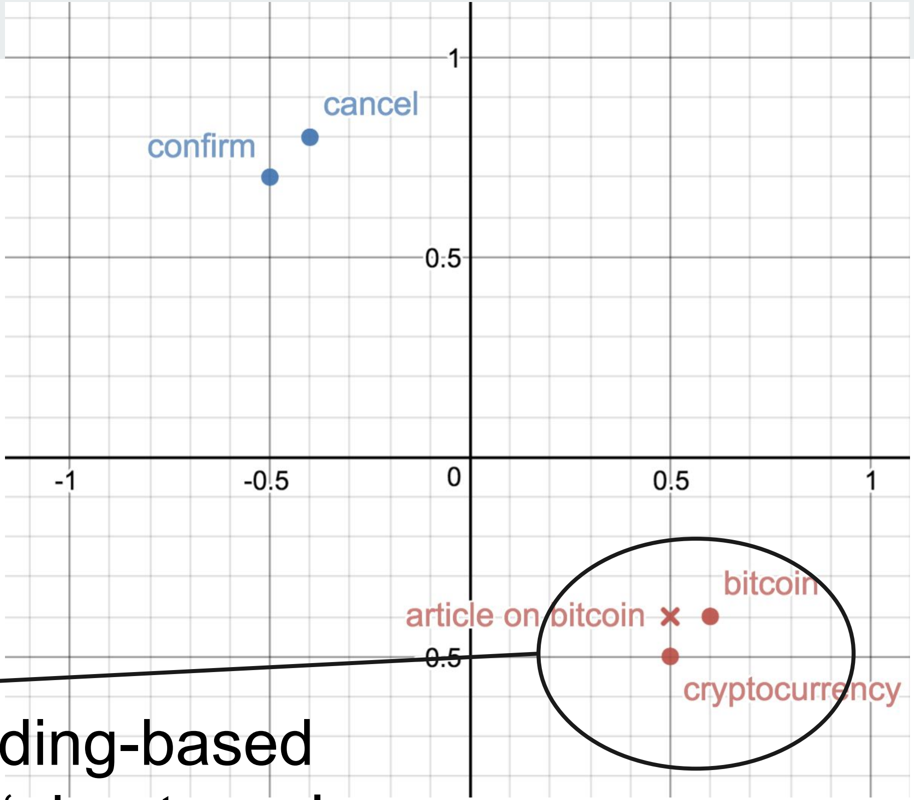
Adobe® OpenSource Connections

Search is based on similar vector similarity



(Term 'aboutness' usually measured via TF*IDF based stats)

Documents also can be embeddings



2D Vector:

```
[0.5,-0.5] cryptocurrency
[0.6,-0.4] bitcoin
[0.5,-0.4] article on bitcoin
```

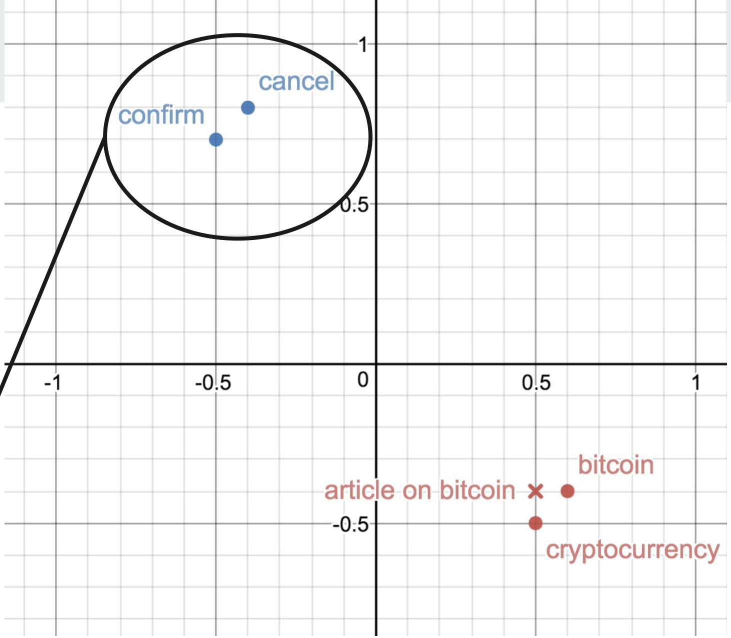
Hypothesis: embedding-based similarity related to ‘aboutness’ in relevance

(ie article is ‘about’ cryptocurrency even if it never says the term ‘cryptocurrency’)

Hypothesis doesn't always hold... (corpuses & search probs vary)

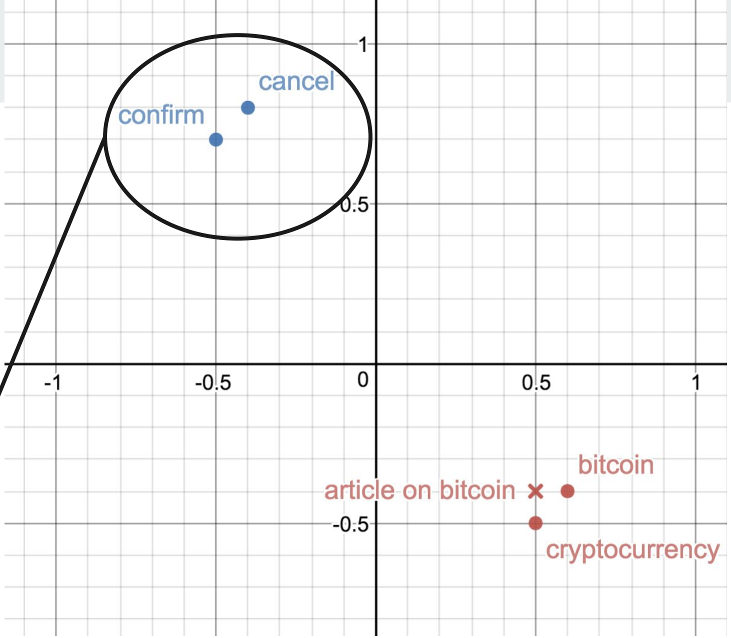
2D Vector:

```
[0.5,-0.5] cryptocurrency
[0.6,-0.4] bitcoin
[0.5,-0.4] article on bitcoin
```



Hypothesis Fails: articles here might be similar in embedding space, but may not map to user notions of relevance

Context alone is not enough (sometimes)



2D Vector:

```
[0.5,-0.5] cryptocurrency
[0.6,-0.4] bitcoin
[0.5,-0.4] article on bitcoin
```

We can adjust them using:

