Document Reprentations

Natural Language Processing — Lecture 3

Kenneth Enevoldsen | 2024

CENTER FOR HUMANITIES COMPUTING

Before we start

Perspectives in NLP x Humanities, Cognition, and Social Sciences

Workshop at IMC

TIME
Tuesday 1 October 2024, at 09:00 - 17:00
Add to calendar

LOCATION
Jens Chr. Skous Vej 4, 8000 Aarhus C, building 1483, room 344

ORGANIZER
Roberta Rocca, Yuri Bizzoni and Kenneth Enevoldsen

Register

No later than Wednesday 25 September 2024, at 23:59

https://interactingminds.au.dk/events/single-events/artikel/perspectives-in-nlp-x-humanities-cognition-and-social-sciences

Before we start

Kasper Groes 10:32 AM

📢 Do you want to use our Nvidia A100 GPU for free? Hit us up on ddsc.kontakt@gmail.com 📢

Arrow and Nvidia are offering to let us use an A100 GPU for free for the rest of 2024. We need your help to make the most of it! 🚀

So do you want to use the GPU for a project of yours? School projects, personal projects, commercial projects etc are welcome .

Or are you interested in joining a group effort on one of the cool open source projects listed below?

1. Use an LLM to generate a synthetic dataset for training an embedding model
2. Run the scandeval LLM evaluation framework on models not yet included in the benchmark
3. Fine tune LLMs, e.g. Llama 3.2 8b
4. Translate the Fineweb dataset to Danish
5. Train a Danish/Scandinavian version of the Knowledgator gliner models

Express your interest at ddsc.kontakt@gmail.com or send us a brief description of what you want to do with the GPU.

@channel

🙌 3 🇩🇰 2 🇳🇴 1

Sources
& Notes

Invite link to DDSC: https://join.slack.com/t/danskdatascie-o8m9638/shared_invite/zt-1jh2dwmj4-D_mjywfXERvVP75n9O0ykg

If you have an interest in this area do also feel free to contact me.

Agenda

- Document Representations
  - What **attributes do we want?**
  - What do we use document representations for
- How do we construct them
  - **Sparse approaches**: Count-based approached
  - **Reweighting**: TF-IDF
  - **Dense approaches**:
    - Matrix decomposition
    - Aggregation of word embeddings
- Optional: Perspectives from last class

Document representations

Which should be close?

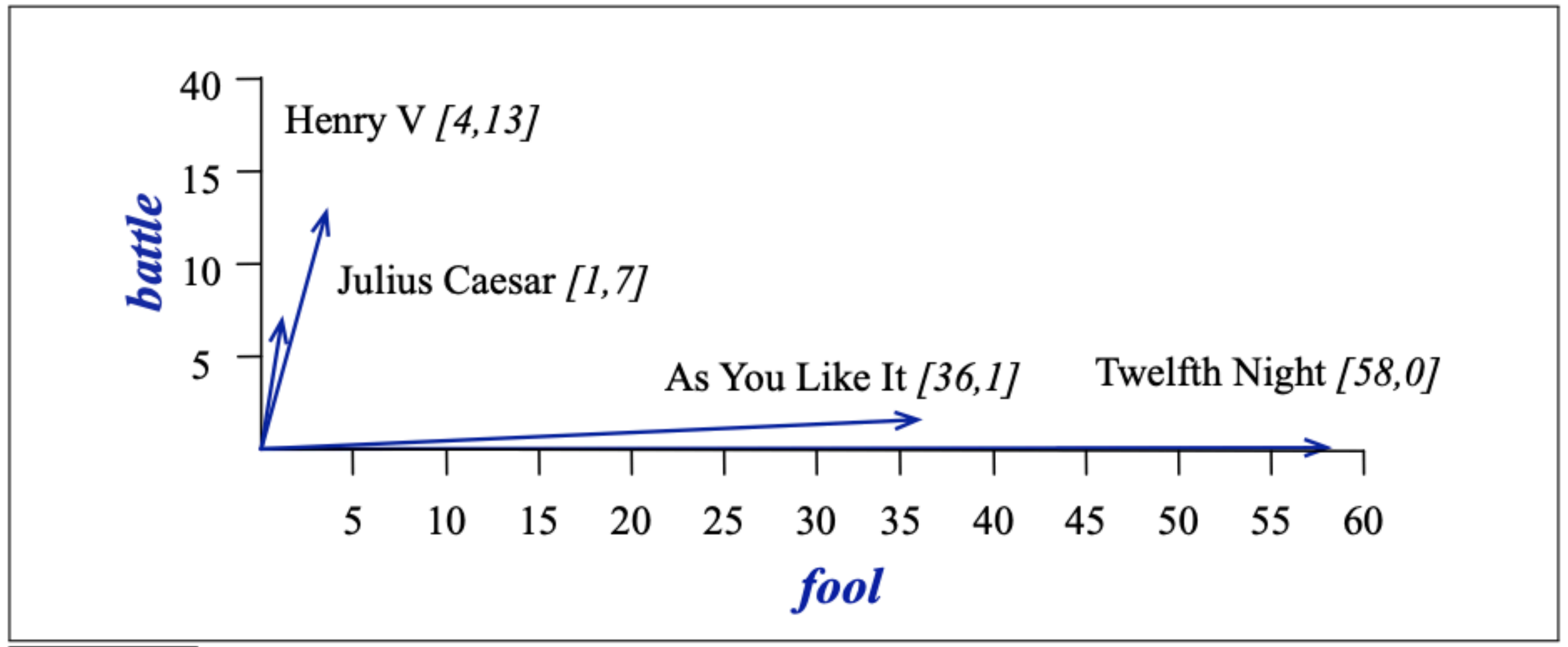
Text (1)	Text (2)	Text (3)
People are shopping.	Numerous customers browsing for produce in a market	People are showering.

