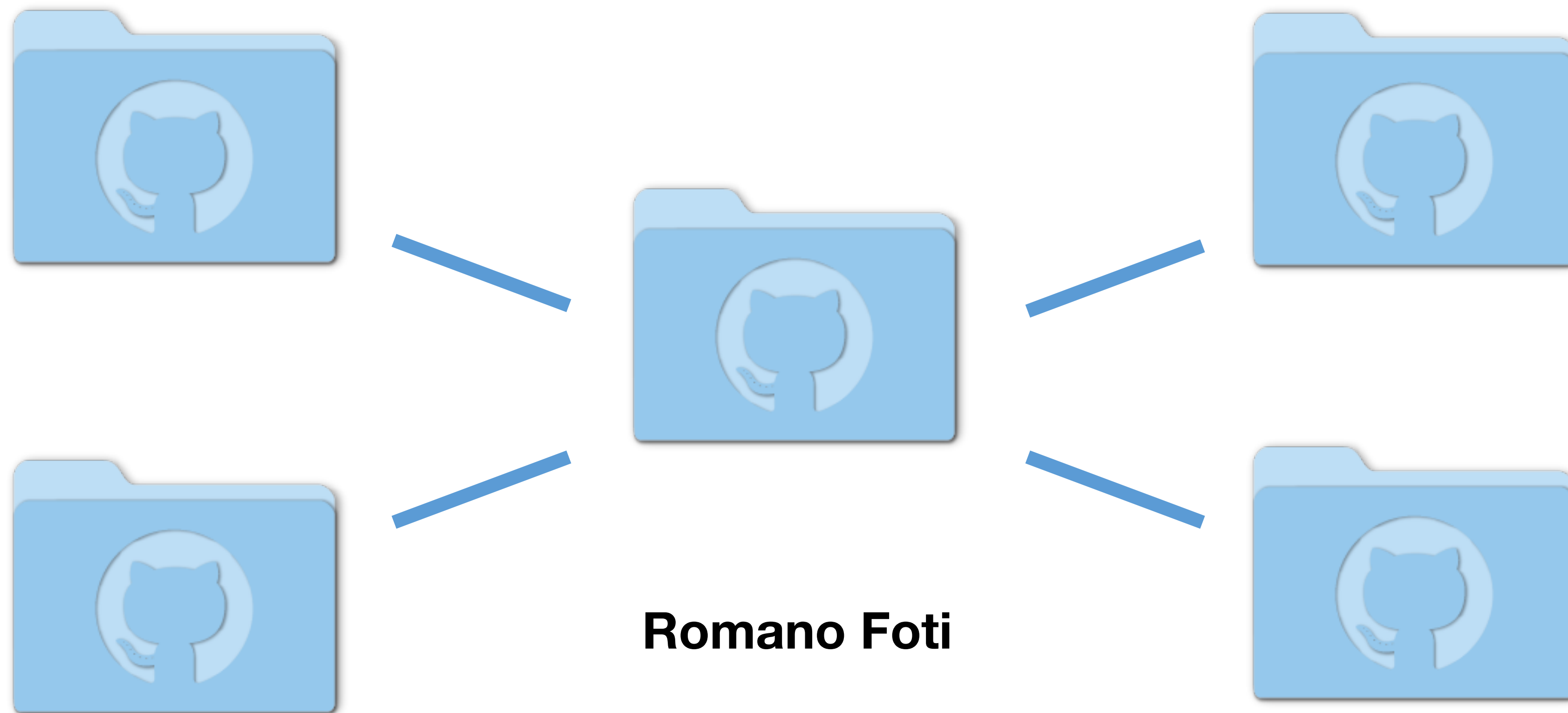


# OPEN DATA SCIENCE CONFERENCE



London | Nov. 19 - Nov. 22 2019

# Characterization of GitHub Repositories: A Natural Language Processing Approach



**Romano Foti**

2019 ODSC Europe  
London, November 2019



## Why is repository characterization important?

- Contain the vast majority of the code hosted at GitHub
- Primary GitHub resource for developers, contributors and students
- Facilitates content discoverability
- Promotes collaboration

## Challenges

- Almost 150M count
- Highly variable content
- Highly variable quality
- Meta-information not always informative or predictive
- Majority of content (code) is not written in a Natural Language



## Objective

**Characterizing repositories to provide personalized recommendations**

## Strategy

**Leveraging NLP without looking at repository content**

## The meaning of words



**How do people learn to speak, read, or write?**

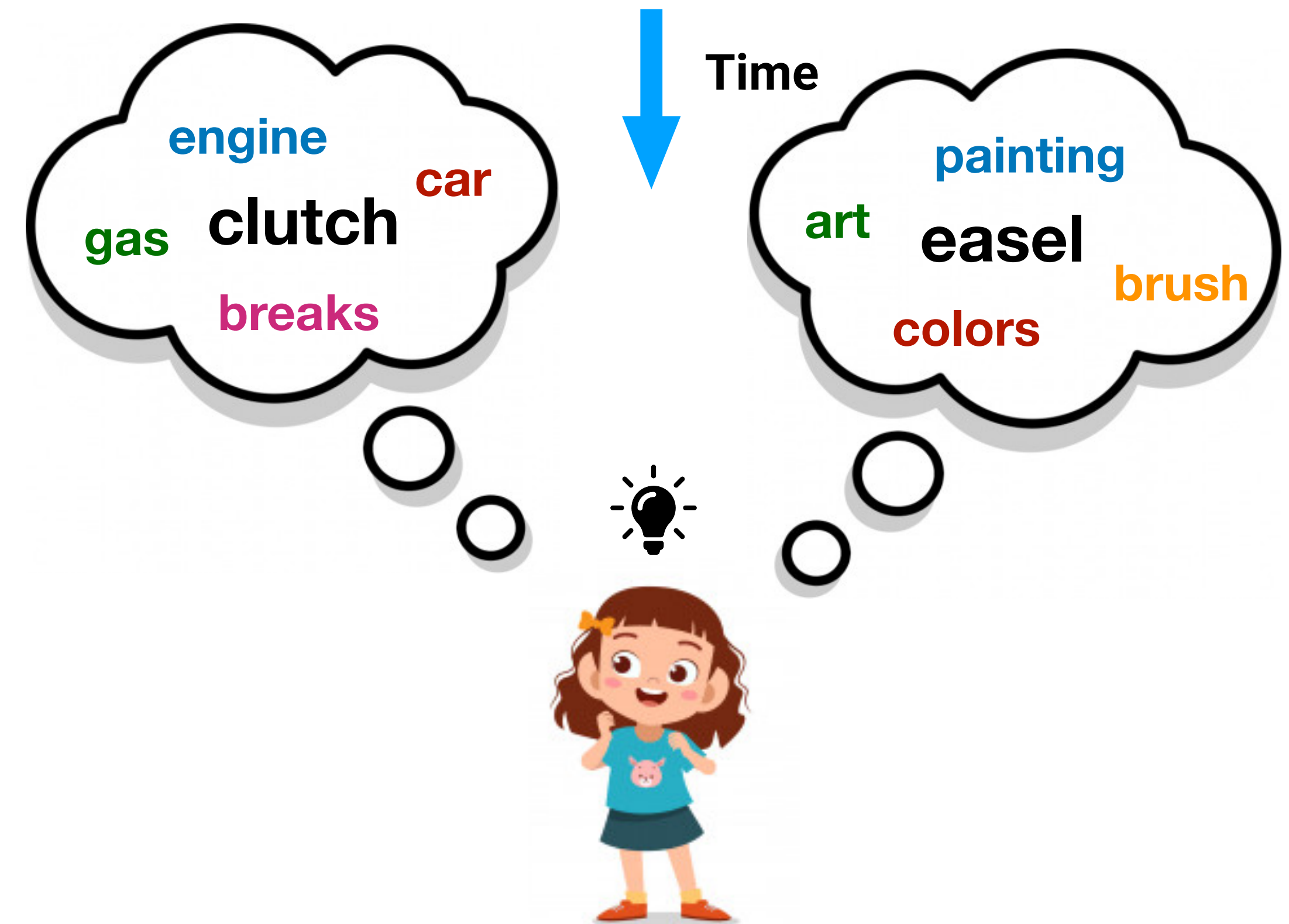
- No previous knowledge of the meaning of words
- No idea of how to build sentences with them

## The meaning of words



**How do people learn to speak, read, or write?**

- No previous knowledge of the meaning of words
- No idea of how to build sentences with them