- Synsets
- Sentiment
- e.g. https://arxiv.org/abs/1805.07966

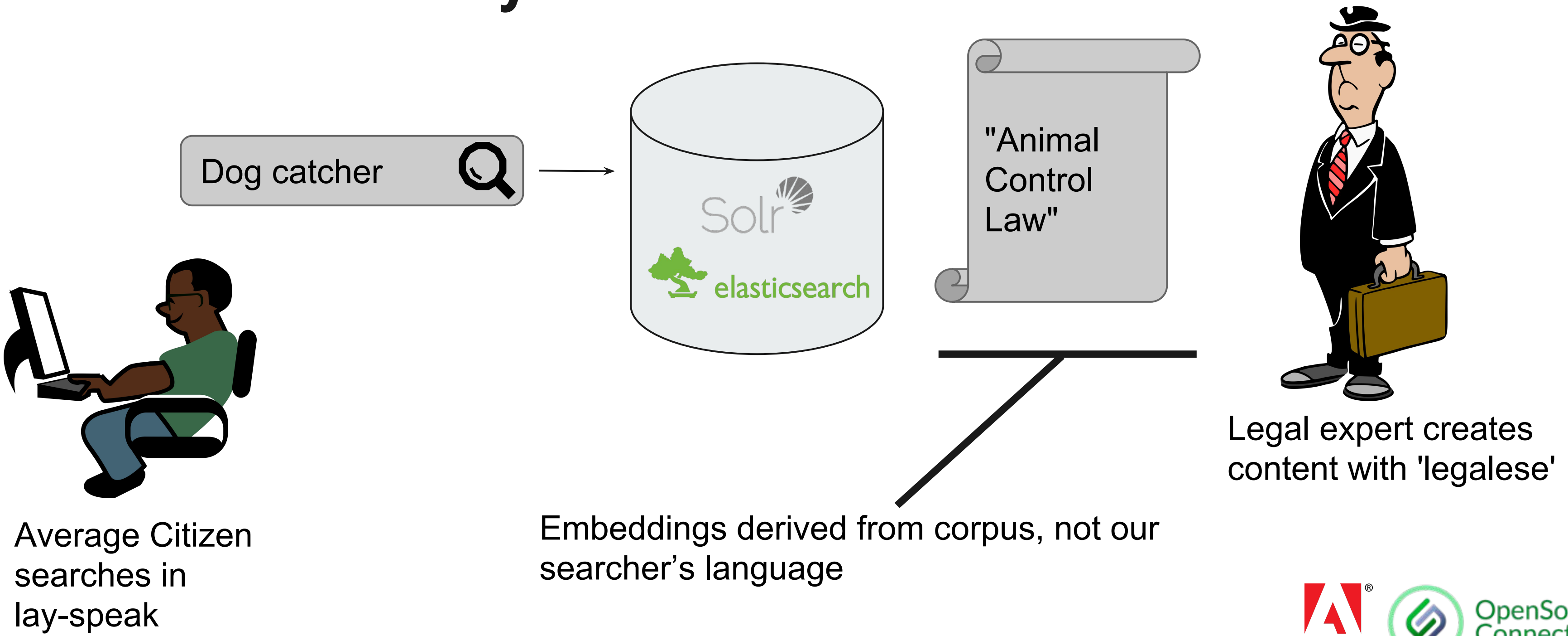
Low hanging fruit #1

The probability distribution of words in the corpus can provide enough information to predict whether two words appear in similar contexts

<u>Synonyms</u>

```
[0.5,-0.5] cryptocurrency
[0.6,-0.4] bitcoin
```

Fundamentally there's still a mismatch



Ranking-based embeddings

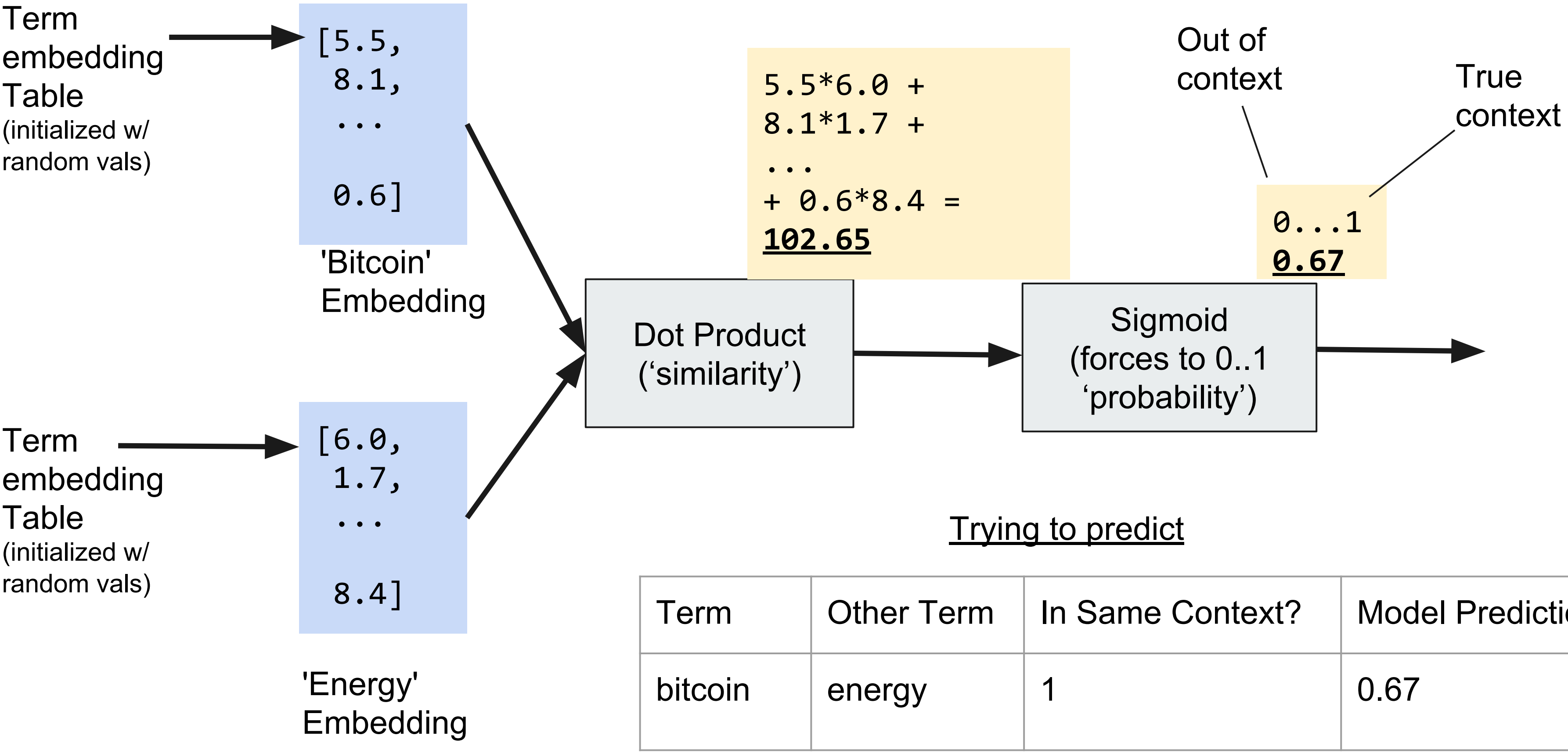
Embetterings - Neural Networks 101

Can we build embeddings that are closer to our searcher's notion of 'relevance'?

Maybe?

But to hack embeddings… we must first understand them

The word2vec Model



<u>Trying to predict</u>

Term	Other Term	In Same Context?	Model Prediction
bitcoin	energy	1	0.67

To get into the weeds on the loss function & gradient descent for word2vec:Stanford CS224D Deep Learning for NLP Lecture Notes Chaubard, Mundra, Socher

Tweak...

Terms sharing context

Random unrelated terms

Term	Other Term	True Context?	Prediction
bitcoin	energy	1	0.67
bitcoin	dongles	0	0.78
bitcoin	relevance	0	0.56
bitcoin	cables	0	0.34

Goal:
Tweak ‘bitcoin’ to get prediction closer to 1

Goal:
Tweak ‘bitcoin’ to get prediction closer to 0

(showing negative sampling method)

Maximize Me!

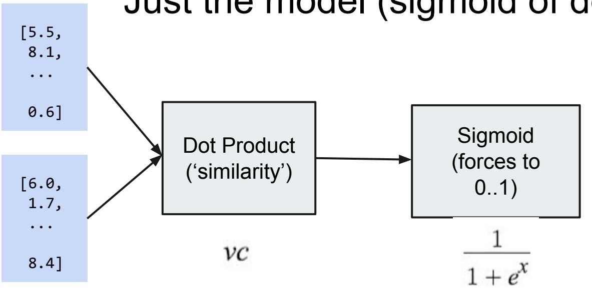
Maximize 'True' Contexts

Minimize 'False' Contexts

$$L = \log\left(\boxed{\frac{1}{1+e^{vc}}}\right) - \sum_{i=0}^{k} \log\left(\boxed{\frac{1}{1+e^{vc_i}}}\right)$$

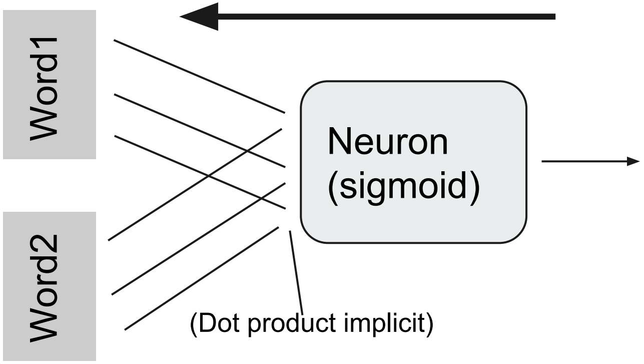
Just the model (sigmoid of dot product)

Objective /
'Loss' Function /
'Cost' Function



Did you just neural network?