Go to:
 https://huggingface.co/spaces/mteb/arena
Then
 Click STS
Try 1-3 examples

Document representations

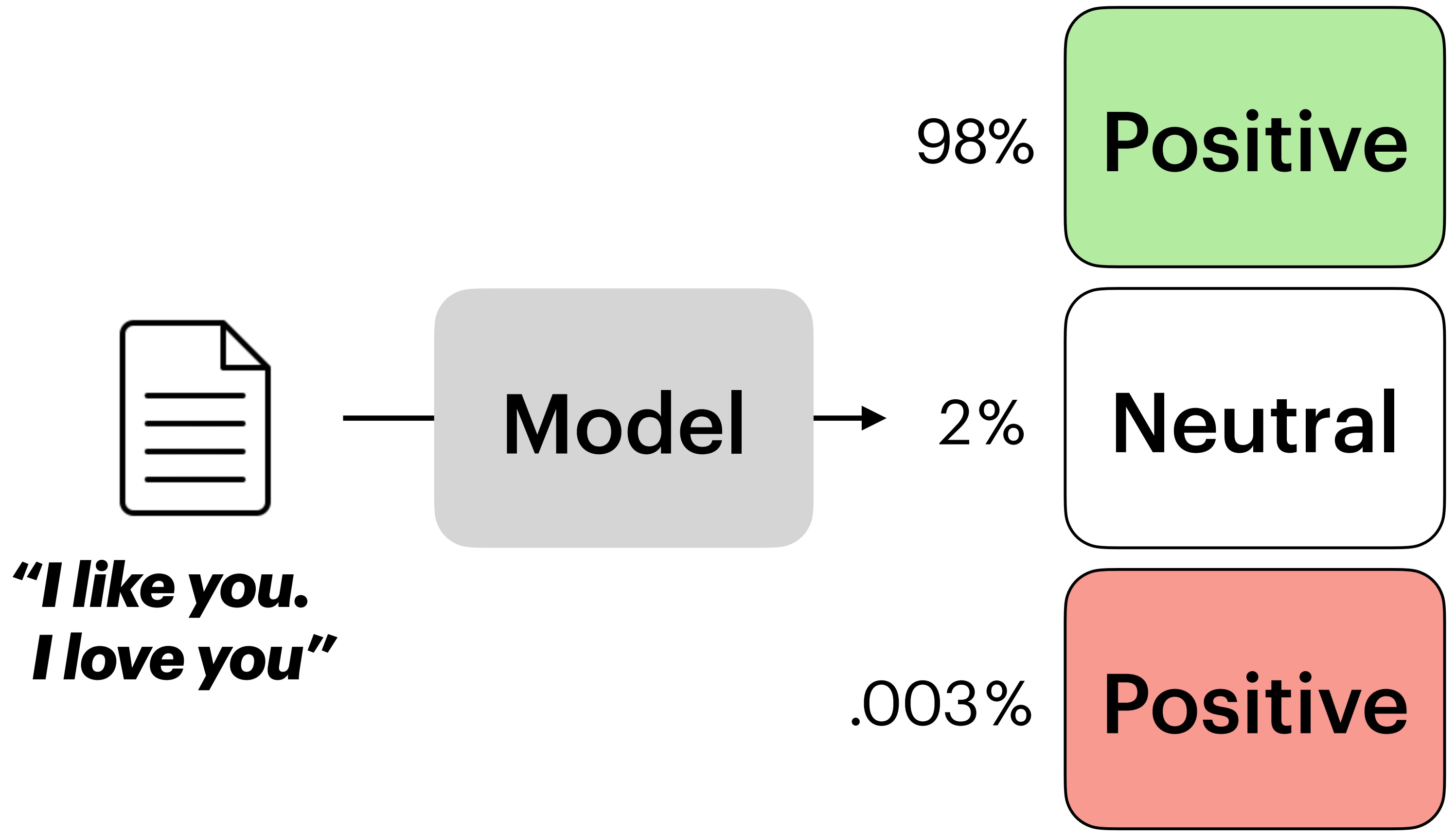
- What attributes do we want?

Text (1)	Text (2)	Text (3)
People are shopping.	Numerous customers browsing for produce in a market	People are showering.



Uses of document representations

- **Classification**



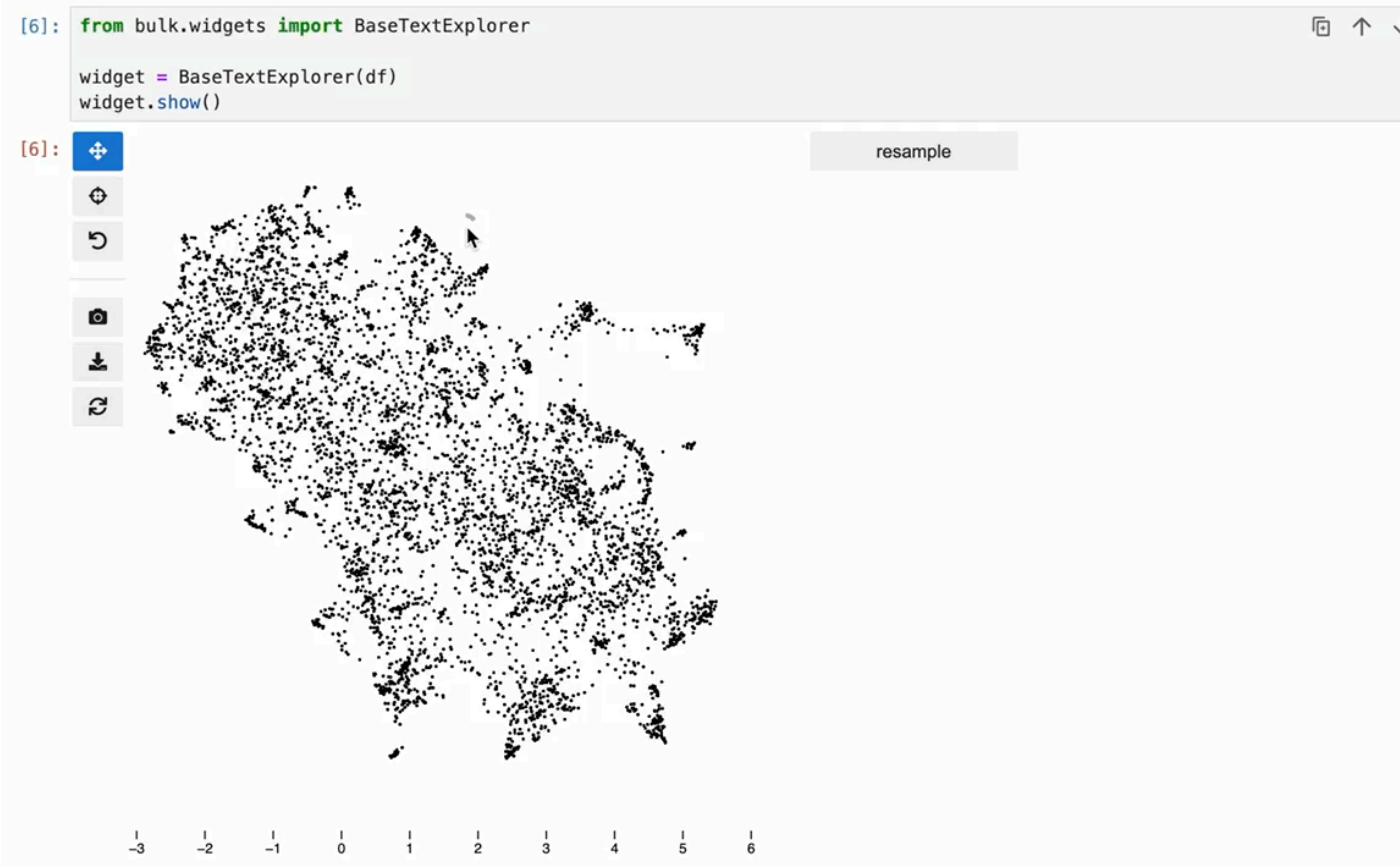

```python
from sklearn.linear_model import LogisticRegression

embeddings = ... # shape (n_docs, embedding)
y = ... # shape (n_docs)
clf = LogisticRegression()
clf.fit(embeddings, y)

new_embeddings = ...
clf.predict(new_embeddings)
```