## The meaning of words

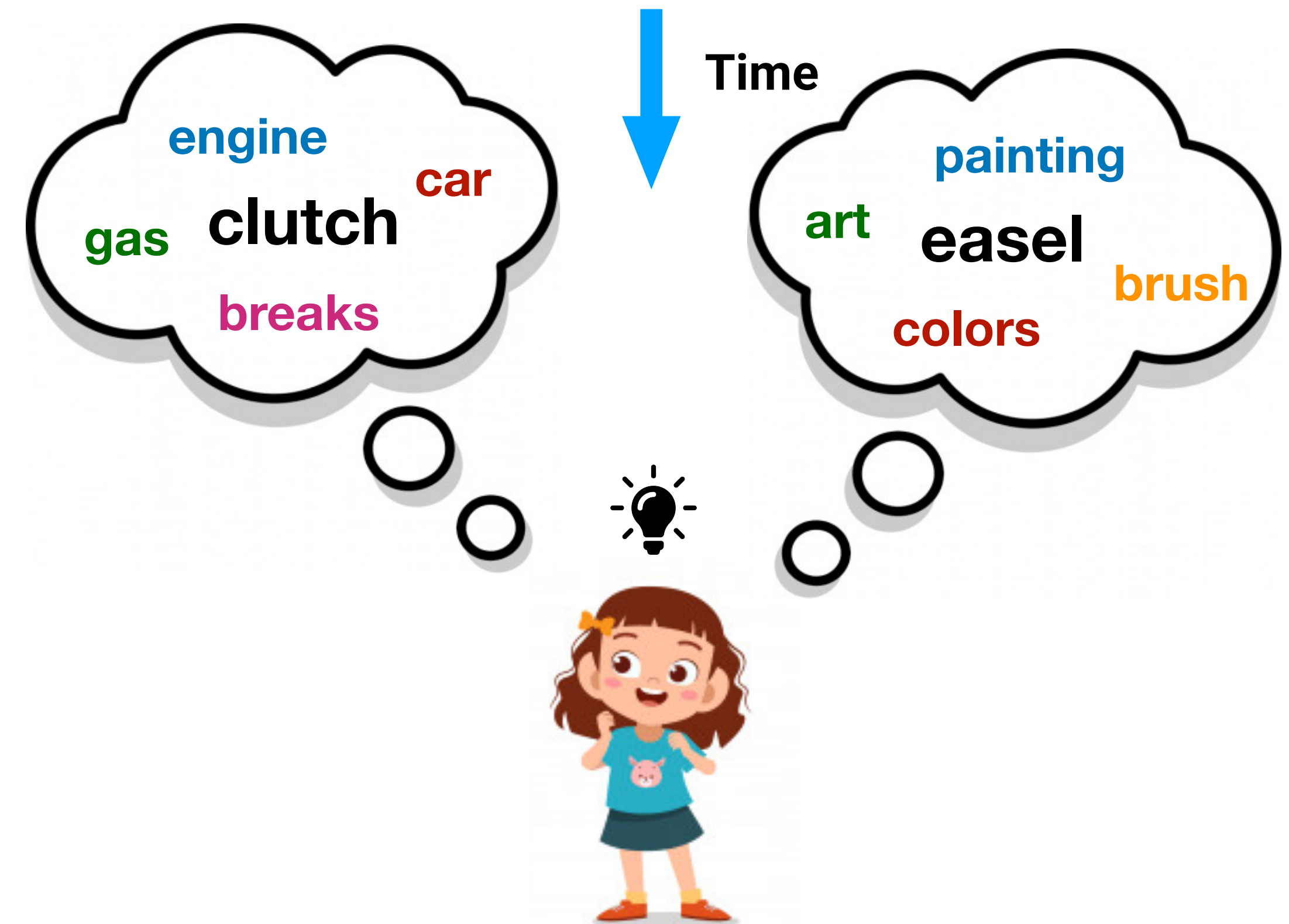


How?

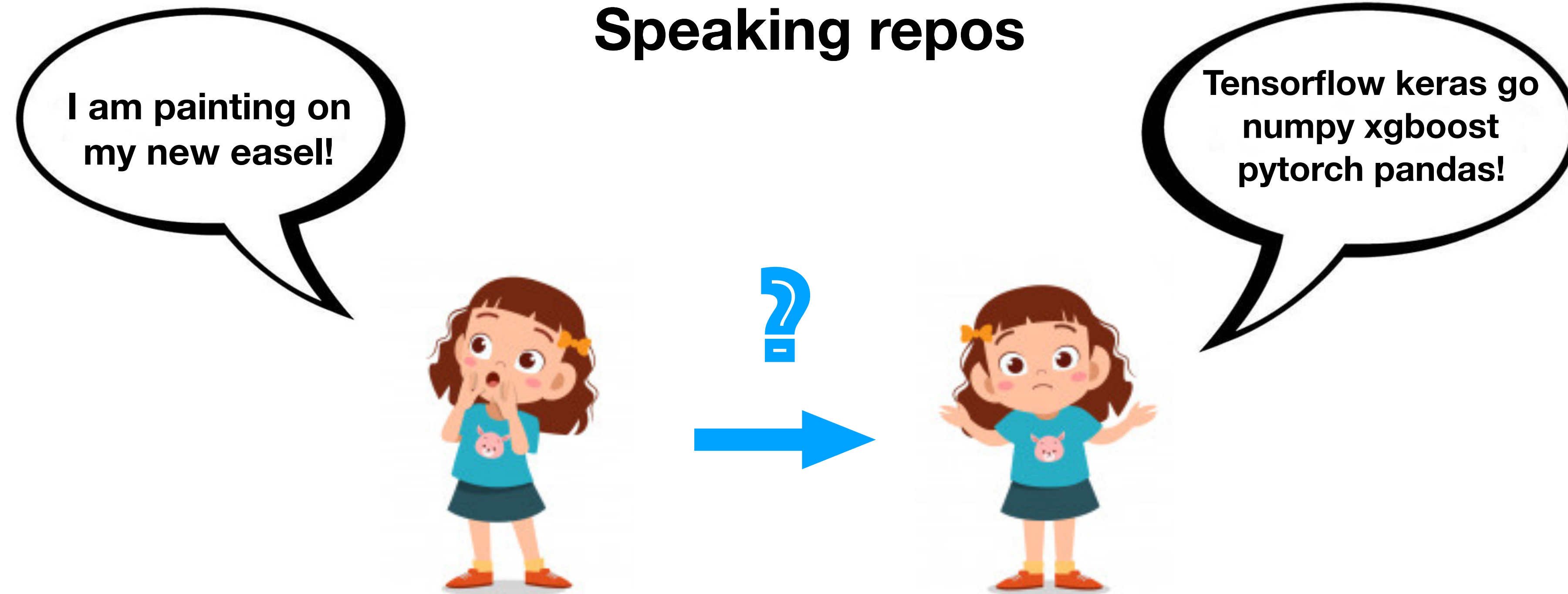
**Patterns & repetition**

**How do people learn to speak, read, or write?**

- No previous knowledge of the meaning of words
- No idea of how to build sentences with them

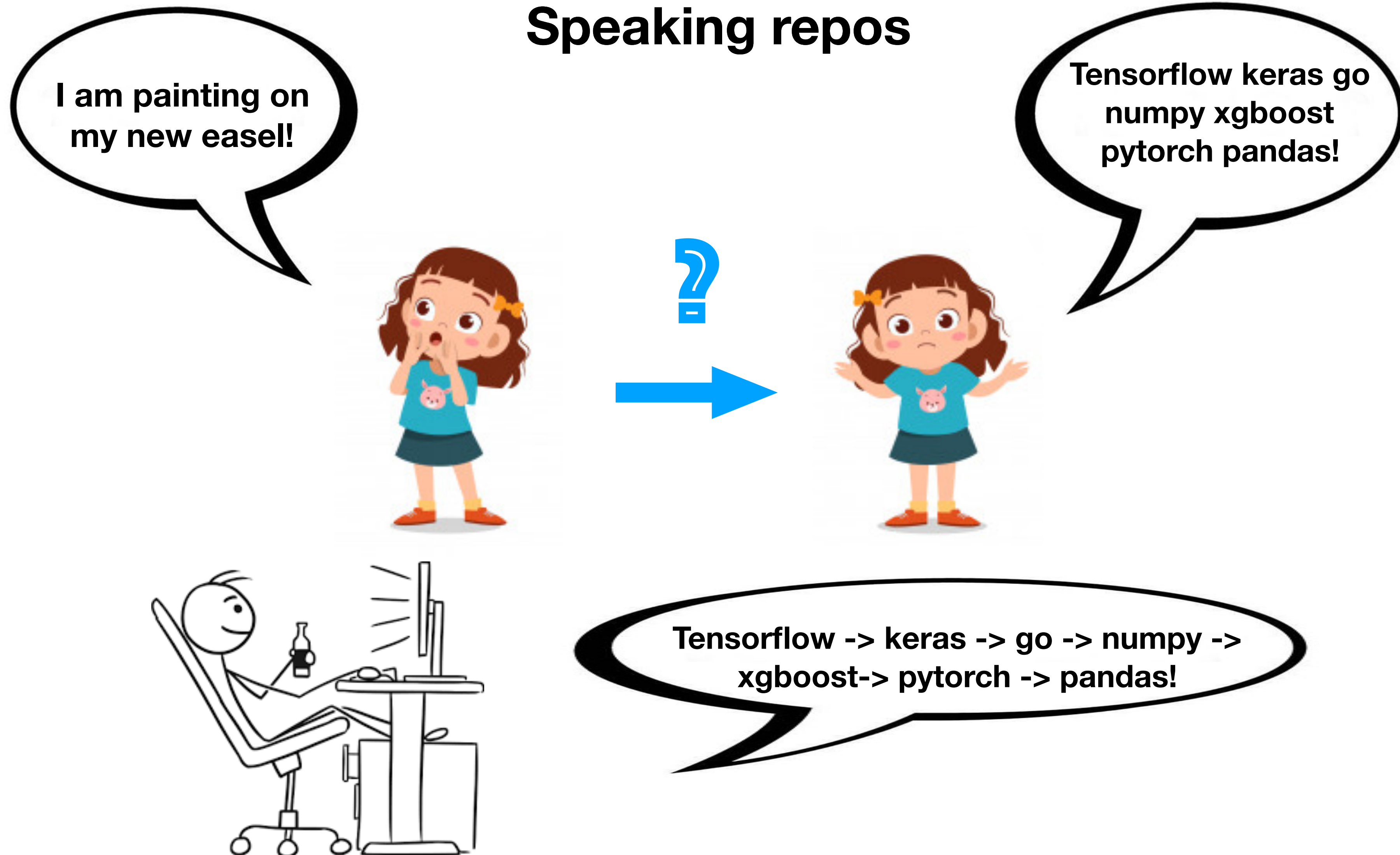


## Speaking repos





## Speaking repos





**Leveraging user activity to (indirectly) extract information on repositories**

## Leveraging user activity to (indirectly) extract information on repositories



sentence

Tensorflow -> keras -> go -> numpy -> xgboost-> pytorch -> pandas!