Backpropagate tweaks to weights to reduce error



Build a loss function

$$\frac{d\,L}{d\,v[0]}$$

Pick weight to tweak up or down to minimize loss

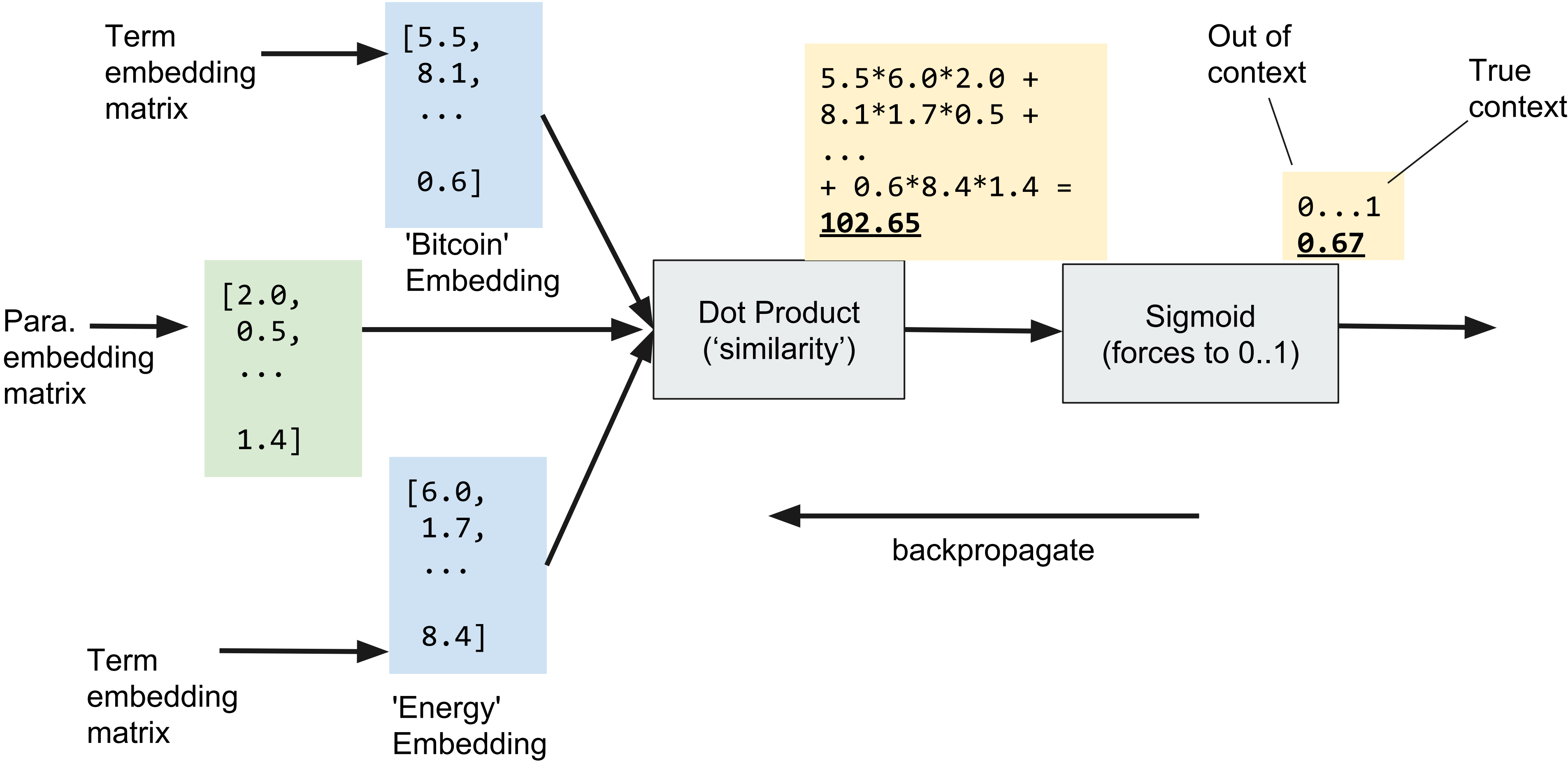
$$\frac{d\,L}{d\,v[1]}$$

$$\frac{d\,L}{d\,v[2]}$$

...

More complex/'deep' neural nets, can **propagate** back error to earlier layers to learn weights

Doc2Vec, train paragraph vector w/ term vectors



I'm continuing the thread of negative sampling, but doc2vec need not use negative sampling

Same neg sampling can apply here

Terms sharing context

Random unrelated docs

Term	Para	True Context?	Prediction
bitcoin	Doc 1234 Para 1	1	0.67
bitcoin	Doc 5678 Para 5	0	0.78
bitcoin	Doc 1234 Para 8	0	0.56
bitcoin	Doc 1537 Para 1	0	0.34

Tweak embeddings to get prediction closer to 1

Tweak embeddings to get prediction closer to 0

Low hanging fruit #2

Similar word embeddings lie close to one another

Similar document embeddings lie close to one another

Document embeddings lie close to word embeddings representing important words or topics for them

<u>Retrieval</u>
Did anyone say **terms**???

<u>Ranking</u>
Did anyone say **frequencies** ???

~~Low~~ hanging fruit #2

Corpus and search vocabularies can differ a lot

Combining traditional IR and neural models can help in filling the gaps

<u>Retrieval</u>
Terms and vectors

<u>Ranking</u>
Frequencies and distances

Neural Search Hack1: Relevance-based embeddings?

Query Term	Doc	Relevant?	Relevance Score
bitcoin	Doc 1234	1	0.67
bitcoin	Doc 5678	0	0.78
bitcoin	Doc 1234	0	0.56
bitcoin	Doc 1537	0	0.34

Relevant doc for term

Random unrelated docs

Tweak embeddings to get query-doc score closer to 1

Tweak embeddings to get prediction closer to 0

(derived from clicks, etc)

What about queries/docs we haven't seen?

Pretrain word2vec with corpus/sessions?

(a kind of prior)

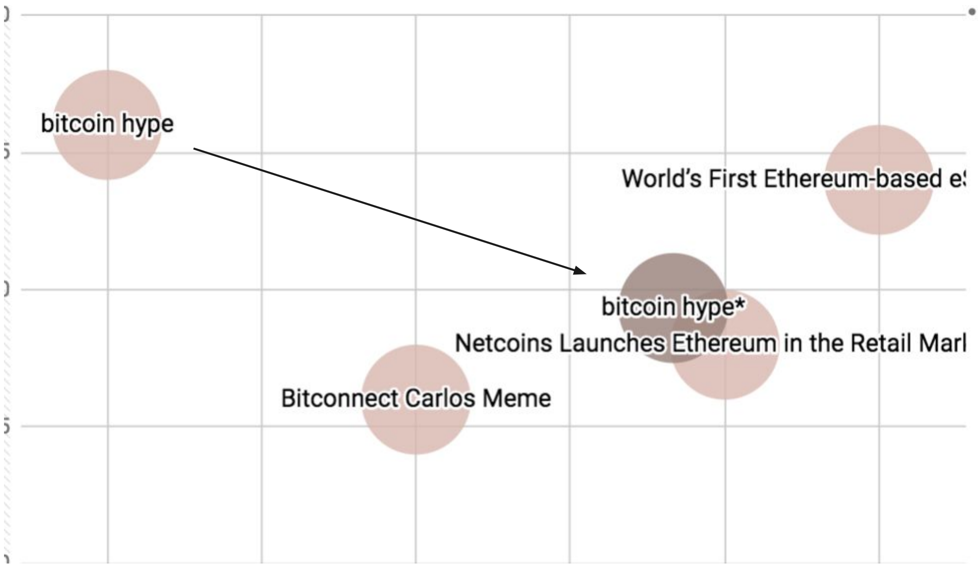
Query Term	Doc	Corpus Model	True Relevance	Tweaked Score
bitcoin	Doc 1234	0	1	0.01
bitcoin	Doc 5678	1	0	0.99
bitcoin	Doc 1234	0	0	0.56
bitcoin	Doc 1537	0	0	0.34

Takes a lot to overcome original unified model, and its a case-by-case basis

An online / evolutionary approach to converge on improved embeddings?