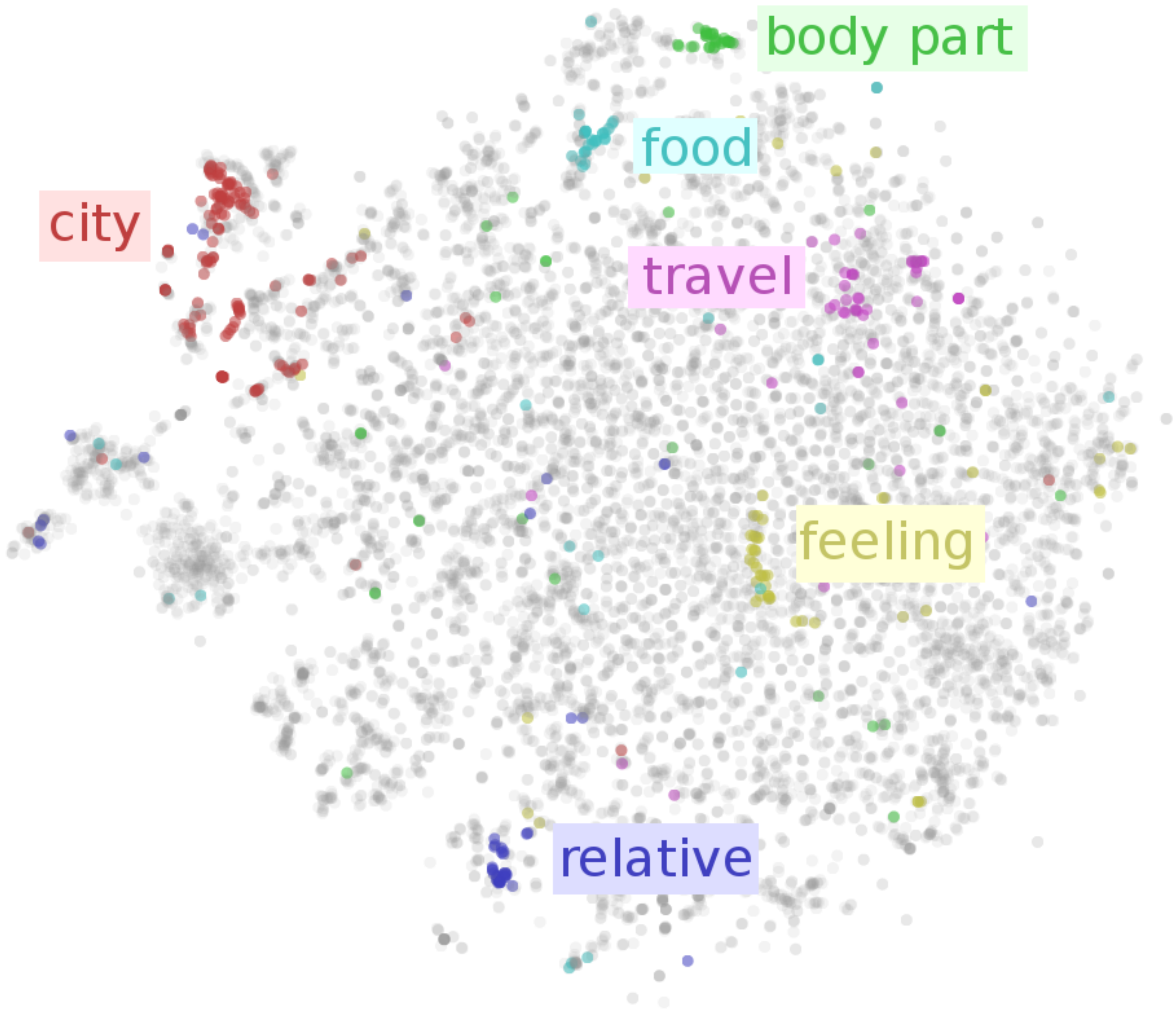
Uses of document representations

- Classification
- Semantic representations
    - **Bulk labelling**



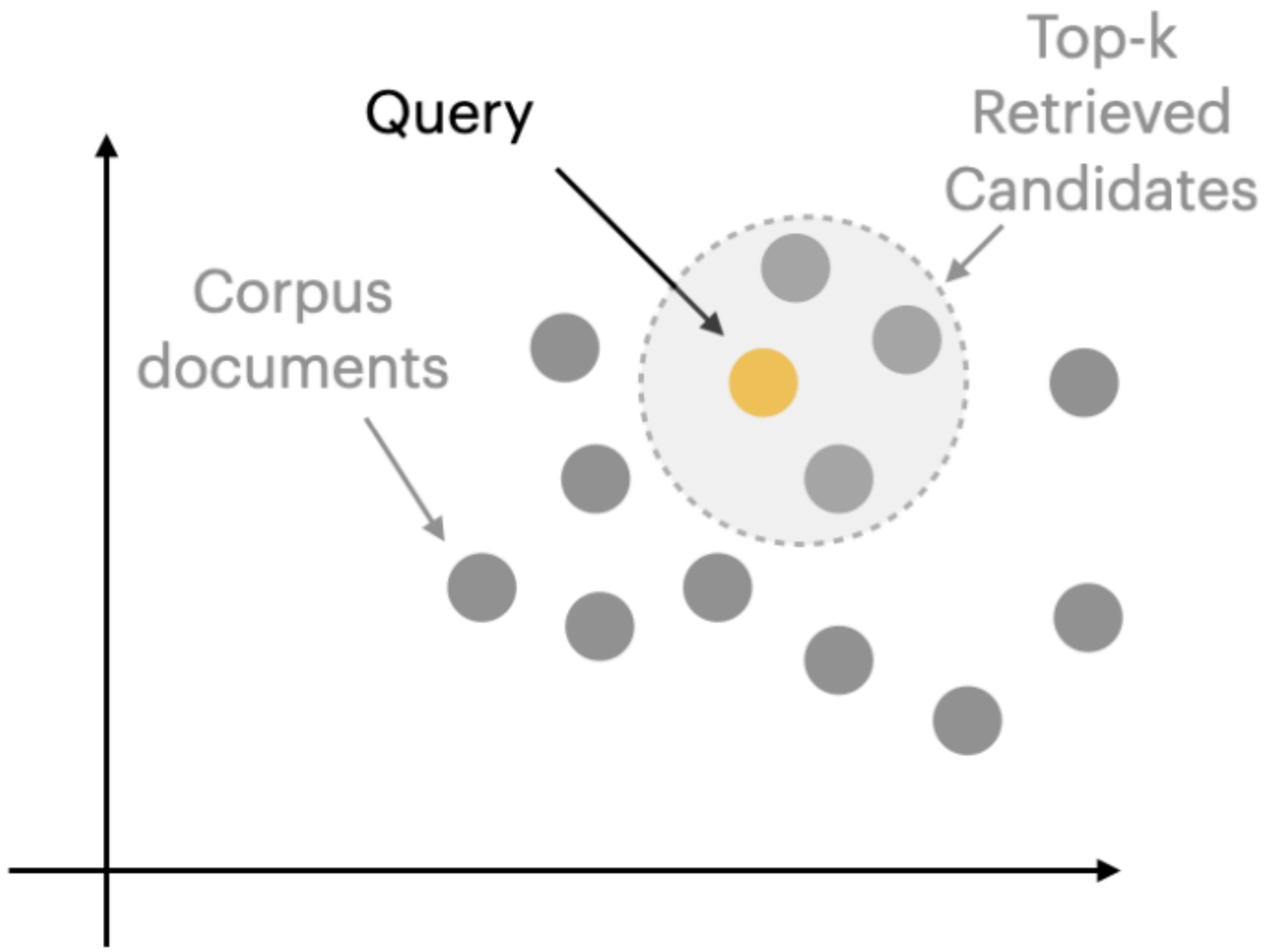
Uses of document representations

- Classification
- Semantic representations
  - Bulk labelling
  - **Thematic clustering** https://huggingface.co/spaces/mteb/arena



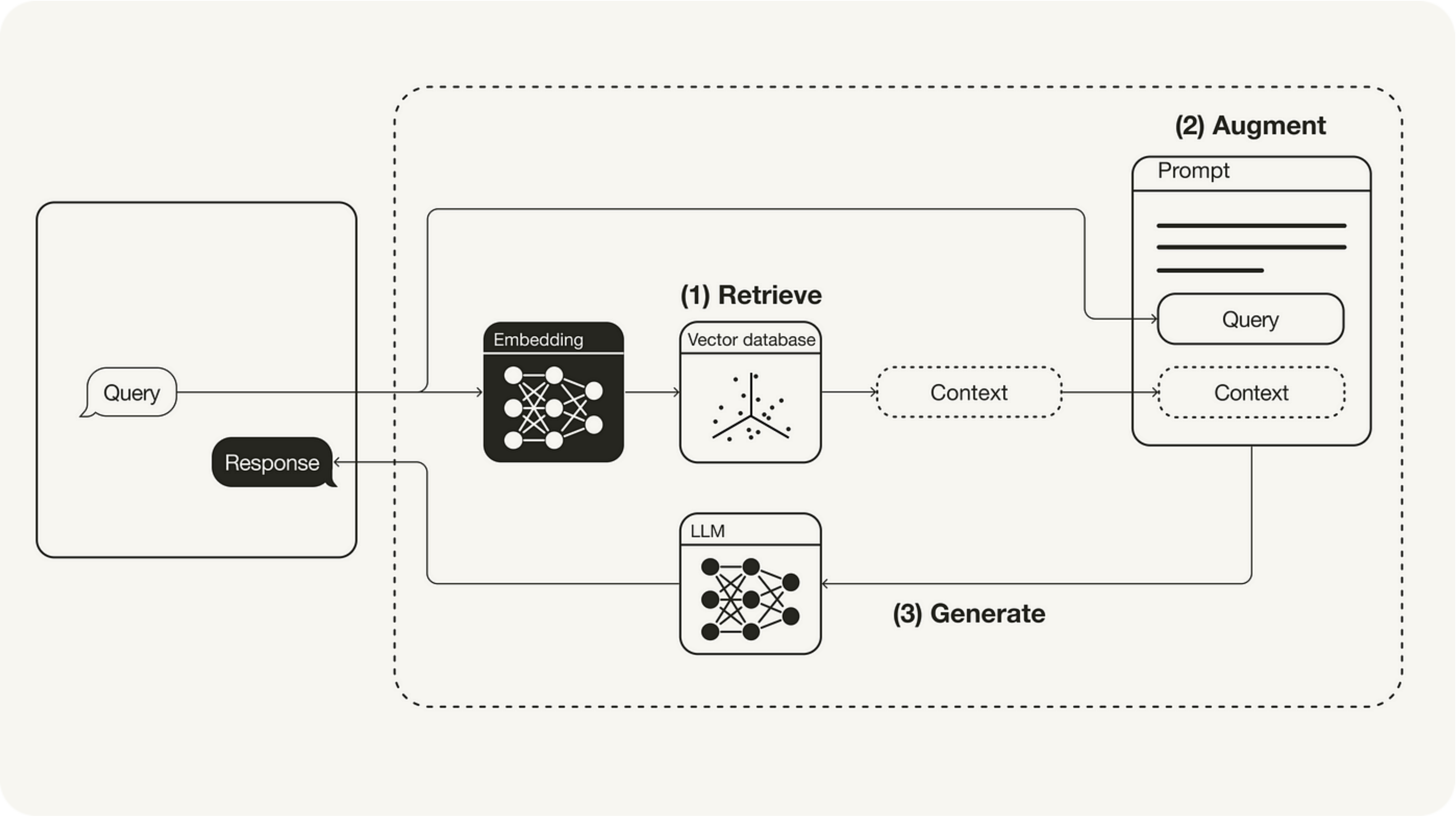
Uses of document representations

- Classification
- Semantic representations
  - Bulk labelling
  - Thematic clustering
- **Retrieval**
  https://huggingface.co/spaces/mteb/arena



Uses of document representations

- Classification
- Semantic representations
  - Bulk labelling
  - Thematic clustering
- Retrieval
  - **Retrieval augmented generation***



* We will get back to this

Uses of document representations

- Classification
- Semantic representations
  - Bulk labelling
  - Thematic clustering
- Retrieval
  - Retrieval augmented generation*
- …
- **What other uses could you imagine?**