word

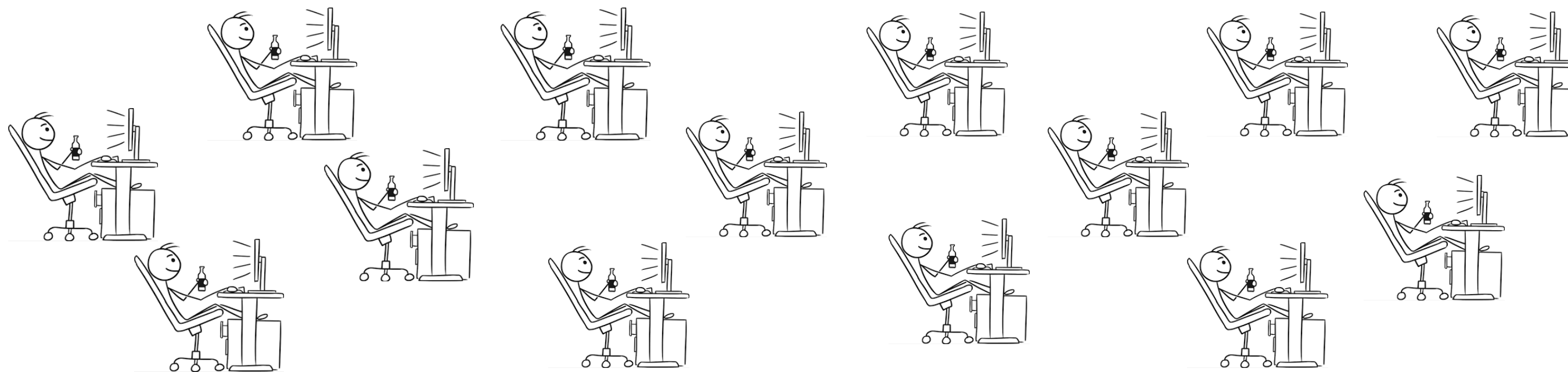
## Leveraging user activity to (indirectly) extract information on repositories



sentence

Tensorflow -> keras -> go -> numpy -> xgboost-> pytorch -> pandas!

word



Patterns & repetition



## Doc2vec: Learning meaning from context

### Source text

Tensorflow   keras   go   numpy   xgboost   pytorch   pandas

### Source text (natural language)

I am painting with my new easel





## Doc2vec: Learning meaning from context

**Source text**

Tensorflow   keras   go   numpy   xgboost   pytorch   pandas

**Source text (natural language)**

I am painting with my new easel

## Doc2vec: Learning meaning from context

**Source text**

Tensorflow	keras	go
------------	-------	----

numpy   xgboost   pytorch   pandas

**Source text (natural language)**

I am painting with my new easel

## Doc2vec: Learning meaning from context

**Source text**

Tensorflow	keras	go
------------	-------	----

 numpy xgboost pytorch pandas

**Source text (natural language)**

I am painting with my new easel

Tensorflow

**Seed word**

keras	go
-------	----

**Context words**

## Doc2vec: Learning meaning from context

### Source text

Tensorflow	keras	go	numpy	xgboost	pytorch	pandas
------------	-------	----	-------	---------	---------	--------

### Source text (natural language)

I am painting with my new easel



### Source text

Tensorflow	keras	go	numpy	xgboost	pytorch	pandas
Tensorflow	keras	go	numpy	xgboost	pytorch	pandas
Tensorflow	keras	go	numpy	xgboost	pytorch	pandas
Tensorflow	keras	go	numpy	xgboost	pytorch	pandas

### Training samples

(tensorflow, keras) (tensorflow, go)

(keras, tensorflow) (keras, go)  
(keras, numpy)

(go, tensorflow) (go, keras)  
(go, numpy) (go, xgboost)

(numpy, keras) (numpy, go) (numpy, xgboost)  
(numpy, pytorch)

## Doc2vec: Model architecture

### Vocabulary

### OneHot representation

tensorflow	(1, 0, 0, 0, 0, 0, 0)
keras	(0, 1, 0, 0, 0, 0, 0)
go	(0, 0, 1, 0, 0, 0, 0)
numpy	(0, 0, 0, 1, 0, 0, 0)
xgboost	(0, 0, 0, 0, 1, 0, 0)
pytorch	(0, 0, 0, 0, 0, 1, 0)
pandas	(0, 0, 0, 0, 0, 0, 1)





## Doc2vec: Model architecture

Vocabulary	OneHot representation	Training pair: (tensorflow, keras)	
		Input	Output
tensorflow	(1, 0, 0, 0, 0, 0, 0)	(1, 0, 0, 0, 0, 0, 0)	
keras	(0, 1, 0, 0, 0, 0, 0)		
go	(0, 0, 1, 0, 0, 0, 0)		
numpy	(0, 0, 0, 1, 0, 0, 0)		
xgboost	(0, 0, 0, 0, 1, 0, 0)		(0, 1, 0, 0, 0, 0, 0)
pytorch	(0, 0, 0, 0, 0, 1, 0)		
pandas	(0, 0, 0, 0, 0, 0, 1)		



## Doc2vec: Model architecture

Vocabulary	OneHot representation	Training pair: (tensorflow, keras)	
tensorflow	(1, 0, 0, 0, 0, 0, 0)	Input	Output
keras	(0, 1, 0, 0, 0, 0, 0)	(1, 0, 0, 0, 0, 0, 0)	(0, 1, 0, 0, 0, 0, 0)
go	(0, 0, 1, 0, 0, 0, 0)		
numpy	(0, 0, 0, 1, 0, 0, 0)		
xgboost	(0, 0, 0, 0, 1, 0, 0)		
pytorch	(0, 0, 0, 0, 0, 1, 0)		
pandas	(0, 0, 0, 0, 0, 0, 1)		
Vocabulary size	7		
Embedding size	10 (arbitrary)		



# Doc2vec: Model architecture

Vocabulary

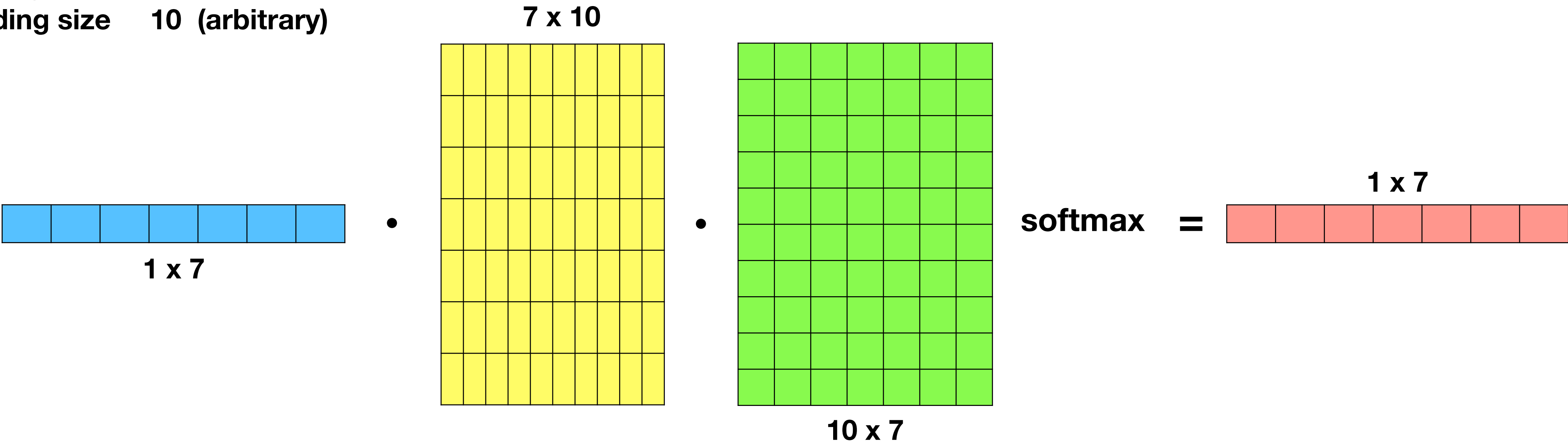
OneHot representation

tensorflow	(1, 0, 0, 0, 0, 0, 0)
keras	(0, 1, 0, 0, 0, 0, 0)
go	(0, 0, 1, 0, 0, 0, 0)
numpy	(0, 0, 0, 1, 0, 0, 0)
xgboost	(0, 0, 0, 0, 1, 0, 0)
pytorch	(0, 0, 0, 0, 0, 1, 0)
pandas	(0, 0, 0, 0, 0, 0, 1)

Training pair: (tensorflow, keras)

Input	Output
(1, 0, 0, 0, 0, 0, 0)	(0, 1, 0, 0, 0, 0, 0)

Vocabulary size 7  
Embedding size 10 (arbitrary)