Adobe OpenSource Connections

Geometrically adjust query embeddings

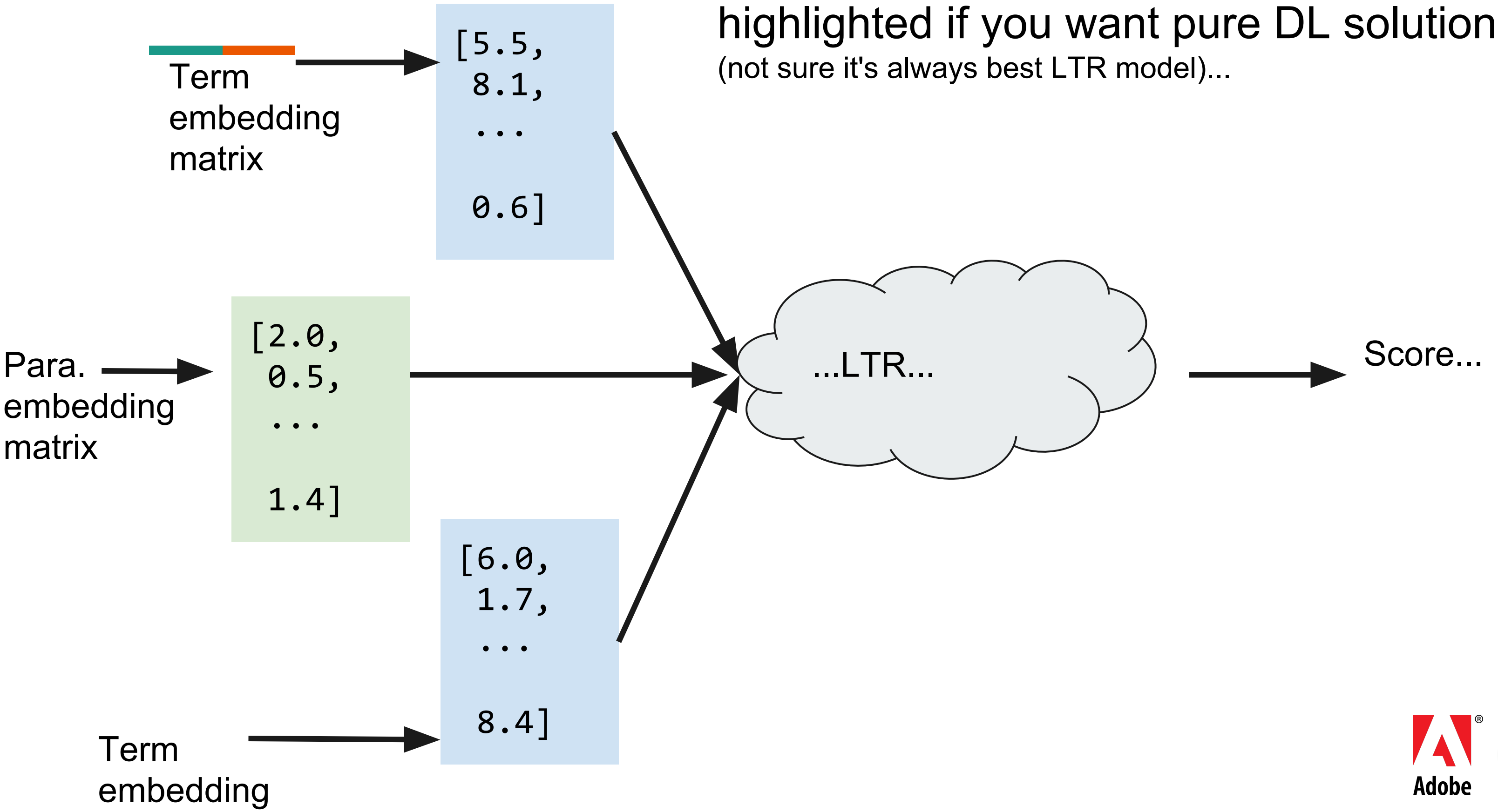


Averaging query vectors by means of some possibly relevant documents

Adobe OpenSource Connections

LTR feature...

See today's mobile.de talk where **RankNet** is highlighted if you want pure DL solution
(not sure it's always best LTR model)...



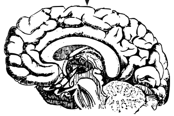
Adobe OpenSource Connections

Neural Language Models

Language Model

Given what's been spoken thus far, predict the next word / character

"eat __?______"



```
cat: 0.001
...
chair: 0.001
pizza: 0.02
...
nap: 0.001
```

Highest Probability

Obvious applications: autosuggest, etc

Adobe OpenSource Connections

Markov language model

From corpus we can build a **transition matrix** reflecting frequency of word transitions:

V words

V words

	pizza	chair	nap	...
eat	0.02	0.0002	0.0001	...
cat	0.0001	0.0001	0.02	...
...	...	...	...	

Probability of ‘pizza’ following ‘eat’ (as in “eat pizza”)

Markov model for embeddings?

Term embedding matrix

'h' dimensions

```
[5.5,
 8.1,
 ...

 0.6]
```

'eat' embedding

X

	1	2	3	...	h
0	0.2	-0.4	0.1		
1	-0.1	0.3	-0.24		
2					
...					
h					

A transition matrix of sorts - **weights** learned through backpropagation

=

```
[0.5,
 6.1,
 ...

 -4.8]
```

Embedding of next word

(probably clusters near 'pizza' etc)