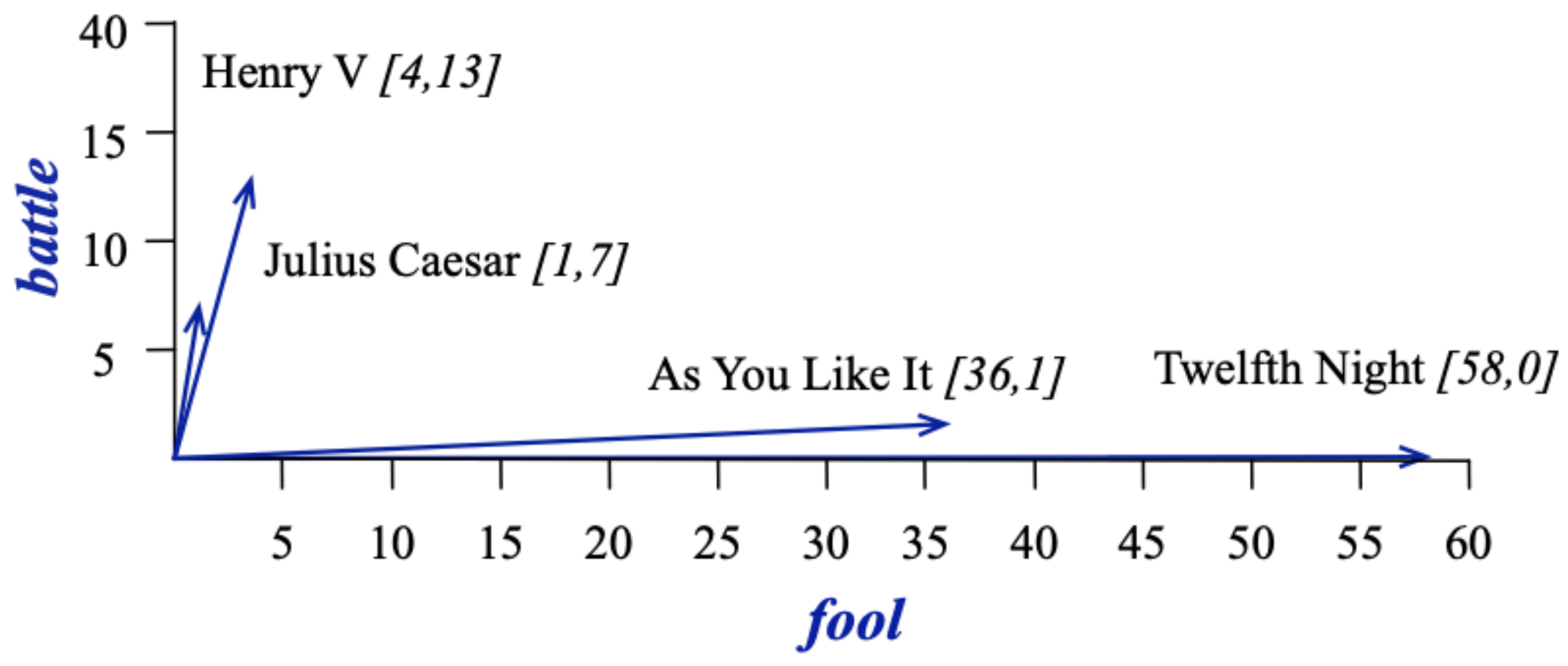
* We will get back to this

Count-based Representations

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

From **word** to **document** vectors

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3



Count-based Representations

- Code example

```python
from sklearn.feature_extraction.text import CountVectorizer
texts = ["This is a text", "This is another text", "This is also a text"]
vectorizer = CountVectorizer()
embeddings = vectorizer.fit_transform(texts)
```

✓ 0.9s

```python
vectorizer.get_feature_names_out()
```

✓ 0.0s

```
array(['also', 'another', 'is', 'text', 'this'], dtype=object)
```

```python
embeddings.toarray()
```

✓ 0.0s

```
array([[0, 0, 1, 1, 1],
       [0, 1, 1, 1, 1],
       [1, 0, 1, 1, 1]])
```

Count-based Representations

A lot of assumptions go here

- Code example
- **What could these assumptions be?**

```python
from sklearn.feature_extraction.text import CountVectorizer
texts = ["This is a text", "This is another text", "This is also a text"]
vectorizer = CountVectorizer()
embeddings = vectorizer.fit_transform(texts)
```

✓ 0.9s

```python
vectorizer.get_feature_names_out()
```

✓ 0.0s

```
array(['also', 'another', 'is', 'text', 'this'], dtype=object)
```

```python
embeddings.toarray()
```

✓ 0.0s

```
array([[0, 0, 1, 1, 1],
       [0, 1, 1, 1, 1],
       [1, 0, 1, 1, 1]])
```

Count-based Representations

- Code example
- What could these assumptions be?
  - **Tokenization**
    - What happened to the t?
    - Couldn't the questionmark be meaningful?
    - Casing?

```
from sklearn.feature_extraction.text import CountVectorizer
```
✓ 8.4s

```
texts = ["isn't this a word?"]
vectorizer = CountVectorizer()
embeddings = vectorizer.fit_transform(texts)
vectorizer.get_feature_names_out()
```
✓ 0.0s

```
array(['isn', 'this', 'word'], dtype=object)
```

```
embeddings = vectorizer.fit_transform(["CASING MATTERS!"])
vectorizer.get_feature_names_out()
```
✓ 0.0s

```
array(['casing', 'matters'], dtype=object)
```

Count-based Representations

- Code example
- What could these assumptions be?
  - Tokenization
  - **N-grams**
    - Why is this important?
    - Why is it problematic?

```python
texts = ["This is a text", "This is another text", "This is also a text"]
vectorizer = CountVectorizer(ngram_range=(1, 2))
embeddings = vectorizer.fit_transform(texts)
vectorizer.get_feature_names_out()
```

✓ 0.0s

```
array(['also', 'also text', 'another', 'another text', 'is', 'is also',
       'is another', 'is text', 'text', 'this', 'this is'], dtype=object)
```

```python
texts = ["This is a text", "This is another text", "This is also a text"]
vectorizer = CountVectorizer(ngram_range=(1, 3))
embeddings = vectorizer.fit_transform(texts)
vectorizer.get_feature_names_out()
```

✓ 0.0s

```
array(['also', 'also text', 'another', 'another text', 'is', 'is also',
       'is also text', 'is another', 'is another text', 'is text', 'text',
       'this', 'this is', 'this is also', 'this is another',
       'this is text'], dtype=object)
```