# Doc2vec: Model architecture

Vocabulary

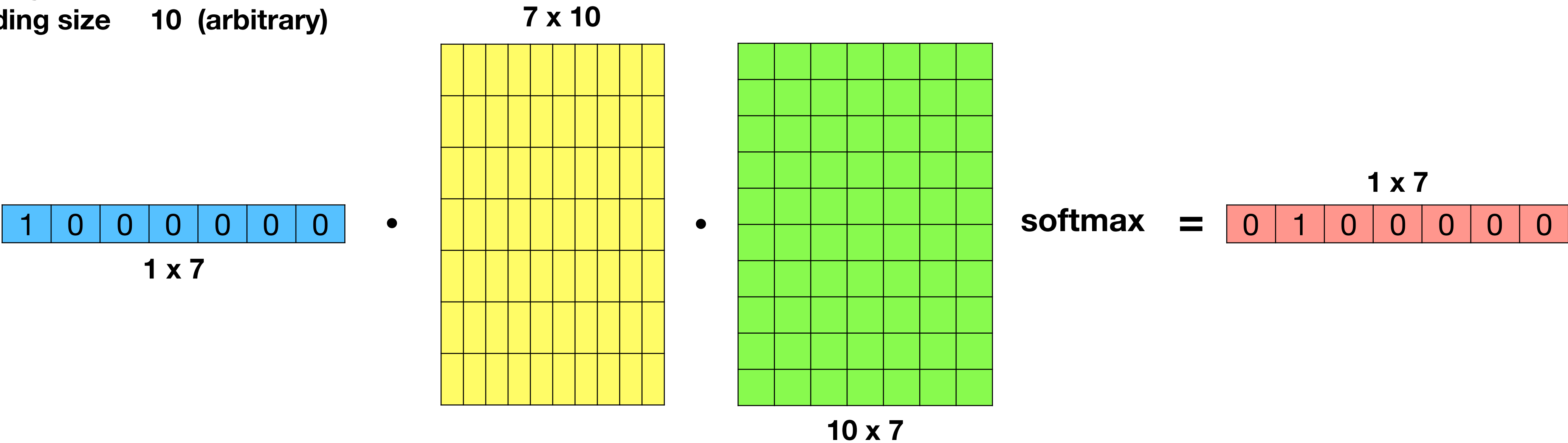
OneHot representation

tensorflow	(1, 0, 0, 0, 0, 0, 0)
keras	(0, 1, 0, 0, 0, 0, 0)
go	(0, 0, 1, 0, 0, 0, 0)
numpy	(0, 0, 0, 1, 0, 0, 0)
xgboost	(0, 0, 0, 0, 1, 0, 0)
pytorch	(0, 0, 0, 0, 0, 1, 0)
pandas	(0, 0, 0, 0, 0, 0, 1)

Training pair: (tensorflow, keras)

Input	Output
(1, 0, 0, 0, 0, 0, 0)	(0, 1, 0, 0, 0, 0, 0)

Vocabulary size 7  
Embedding size 10 (arbitrary)



# Doc2vec: Model architecture

Vocabulary	OneHot representation
tensorflow	(1, 0, 0, 0, 0, 0, 0)
keras	(0, 1, 0, 0, 0, 0, 0)
go	(0, 0, 1, 0, 0, 0, 0)
numpy	(0, 0, 0, 1, 0, 0, 0)
xgboost	(0, 0, 0, 0, 1, 0, 0)
pytorch	(0, 0, 0, 0, 0, 1, 0)
pandas	(0, 0, 0, 0, 0, 0, 1)

## Training pair: (tensorflow, keras)

# Input

## Output

$$(1, 0, 0, 0, 0, 0, 0) \quad (0, 1, 0, 0, 0, 0, 0)$$

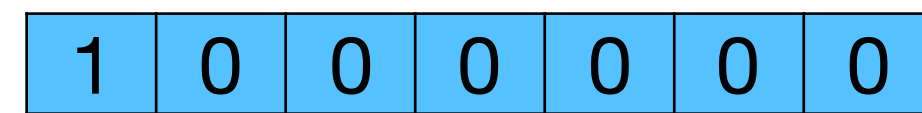
**Vocabulary size**      7

**Embedding size**    10 (arbitrary)

## Representation of tensorflow

$W_1$	$W_2$	$W_3$	$W_4$	$W_5$	$W_6$	$W_7$	$W_8$	$W_9$	$W_{10}$
-------	-------	-------	-------	-------	-------	-------	-------	-------	----------

**(embedding)**



**1 x 7**

**7 x 10**

10 x 7

**softmax** = 

0	1	0	0	0	0	0
---	---	---	---	---	---	---

1 x 7



## Workflow



Generate user view  
activity stream

- **user\_1**: “repo\_1”, “repo\_72”, “repo\_28”, “repo\_1”, “repo\_3”, “repo\_36”,...
- **user\_2**: “repo\_21”, “repo\_41”, “repo\_1”, “repo\_33”, “repo\_1”, “repo\_10”,...
- ...
- **user\_n**: “repo\_40”, “repo\_72”, “repo\_8”, “repo\_96”, “repo\_31”, “repo\_33”,...
- ...

## Workflow



Generate user view  
activity stream



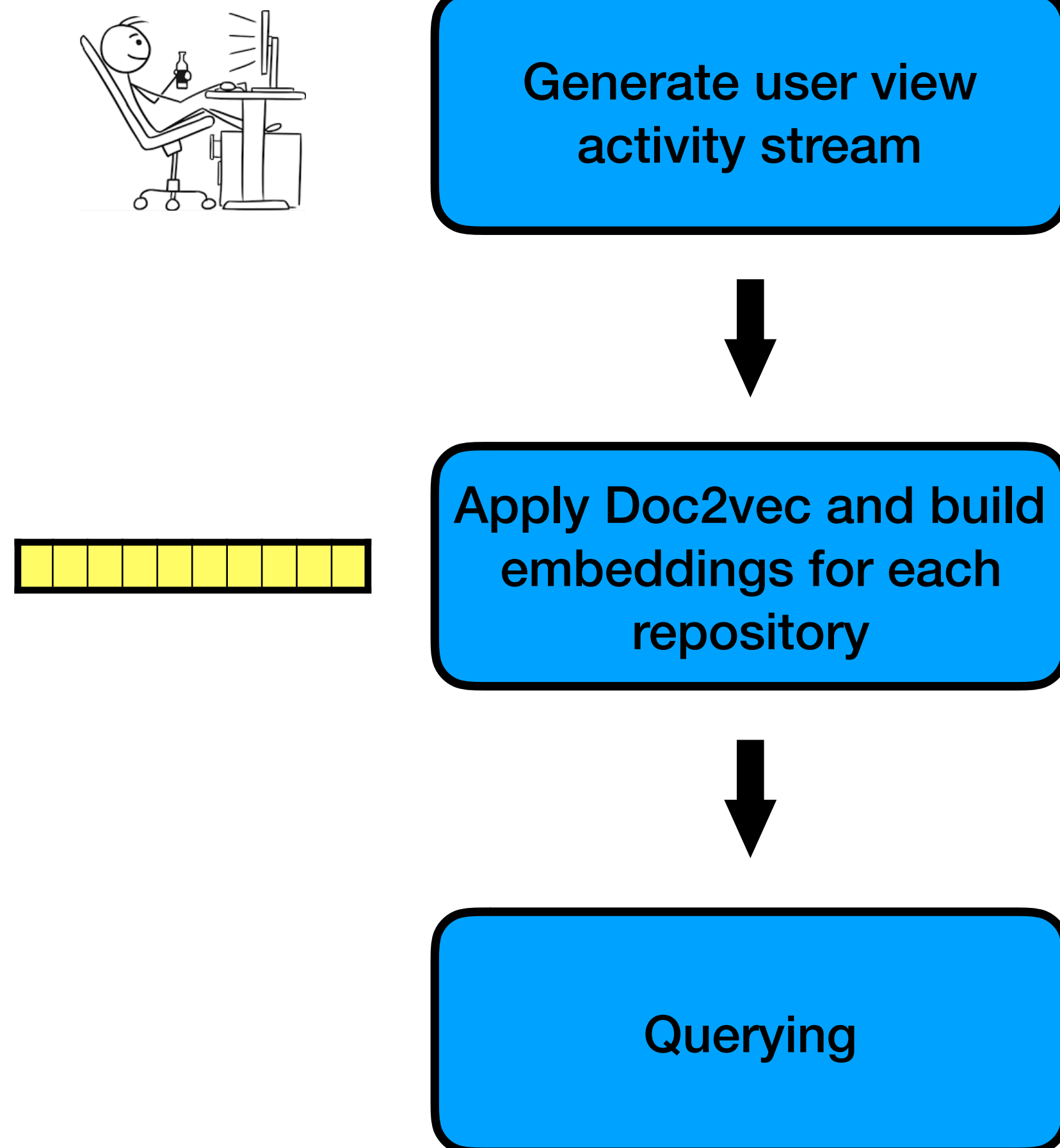
Apply Doc2vec and build  
embeddings for each  
repository



- **user\_1**: “repo\_1”, “repo\_72”, “repo\_28”, “repo\_1”, “repo\_3”, “repo\_36”,...
- **user\_2**: “repo\_21”, “repo\_41”, “repo\_1”, “repo\_33”, “repo\_1”, “repo\_10”,...
- ...
- **user\_n**: “repo\_40”, “repo\_72”, “repo\_8”, “repo\_96”, “repo\_31”, “repo\_33”,...
- ...