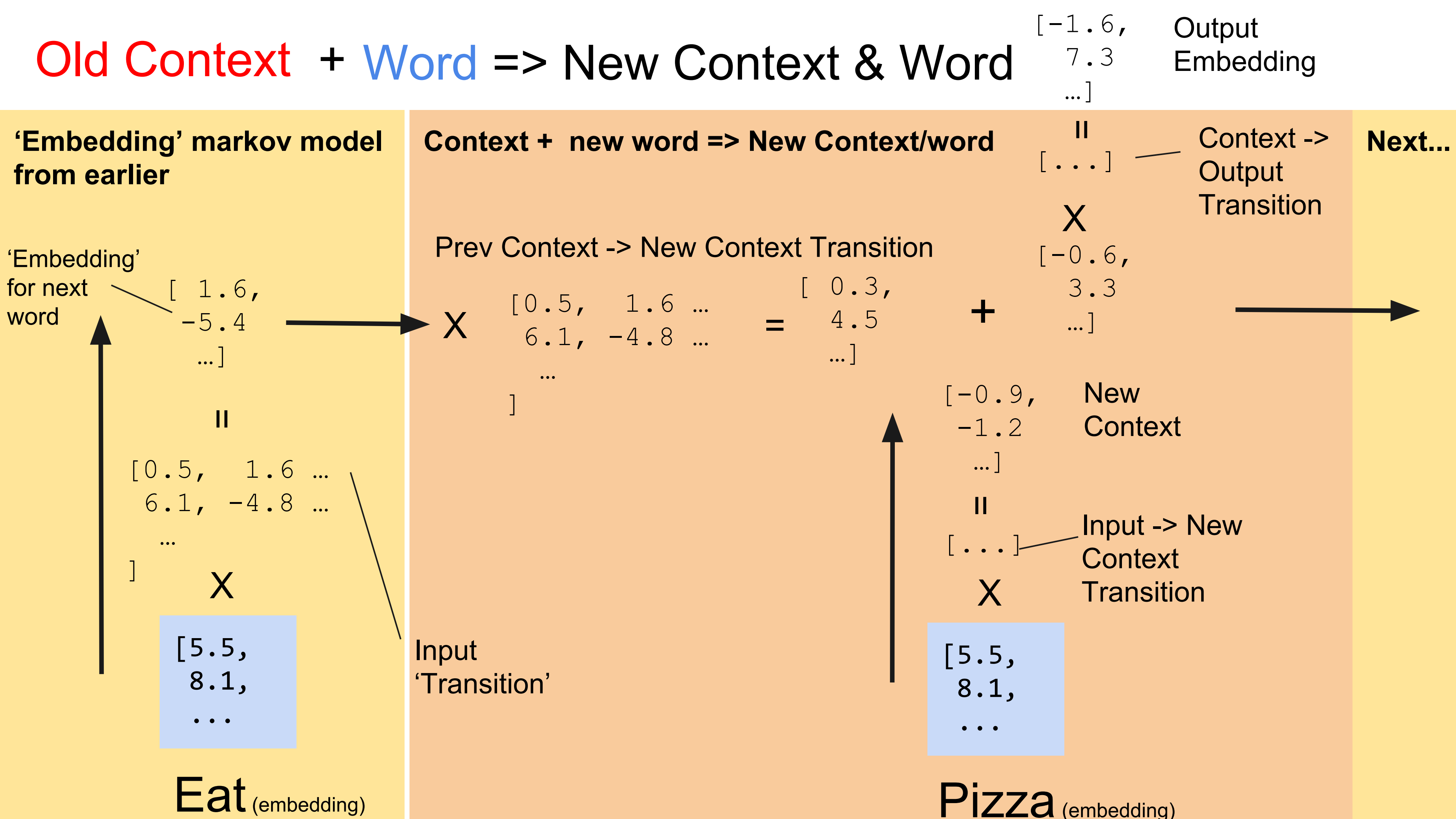
Accuracy requires context...

The race is on! eat pizza!

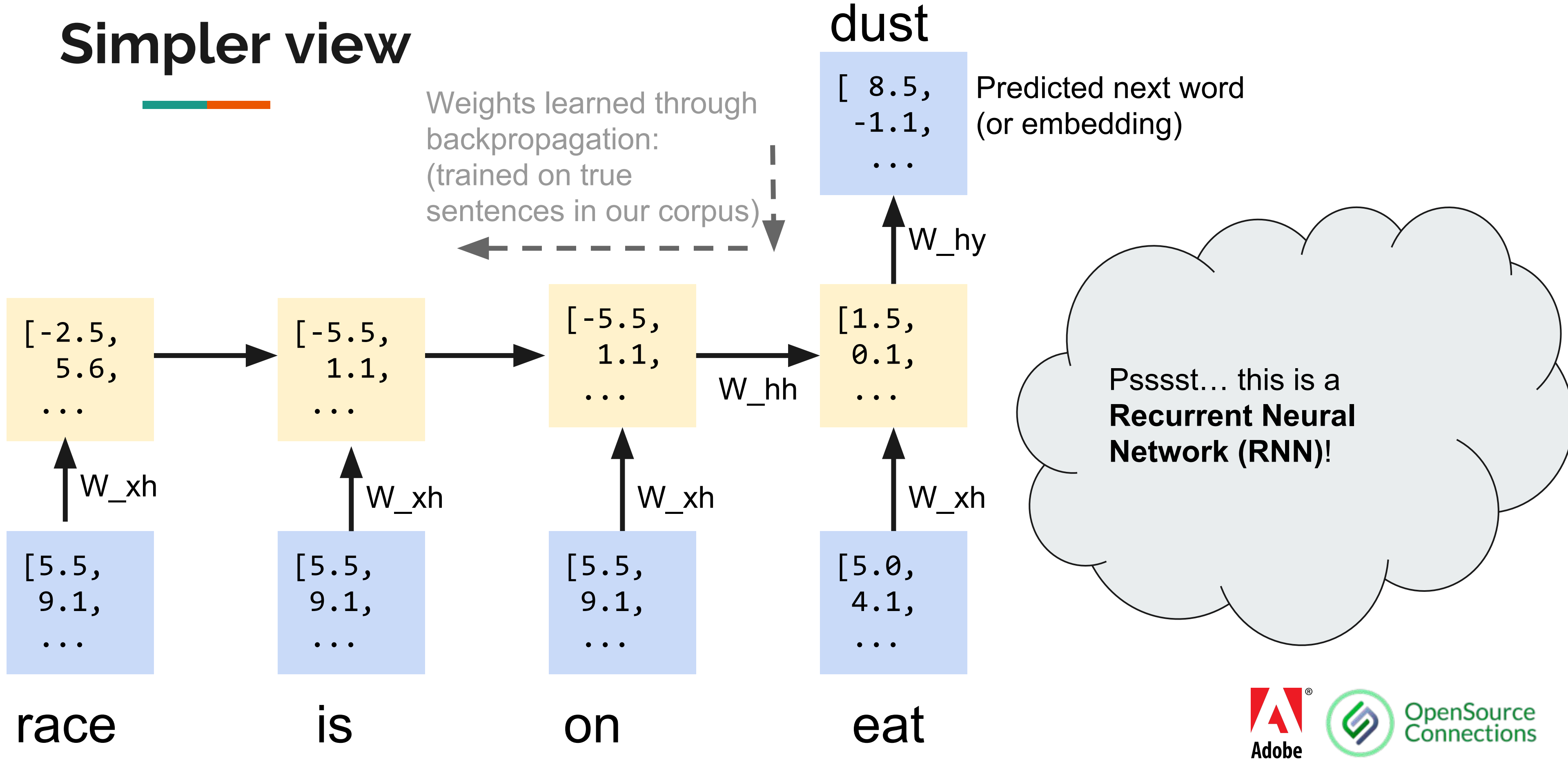
Context + Word => Next Word

(and new context)