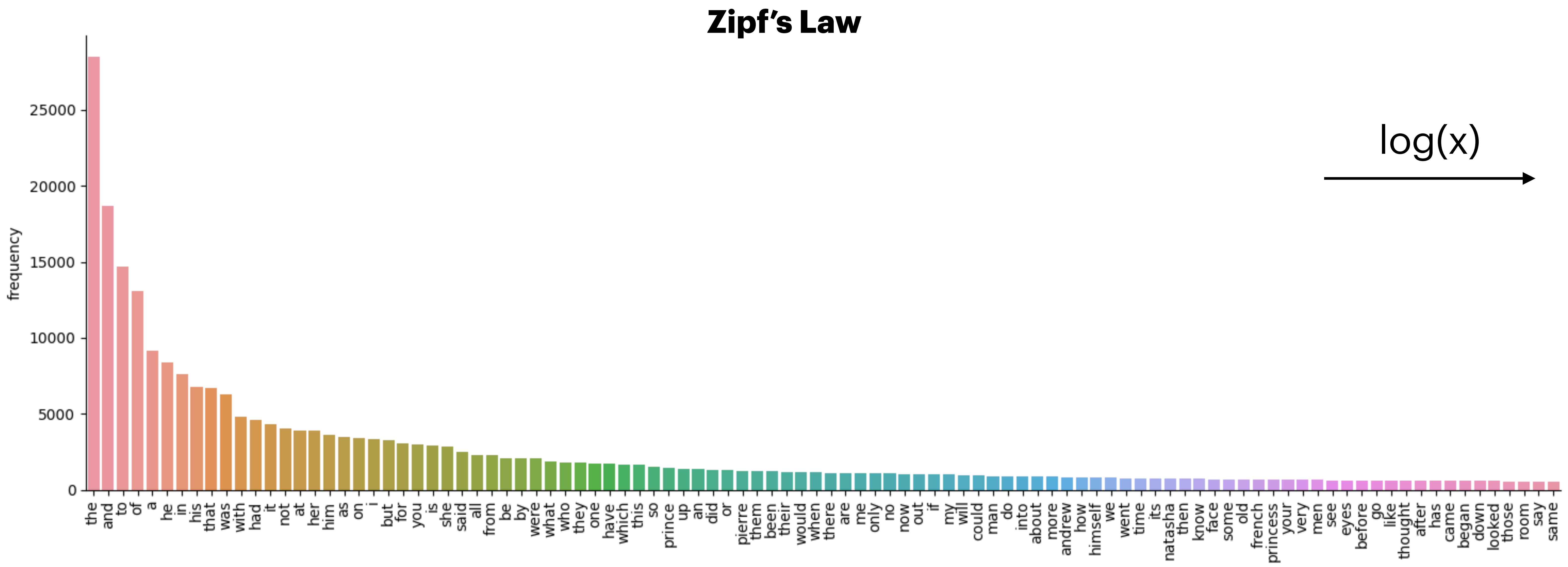
Count-based Representations

- A few tricks:
  - **Binary**
    - Just knowing if a word appear is often enough
  - **Removing stop words**
    - Using a list: *["the", "and", "a", ...]*
    - Word should not appear in >50% of documents
  - Don't overdo it on **vocabulary size**

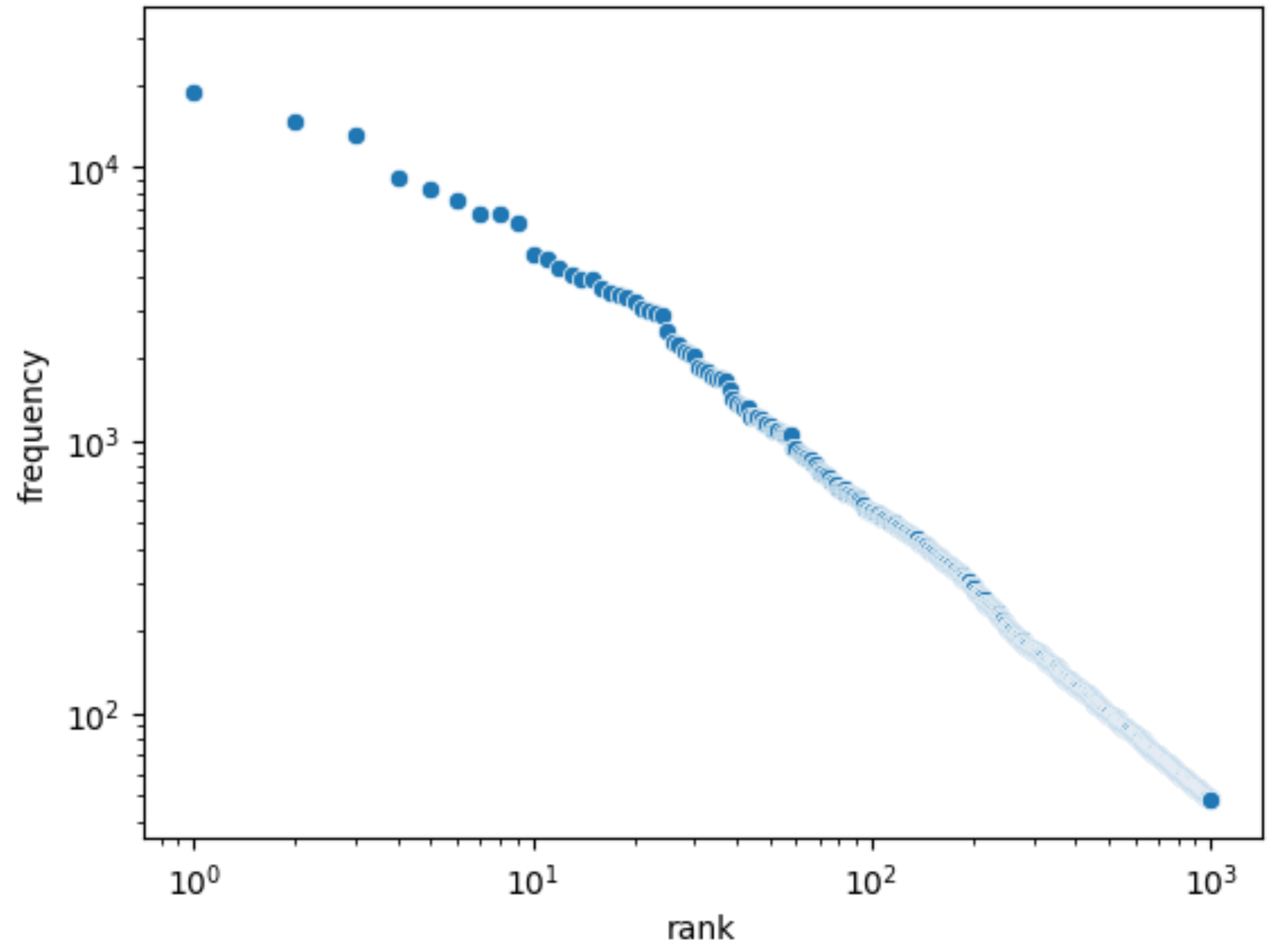
Count-based Representations

- **Problems?**

Count Distributions



log(x)



Any questions?