- **embedding\_repo\_1**: [ 0.021, 0.003, -0.001, -0.041, -0.103, ..., 0.009]
- **embedding\_repo\_2**: [ -0.011, 0.023, -0.102, -0.019, -0.028, ..., 0.003]
- ...
- **embedding\_repo\_n**: [ 0.039, -0.033, -0.007, -0.022, 0.301, ..., -0.015]
- ...

## Workflow



- **user\_1**: “repo\_1”, “repo\_72”, “repo\_28”, “repo\_1”, “repo\_3”, “repo\_36”,...
- **user\_2**: “repo\_21”, “repo\_41”, “repo\_1”, “repo\_33”, “repo\_1”, “repo\_10”,...
- ...
- **user\_n**: “repo\_40”, “repo\_72”, “repo\_8”, “repo\_96”, “repo\_31”, “repo\_33”,...
- ...

- **embedding\_repo\_1**: [ 0.021, 0.003, -0.001, -0.041, -0.103, ..., 0.009]
- **embedding\_repo\_2**: [ -0.011, 0.023, -0.102, -0.019, -0.028, ..., 0.003]
- ...
- **embedding\_repo\_n**: [ 0.039, -0.033, -0.007, -0.022, 0.301, ..., -0.015]
- ...

$\cos(\mathbf{e}_i, \mathbf{e}_j) \approx 1 \quad \longrightarrow \quad \text{Repos } i \text{ and } j \text{ are similar}$

$\cos(\mathbf{e}_i, \mathbf{e}_j) \approx 0 \quad \longrightarrow \quad \text{Repos } i \text{ and } j \text{ are not similar}$

Seed repository	Similar repositories	cosine distance
JetBrains/kotlin	JetBrains/kotlin-web-site	0.80
	Kotlin/KEEP	0.79
	JetBrains/kotlin-native	0.79
rails/rails	theSteveMitchell/after_party	0.82
	fxn/zeitwerk	0.81
	voltrb/volt	0.77
tensorflow/tensorflow	evdcush/TensorFlow-wheels	0.83
	tensorflow/autograph	0.78
	snipsco/tensorflow-build	0.76
outman123/tensorflow	ynulonger/ijcnn	0.92
	xingjianzhang1997/CatVsDog	0.92
	liush5/tensorflow-example	0.91

Seed repository	Similar repositories	cosine distance
JetBrains/kotlin	JetBrains/kotlin-web-site	0.80
	Kotlin/KEEP	0.79
	JetBrains/kotlin-native	0.79
rails/rails	theSteveMitchell/after_party	0.82
	fxn/zeitwerk	0.81
	voltrb/volt	0.77
tensorflow/tensorflow	evdcush/TensorFlow-wheels	0.83
	tensorflow/autograph	0.78
	snipsco/tensorflow-build	0.76
outman123/tensorflow	ynulonger/ijcnn	0.92
	xingjianzhang1997/CatVsDog	0.92
	liush5/tensorflow-example	0.91





- **Assess the model's ability to find similar repositories**
- **Detect cheating (overestimating similarity)**

- Assess the model's ability to find similar repositories
- Detect cheating (overestimating similarity)

## Building a set of similar repositories: alias strategy

• **user\_1**: “repo\_1”, “repo\_72”, “repo\_7”, “repo\_1”, “repo\_36”,...  
• **user\_2**: “repo\_21”, “repo\_1”, “repo\_67”, “repo\_33”, “repo\_1”,...  
• ...  
• **user\_n**: “repo\_40”, “repo\_72”, “repo\_1”, “repo\_96”, “repo\_31”,...



• **user\_1**: “repo\_1x”, “repo\_72”, “repo\_7”, “repo\_1”, “repo\_36”,...  
• **user\_2**: “repo\_21”, “repo\_1”, “repo\_67”, “repo\_33”, “repo\_1”,...  
• ...  
• **user\_n**: “repo\_40”, “repo\_72”, “repo\_1x”, “repo\_96”, “repo\_31”,...

- Assess the model's ability to find similar repositories
- Detect cheating (overestimating similarity)

## Building a set of similar repositories: alias strategy

• **user\_1**: “repo\_1”, “repo\_72”, “repo\_7”, “repo\_1”, “repo\_36”,...  
• **user\_2**: “repo\_21”, “repo\_1”, “repo\_67”, “repo\_33”, “repo\_1”,...  
• ...  
• **user\_n**: “repo\_40”, “repo\_72”, “repo\_1”, “repo\_96”, “repo\_31”,...



• **user\_1**: “repo\_1x”, “repo\_72”, “repo\_7”, “repo\_1”, “repo\_36”,...  
• **user\_2**: “repo\_21”, “repo\_1”, “repo\_67”, “repo\_33”, “repo\_1”,...  
• ...  
• **user\_n**: “repo\_40”, “repo\_72”, “repo\_1x”, “repo\_96”, “repo\_31”,...

## Addressing non-similar repositories: picking a set of random pairs

- Assess the model's ability to find similar repositories
- Detect cheating (overestimating similarity)

## Building a set of similar repositories: alias strategy

• **user\_1**: “repo\_1”, “repo\_72”, “repo\_7”, “repo\_1”, “repo\_36”,...  
• **user\_2**: “repo\_21”, “repo\_1”, “repo\_67”, “repo\_33”, “repo\_1”,...  
• ...  
• **user\_n**: “repo\_40”, “repo\_72”, “repo\_1”, “repo\_96”, “repo\_31”,...

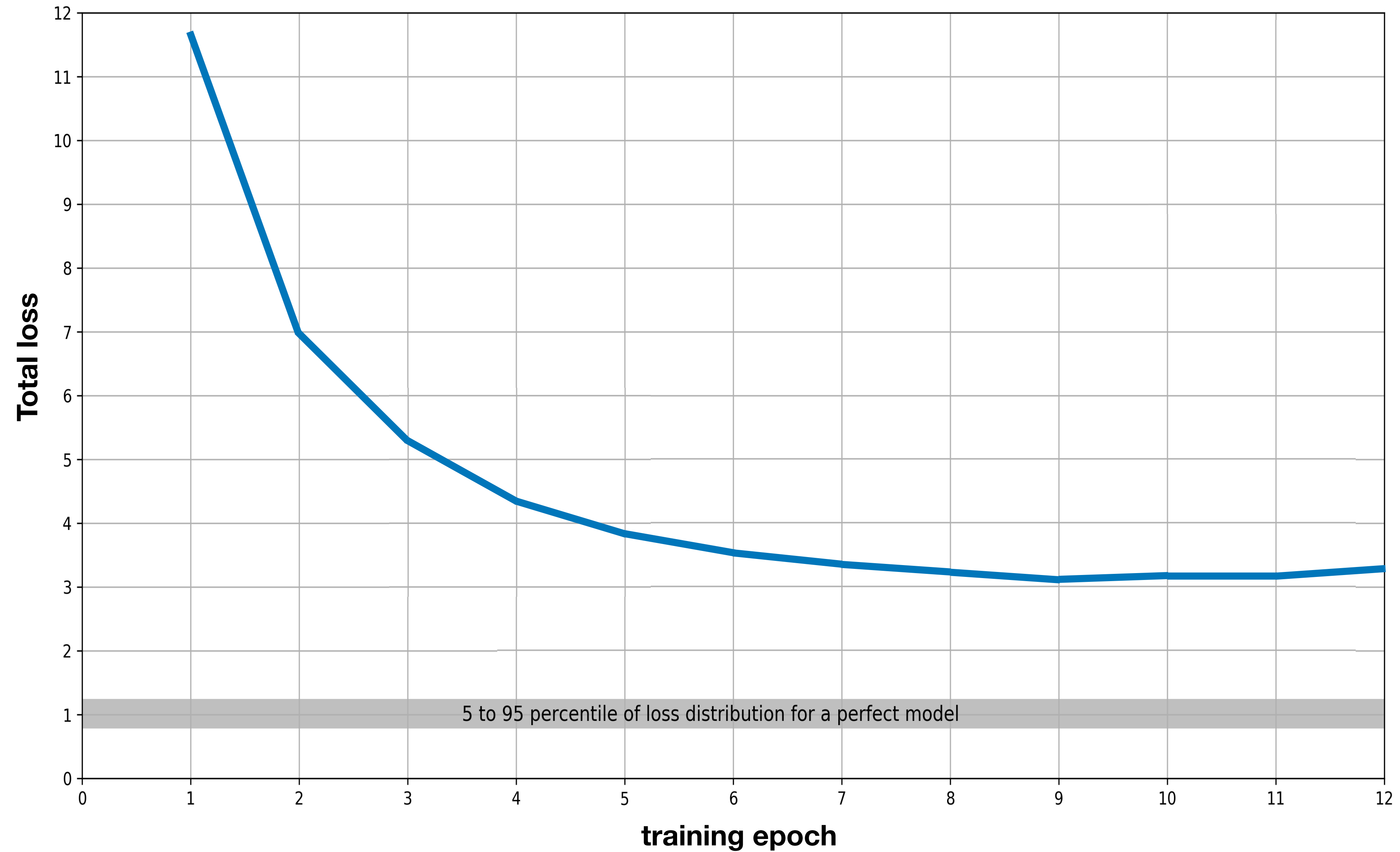


• **user\_1**: “repo\_1x”, “repo\_72”, “repo\_7”, “repo\_1”, “repo\_36”,...  
• **user\_2**: “repo\_21”, “repo\_1”, “repo\_67”, “repo\_33”, “repo\_1”,...  
• ...  
• **user\_n**: “repo\_40”, “repo\_72”, “repo\_1x”, “repo\_96”, “repo\_31”,...