Old Context + Word => New Context & Word



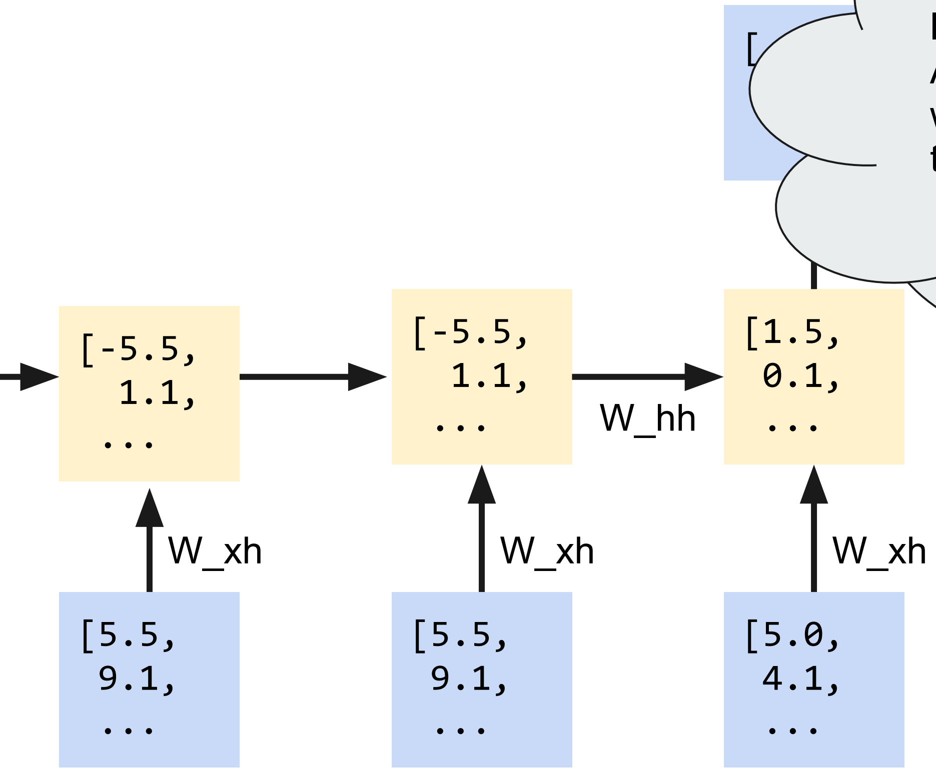
Simpler view

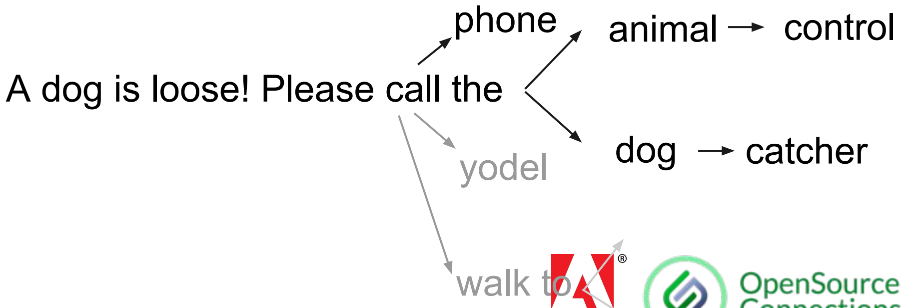


Adobe
OpenSource Connections

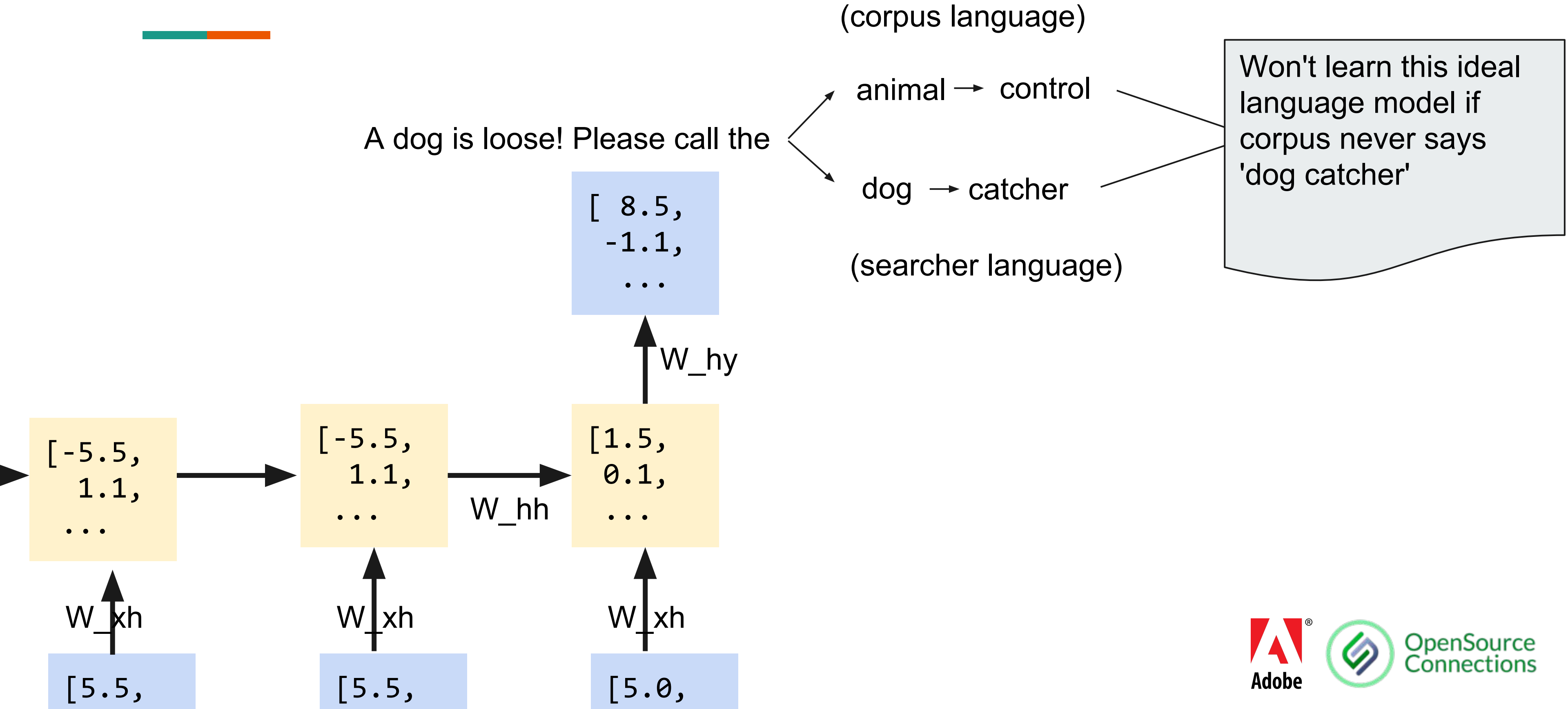
For search...

Inject other contextually likely terms:
At any given point scanning our document, we can get a probability distribution of likely terms





Not a silver bullet



One Vector Space To Rule Them All: Seq2Seq

Sessions as documents

Dog catcher

Dog bit me

Lost dog

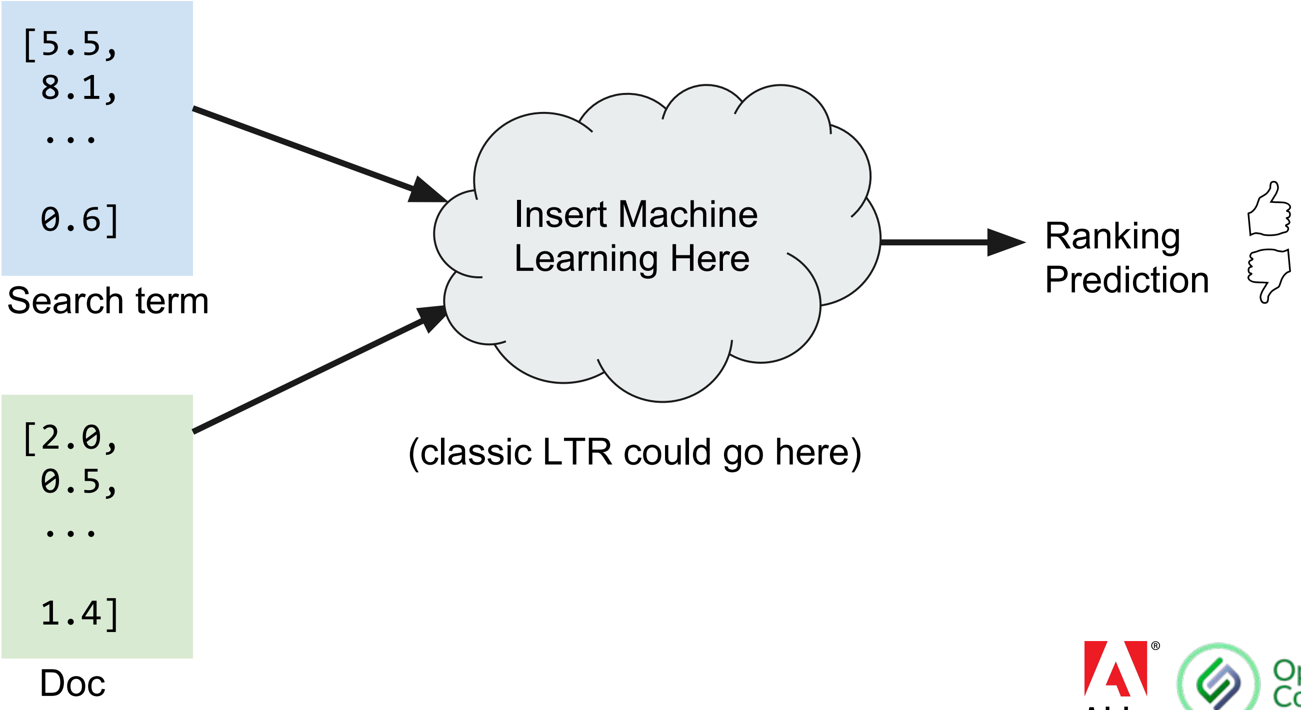
word2vec/doc2vec

```
[5.5,
 8.1,
 ...

 0.6]
```

Embeddings built for just *query terms*

Can we *translate* between searcher & corpus?