CENTER FOR
HUMANITIES
COMPUTING

Reweighting: TF-iDF

- We previously saw PPMI
- **T**erm-**F**requency **i**nverse **D**ocument **F**requency
  - **T**erm-**F**requency

$$tf_{t,d} = count(t, d)$$

Term/word

Document

	ML	Food	Sports	Maths
curve	0	1	1	0
sugar	0	20	10	0
sphere	10	0	15	40
foul	0	0	0	50
the	**352500**	**431230**	**239000**	**748901**

Sources & Notes

The wiki page on this is great: https://en.wikipedia.org/wiki/Tf–idf

CENTER FOR HUMANITIES COMPUTING

Reweighting: TF-iDF

- We previously saw PPMI
- **T**erm-**F**requency **i**nverse **D**ocument **F**requency
  - **T**erm-**F**requency

$$tf_{t,d} = log(count(t, d) + 1)$$

normalizes large numbers
Recall Zipf's law

Why?
hint: what happens if count is 0?

	ML	Food	Sports	Maths
curve	0	1	1	0
sugar	0	20	10	0
sphere	10	0	15	40
foul	0	0	0	50
the	**352500**	**431230**	**239000**	**748901**

Sources & Notes

The wiki page on this is great: https://en.wikipedia.org/wiki/Tf–idf

Reweighting: TF-iDF

- We previously saw PPMI
- **T**erm-**F**requency **i**nverse-**D**ocument-**F**requency
  - **T**erm-**F**requency

$$tf_{t,d} = log(count(t,d) + 1)$$

  - **i**nverse-**D**ocument-**F**requency

$$idf_t = log\left(\frac{N}{df_t}\right)$$

Number of documents

Number of document with t

How much information does the word provides (is it common?)

	ML	Food	Sports	Maths
curve	0	1	1	0
sugar	0	20	10	0
sphere	10	0	15	40
foul	0	0	0	50
the	**352500**	**431230**	**239000**	**748901**

The wiki page on this is great: https://en.wikipedia.org/wiki/Tf–idf

Reweighting: TF-iDF

- We previously saw PPMI
- **T**erm-**F**requency **i**nverse-**D**ocument-**F**requency
  - **T**erm-**F**requency

    $$tf_{t,d} = log(count(t,d) + 1)$$

  - **i**nverse-**D**ocument-**F**requency

    $$idf_t = log\left(\frac{N}{df_t}\right)$$

  - **TF-iDF**

    $$tf_{i,d} \cdot idf_t$$

	ML	Food	Sports	Maths
curve	0	1	1	0
sugar	0	20	10	0
sphere	10	0	15	40
foul	0	0	0	50
the	**352500**	**431230**	**239000**	**748901**

Sources & Notes

The wiki page on this is great: https://en.wikipedia.org/wiki/Tf–idf

TF-iDF Variations

- Multiple formulations exist

Raw counts $$tf_{t,d} = count(t,d)$$

Normalized $$tf_{t,d} = log(count(t,d) + 1)$$

Binary $$tf_{t,d} = \begin{cases} 1, & \text{if } count(t,d) > 0 \\ 0, & \text{otherwise} \end{cases}$$

...

Log normalized $$idf_t = log\left(\frac{N}{df_t}\right)$$

+ Smoothing $$idf_t = log\left(\frac{N}{df_t + 1}\right) + 1$$

...

The wiki page on this is great: https://en.wikipedia.org/wiki/Tf–idf

Reweighting: TF-iDF

```python
from sklearn.feature_extraction.text import TfidfVectorizer
texts = ["This is a text", "This is another text", "This is also a text"]
vectorizer = TfidfVectorizer()
embeddings = vectorizer.fit_transform(texts)
```
✓ 0.0s

Where are the assumptions?

```python
vectorizer.get_feature_names_out()
```
✓ 0.0s

```
array(['also', 'another', 'is', 'text', 'this'], dtype=object)
```

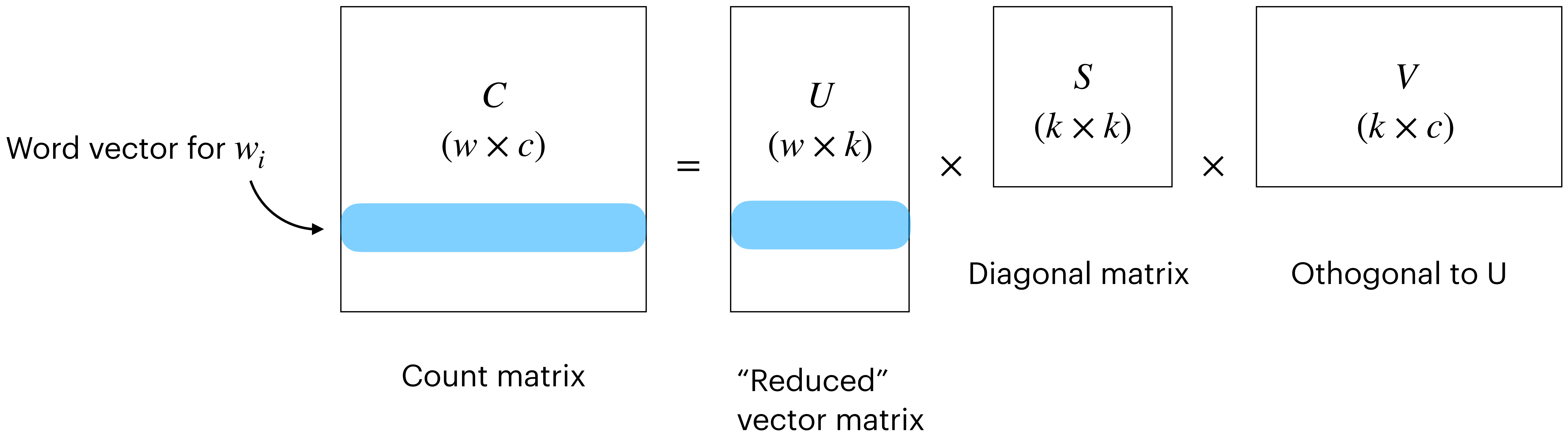
```python
embeddings.toarray()
```
✓ 0.0s

```
array([[0.        , 0.        , 0.57735027, 0.57735027, 0.57735027],
       [0.        , 0.69903033, 0.41285857, 0.41285857, 0.41285857],
       [0.69903033, 0.        , 0.41285857, 0.41285857, 0.41285857]])
```

Any questions?