## Addressing non-similar repositories: picking a set of random pairs

**Total Loss = AliasRepoLoss + RandomRepoLoss**

$$\left\{ \begin{array}{l} \text{AliasRepoLoss} = \sum_{i=1}^{N_{alias}} (1 - \cos(\vec{e}_{i_{original}}, \vec{e}_{i_{alias}}))^2 \\ \text{RandomRepoLoss} = \sum_{i=1}^{N_{random}} \cos(\vec{e}_{i_1}, \vec{e}_{i_2})^2 \end{array} \right.$$





- NLP techniques can successfully be used outside their natural field of application
- NLP can be applied across domains, wherever sequences of actions are performed and their underlying entities need to be characterized
- Simple implementations (e.g. Doc2Vec) can be very powerful while easy to implement
- Validation of such applications is not trivial so one needs often to get creative

# Questions?

## Additional resources:

- **Contact:** [romanofoti@github.com](mailto:romanofoti@github.com)
- **Appendix:** see below
- **Resource repository:** [https://github.com/romanofoti/odsc\\_2019](https://github.com/romanofoti/odsc_2019)



# Exploring incremental training

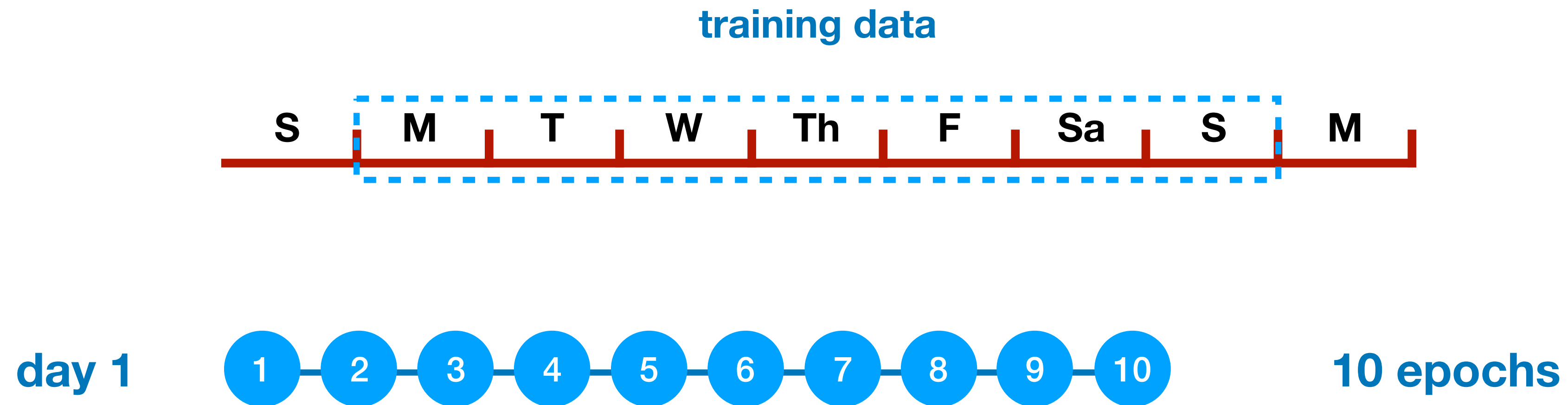
**How about pre-training, saving, then resuming training?**

- Constant need to update
- Time or resources constrains
- Better results

# Exploring incremental training

**How about pre-training, saving, then resuming training?**

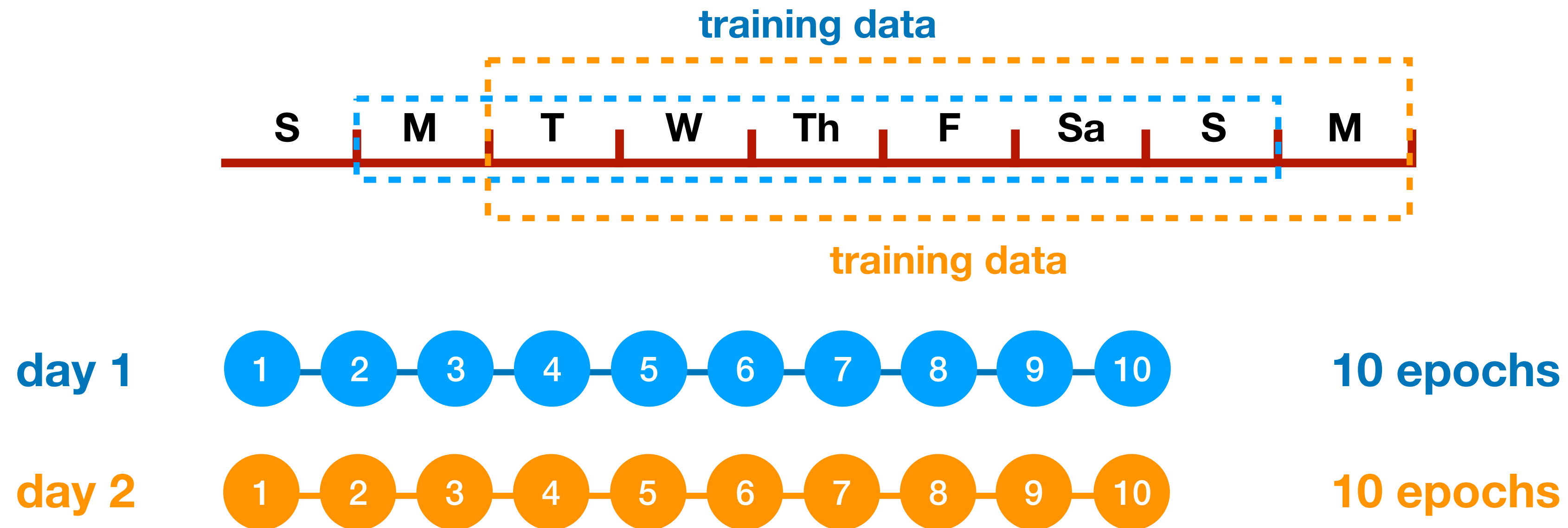
- Constant need to update
- Time or resources constrains
- Better results



# Exploring incremental training

How about pre-training, saving, then resuming training?

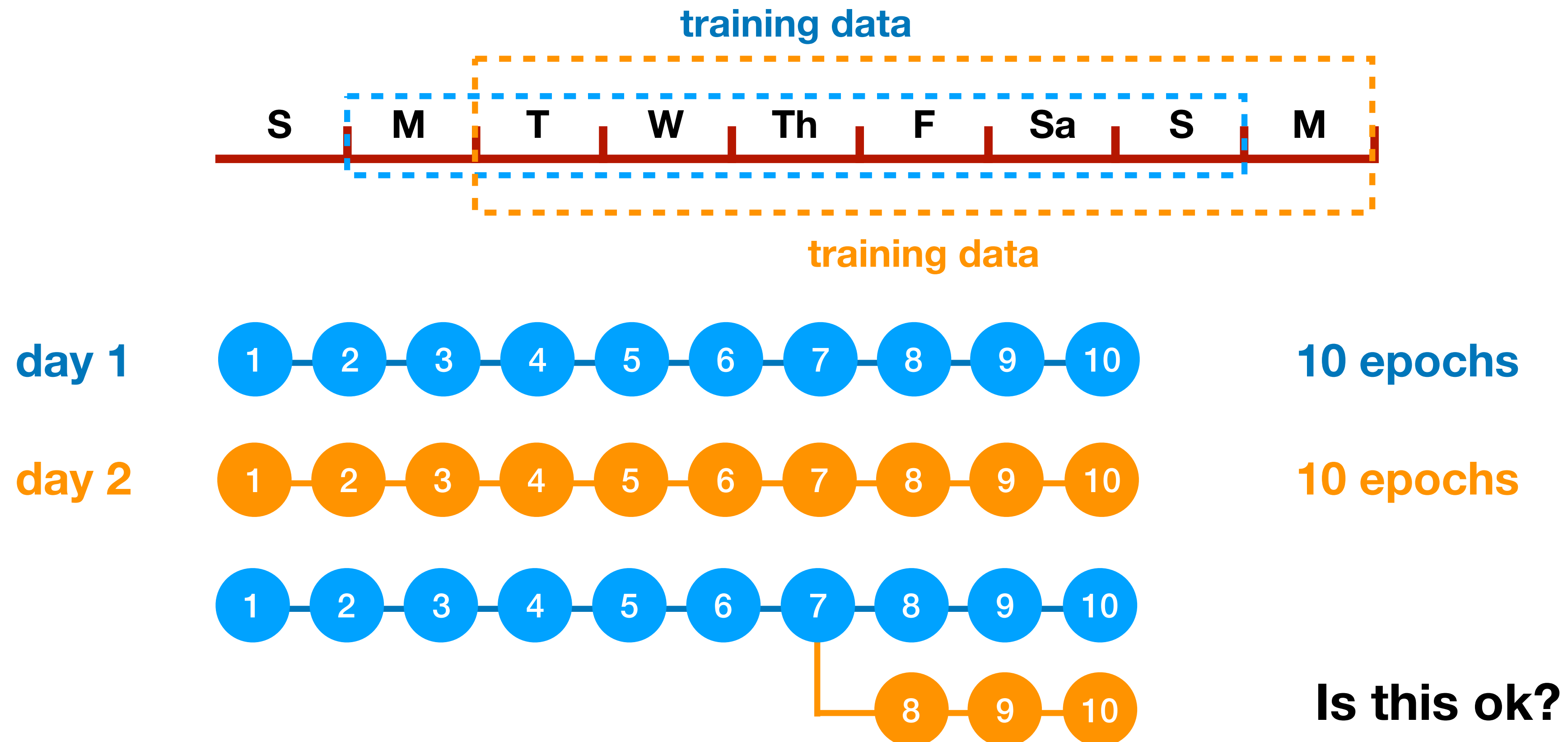
- Constant need to update
- Time or resources constrains
- Better results