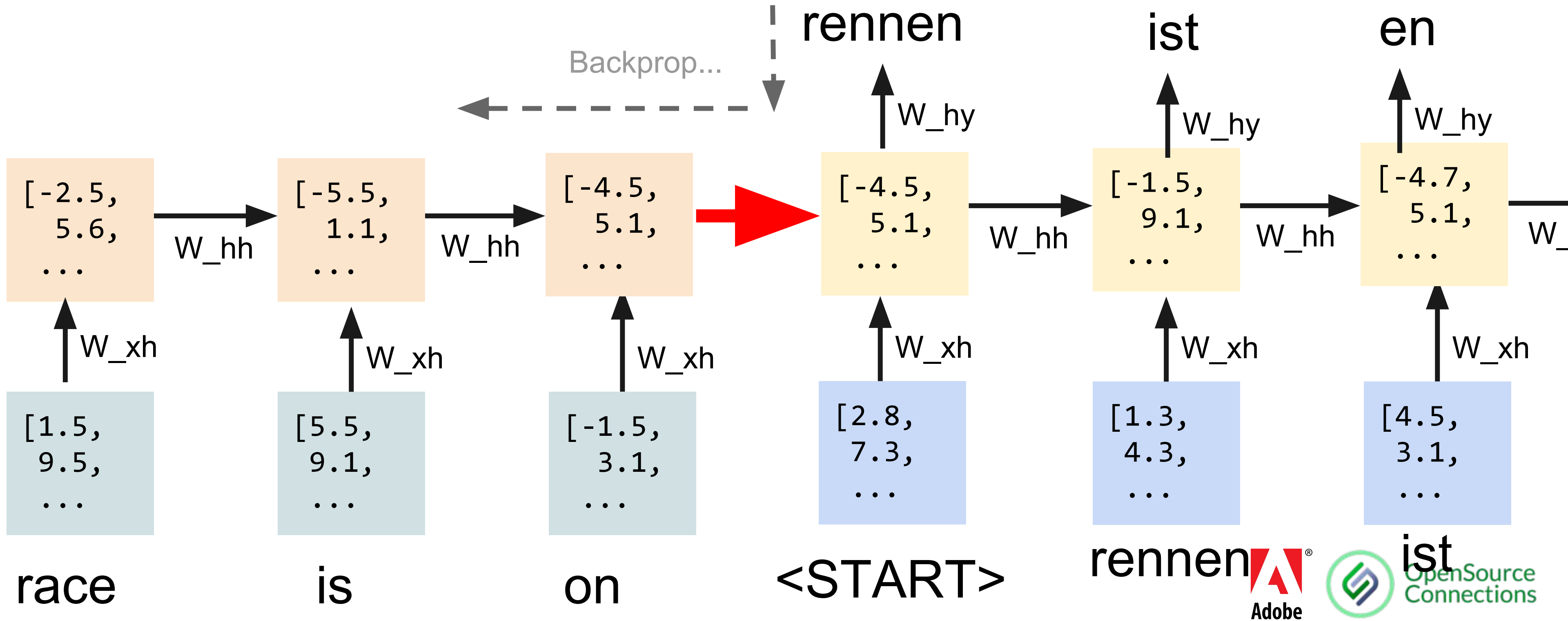


Seq2Seq The Clown Car of Deep Learning

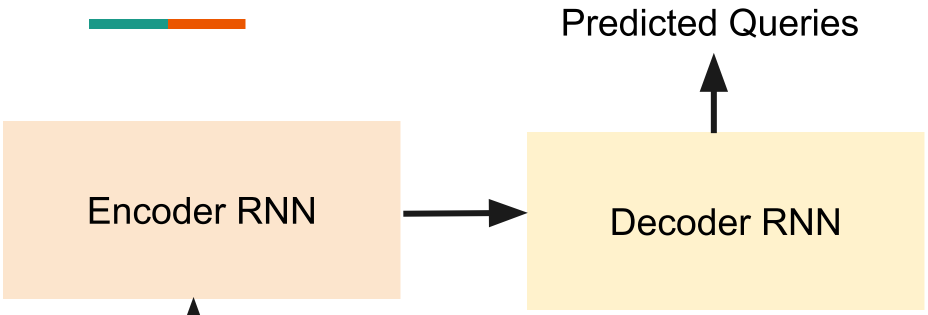
Translation: RNN decoder/encoder

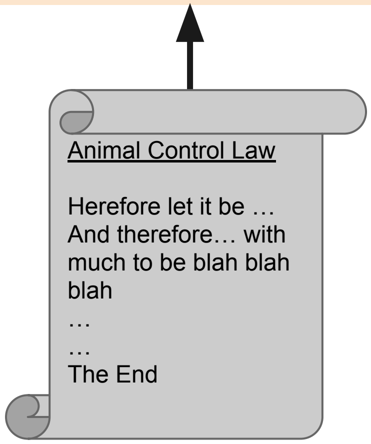
Encoder RNN

Decoder RNN



'Translate' documents to queries?

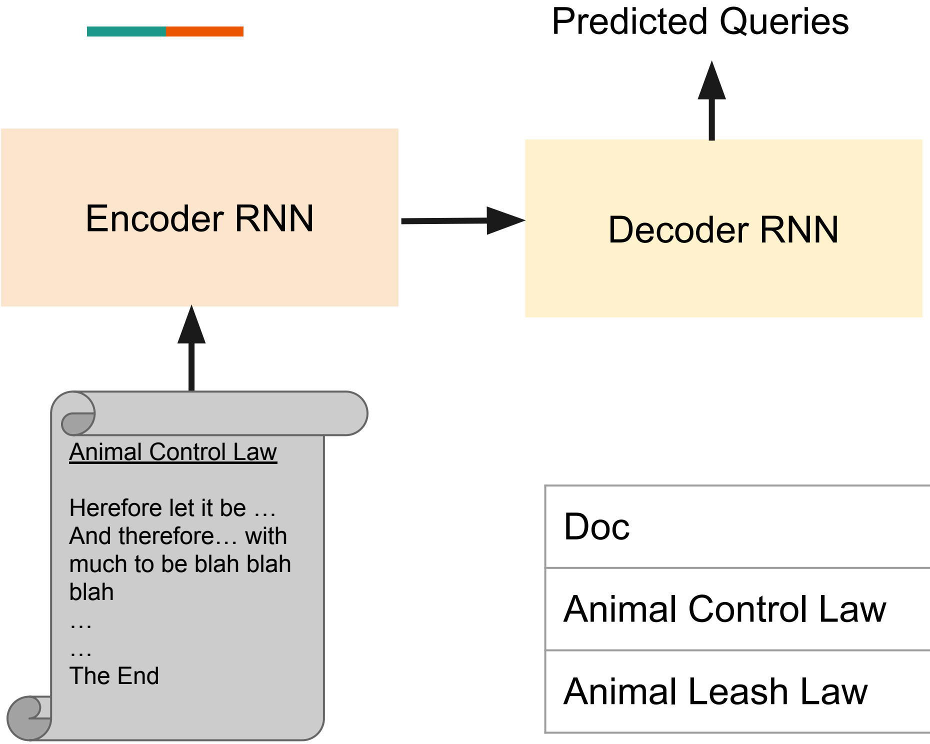




Using judgments/clicks as training data:

Doc	Query	Relevant?
Animal Control Law	Dog catcher	1
Littering Law	Dog Catcher	0

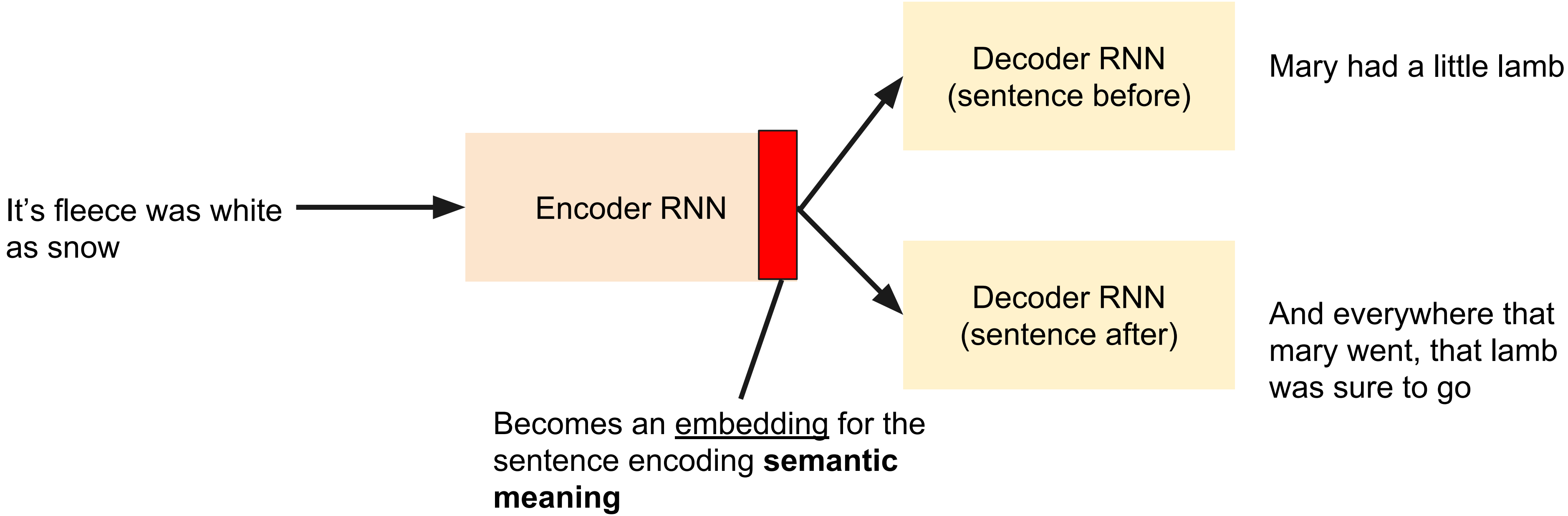
Can we use graded judgments?



Construct training data to reflect weighting

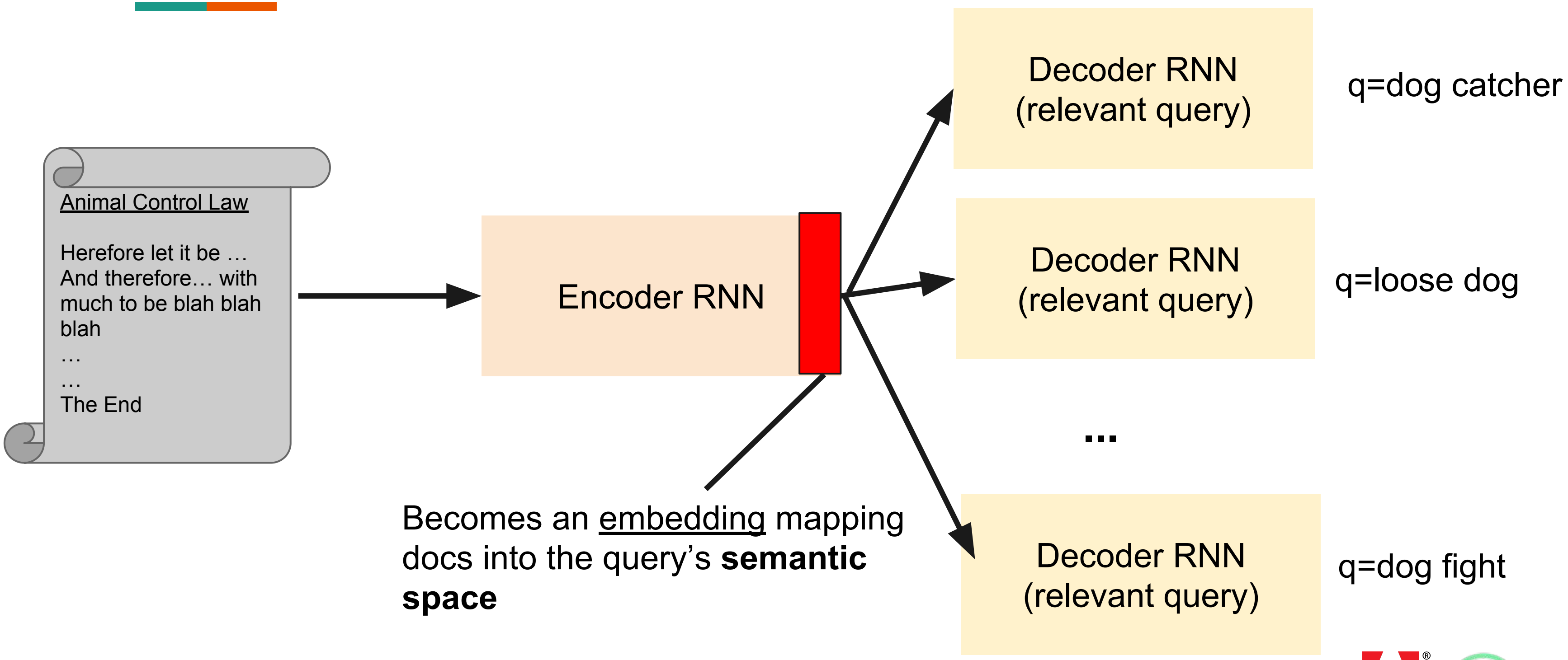
Doc	Query	Relevant?
Animal Control Law	Dog catcher	4
Animal Leash Law	Dog Catcher	2

Skip-Thought Vectors (a kind of ‘sentence2vec’)



[My Thoughts on Skip Thoughts]

Skip-Thought Vectors for queries? (‘doc2queryVec’)



Demo

The Frontier

Anything2vec

Deep learning is especially good at learning representations of :

- Images
- User History
- Songs
- Video
- Text

If everything can be ‘tokenized’ into some kind of token space for retrieval
Everything can be ‘vectorized’ into some embedding space for retrieval / ranking

Solr/ES community needs to get better first-class vector support

Similarity API and vectors ?

long computeNorm(**FieldInvertState** *state*)

SimWeight computeWeight(**float** *boost*, **CollectionStatistics** *collectionStats*, **TermStatistics**... *termStats*)

SimScorer simScorer(**SimWeight** *weight*, **LeafReaderContext** *context*)

Adobe® OpenSource Connections

Unsupervising the future

With supervised we can't get much further than the quality of the data itself

Can we do better with unsupervised learning ?

It’s not magic, it’s math!

Join the Search Relevancy Slack Community

http://o19s.com/slack

(projects (Elastic LTR!), chat, conferences (Haystack!), book authors, and more…)