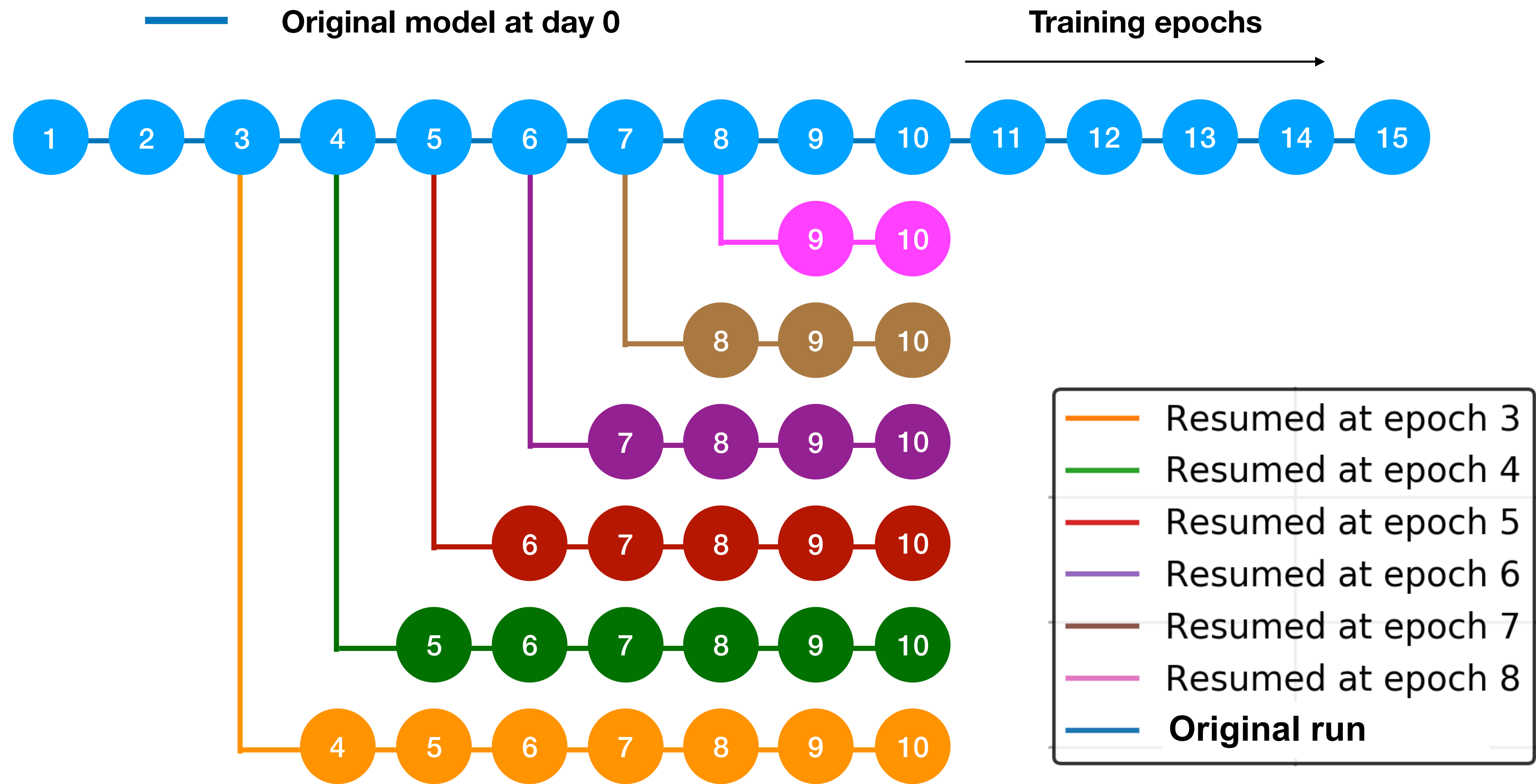
# Exploring incremental training

How about pre-training, saving, then resuming training?

- Constant need to update
- Time or resources constrains
- Better results



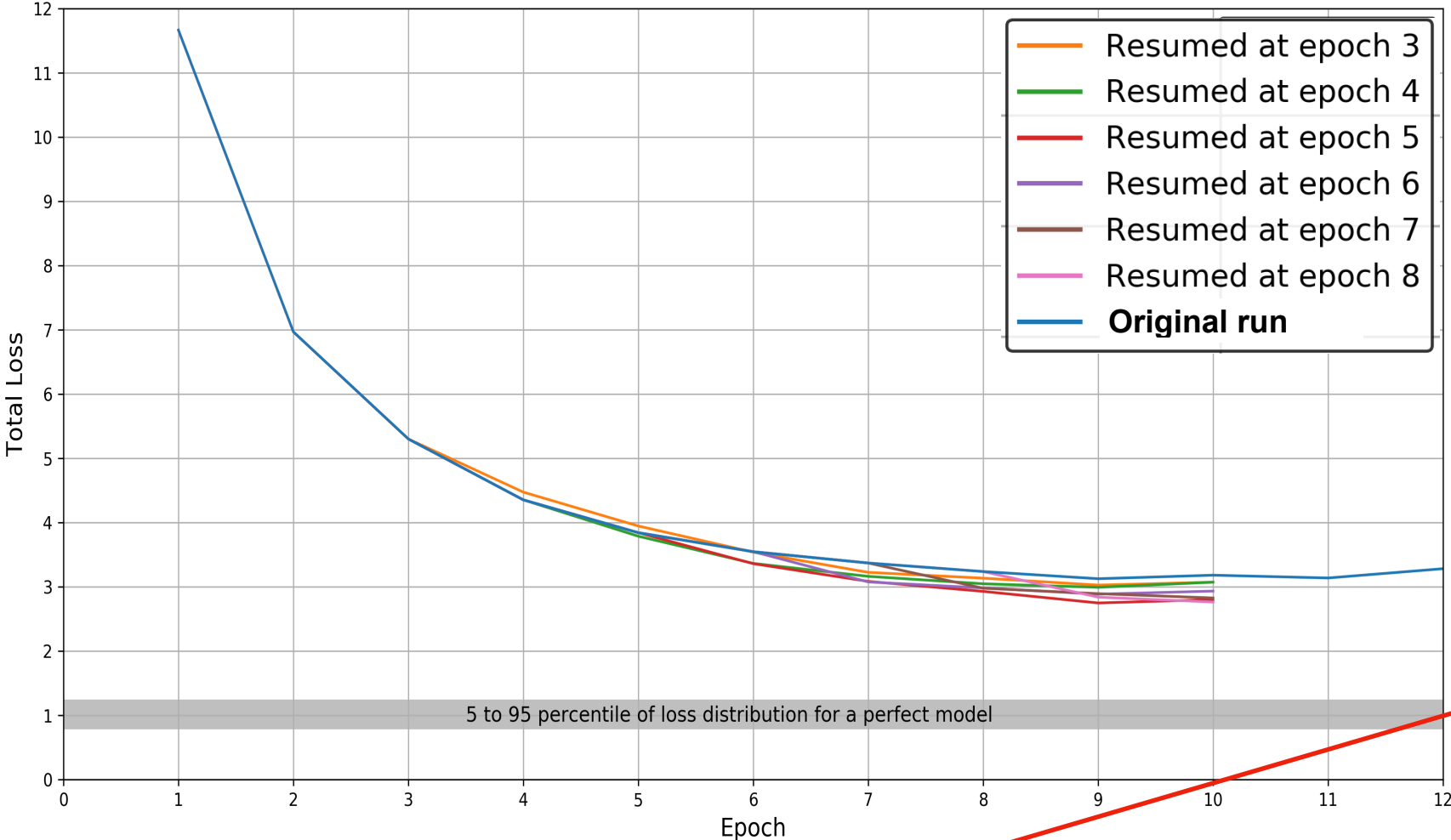
# Exploring incremental training



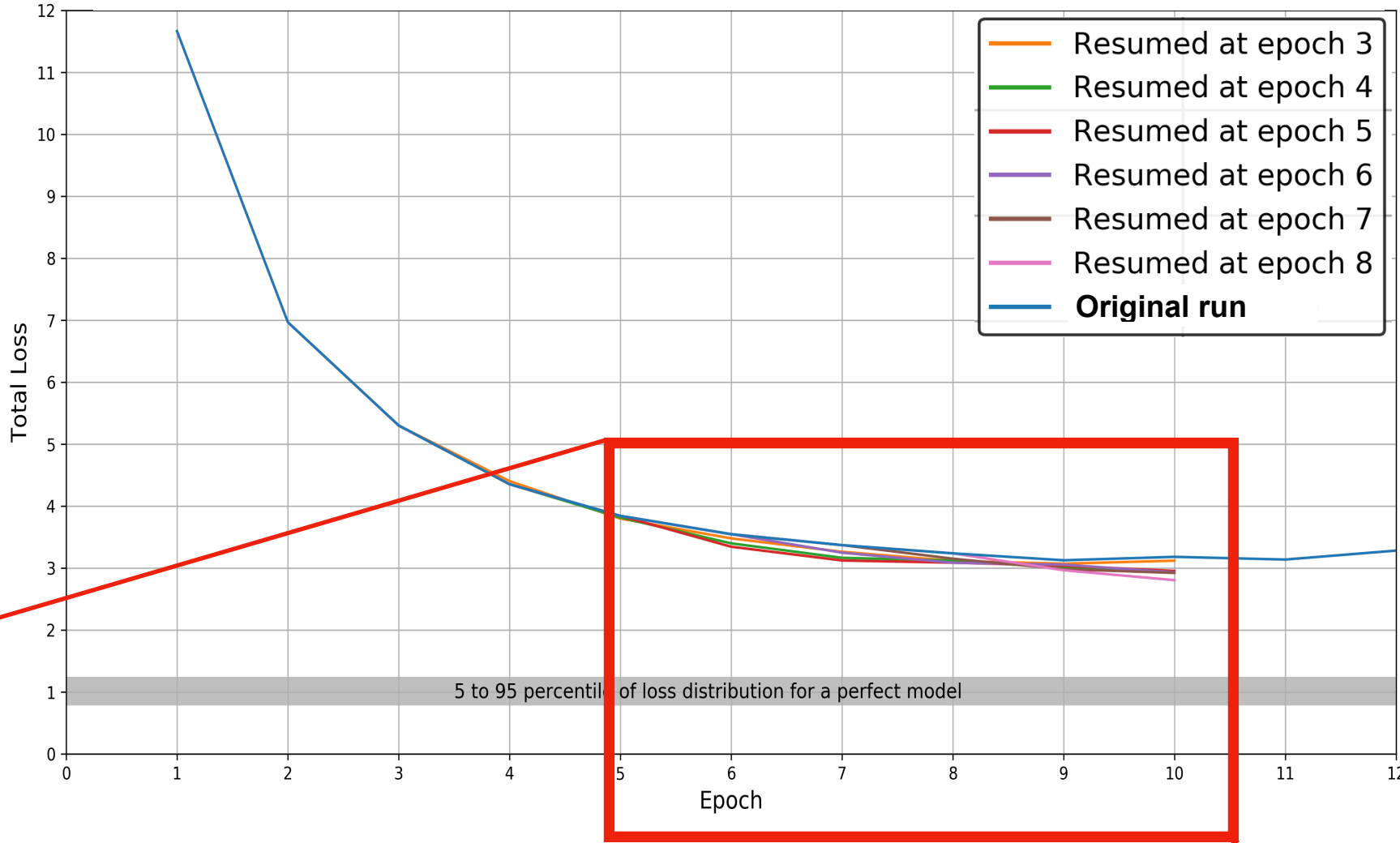
The above branching for training updates is repeated for days 1 through 7

# Results

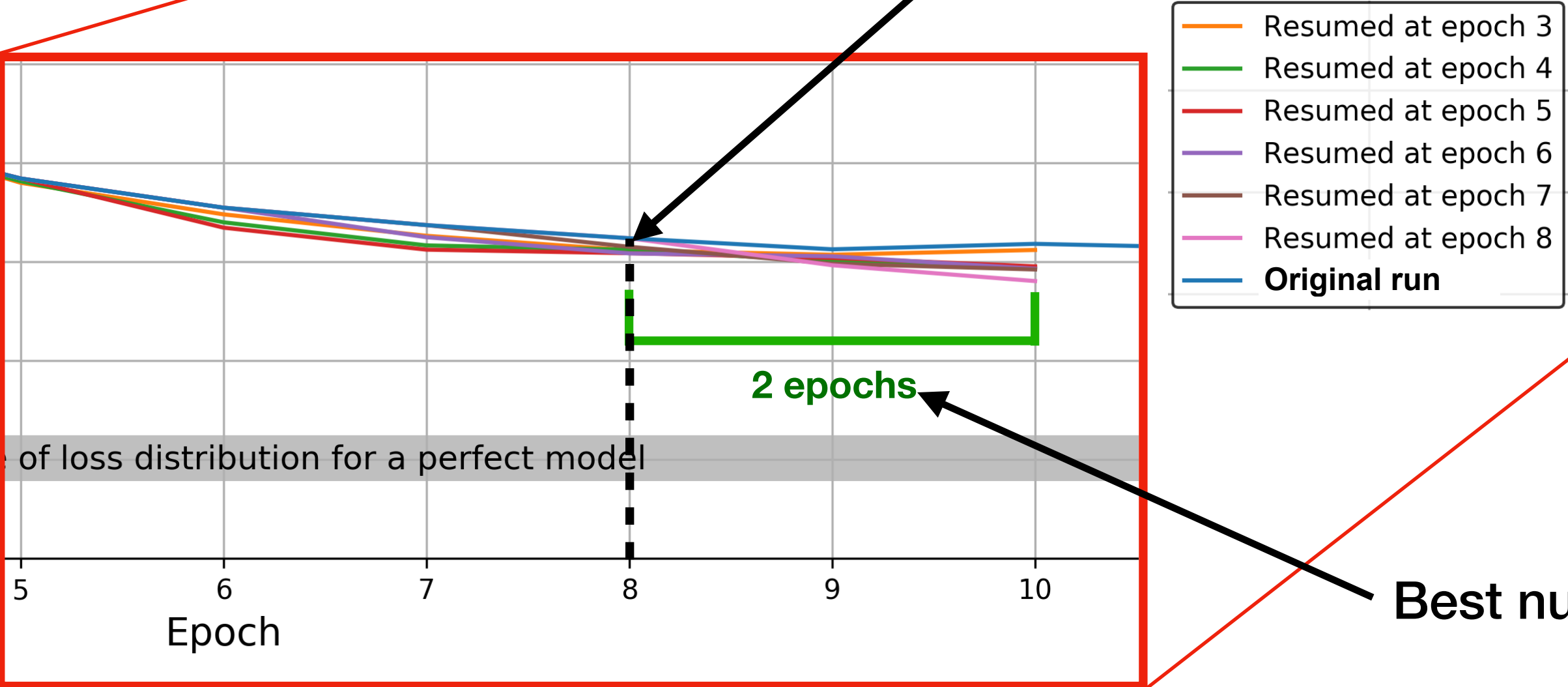
Retraining 1-day stale model with new data



Retraining 7-day stale model with new data



Best resuming point



Best number of updating epochs



# Doc2vec demo

## ▼ Imports

```
In [1]: import pandas as pd
        from gensim.models.doc2vec import Doc2Vec, TaggedDocument
```

## ▼ Loading data

```
In [2]: file_path = '../data/sample_data.csv'
        df = pd.read_csv(file_path, names=['tag', 'stream'])
```

## ▼ Manipulating input dataset

```
In [3]: # Casting integer to string
        df['tag'] = df['tag'].apply(lambda tag: str(tag))
        # Loading input string and casting each element as string
        df['stream'] = df['stream'].apply(lambda stream: [str(el) for el in eval(stream)])
```

```
In [4]: # Visualizing input dataframe
        df.head(3)
```

Out[4]:

	tag	stream
0	677994	[219345042, 172703514, 184153266, 56192185, 52...
1	767275	[96570421, 26516210, 50903853, 26516210, 27729...
2	786423	[187228547, 2791348, 35155700, 2791348, 351557...

## ▼ Preparing training data

```
In [5]: # Tagging each stream
        tagged_streams = [TaggedDocument(stream, [tag])
                           for tag, stream in zip(list(df['tag']), list(df['stream']))]
```

## ▼ Defining the model

```
In [6]: # Initializing the model
        d2v_model = Doc2Vec(min_count=2, window=5, vector_size=20, negative=10)
```

```
In [7]: # Inizializing the vocabulary
        d2v_model.build_vocab(tagged_streams)
```

# Doc2vec demo

## ▼ Training

```
In [8]: d2v_model.train(tagged_streams, total_examples=d2v_model.corpus_count, epochs=10)
```

## ▼ Extracting output

```
In [9]: # Extracting list of items
item_ls = list(d2v_model.wv.vocab.keys())

# Extracting list of tags
tag_ls = d2v_model.docvecs.offset2doctag
```

```
In [10]: # Retrieving embedding for each item
item_vector_ls = [d2v_model[item] for item in item_ls]

# Retrieving embedding for each tag
tag_vector_ls = [d2v_model.docvecs[tag_vect] for tag_vect in tag_ls]
```

```
In [11]: # Printing a sample item and its embedding
print(item_ls[0])
print(item_vector_ls[0])

219345042
[-0.16691333  0.01391616  0.08182272  0.22406493 -0.13018888 -0.08907364
 -0.06540466  0.13903038  0.00235399 -0.0282561  -0.07210001  0.18075605
 -0.06638259  0.04872734 -0.30388135  0.17403638 -0.03945316  0.07126762
  0.03873265 -0.07725172]
```

```
In [12]: # Printing a sample tag and its embedding
print(tag_ls[0])
print(tag_vector_ls[0])

677994
[-0.03200712 -0.06808201  0.09429213  0.20017318 -0.1556576  0.08129843
  0.01421014  0.02463668 -0.09386549  0.03395402  0.13053413  0.16646215
 -0.01707764  0.06892715 -0.35897225  0.20359874 -0.12954955  0.11253071
  0.01763733  0.10113616]
```