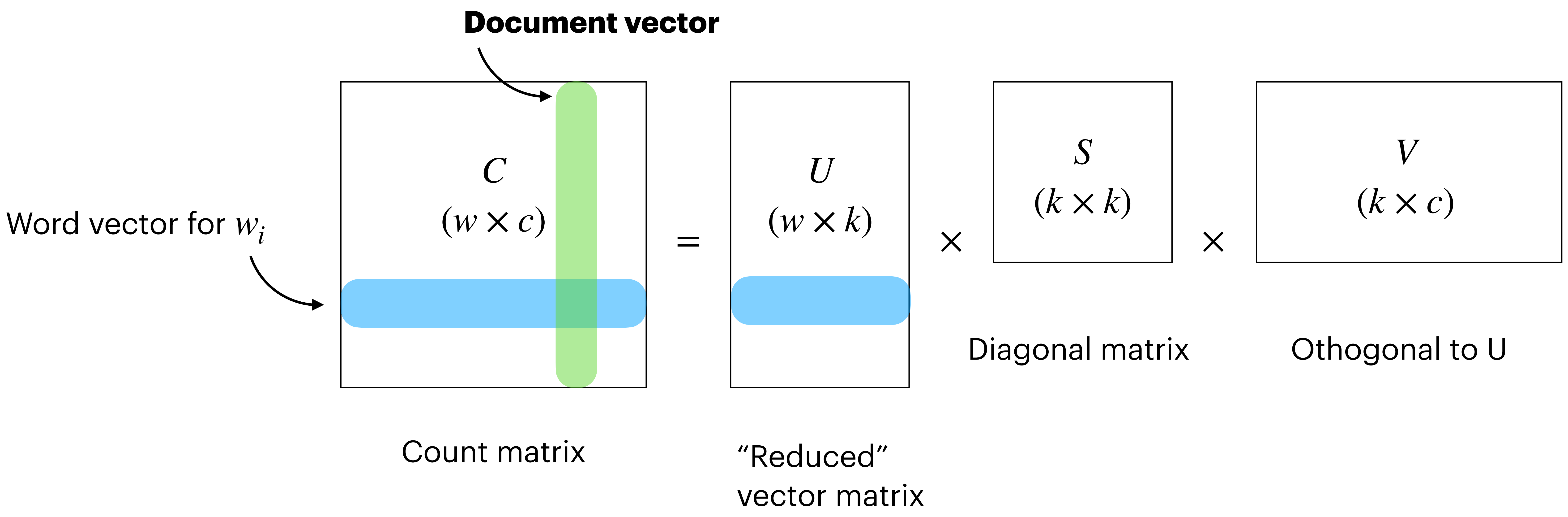
Dense Document representations

- Matrix decomposition
- Recall **S**ingular **V**alue **D**ecomposition



Dense Document representations

- Matrix decomposition
- Recall **S**ingular **V**alue **D**ecomposition



Dense Document representations

- Matrix decomposition
- Recall **S**ingular **V**alue **D**ecomposition

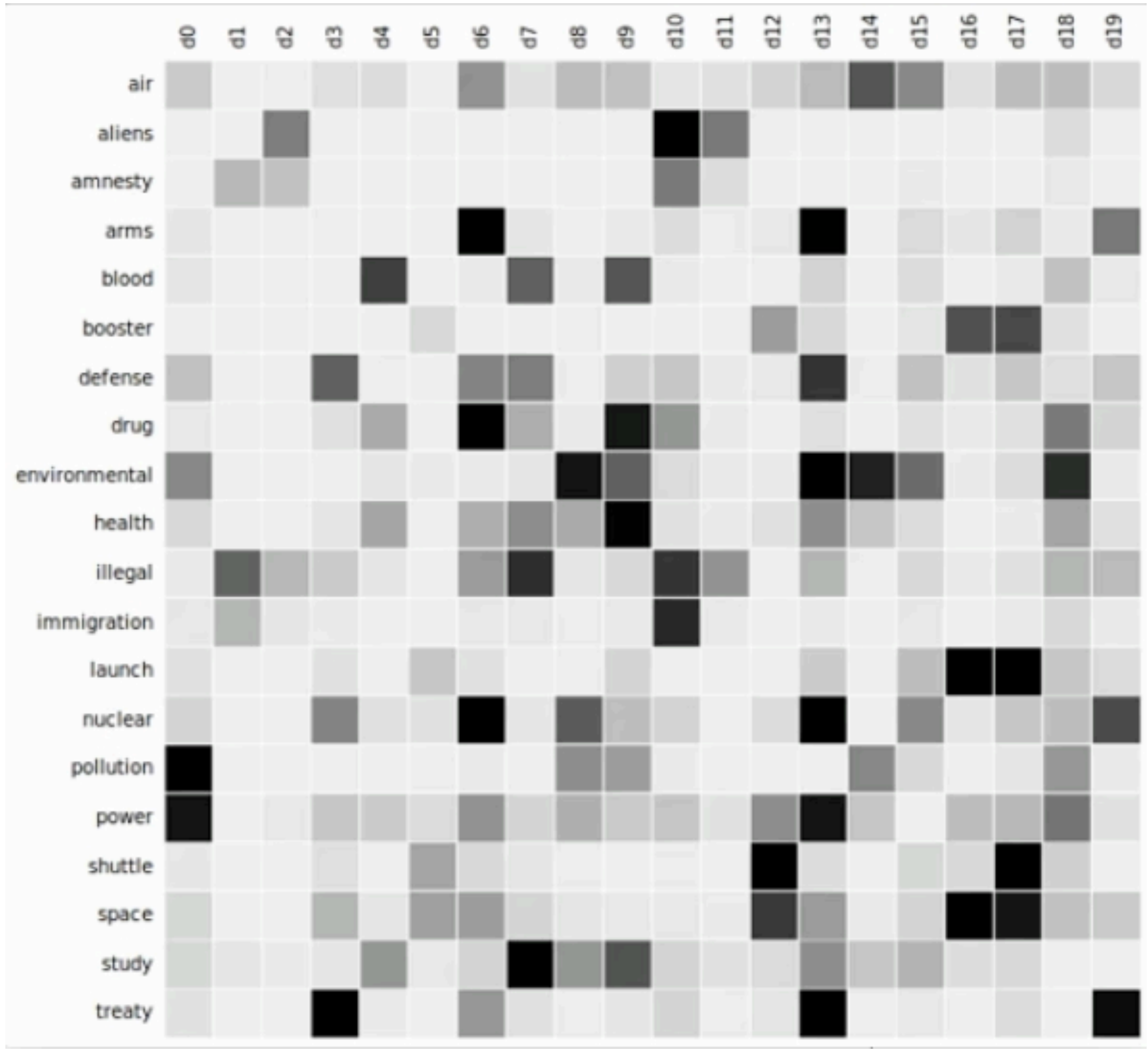
Document vector

$$\underset{(k \times k)}{S} \times \underset{(k \times c)}{V} = \underset{(k \times c)}{D}$$

Diagonal matrix

Othogonal to U

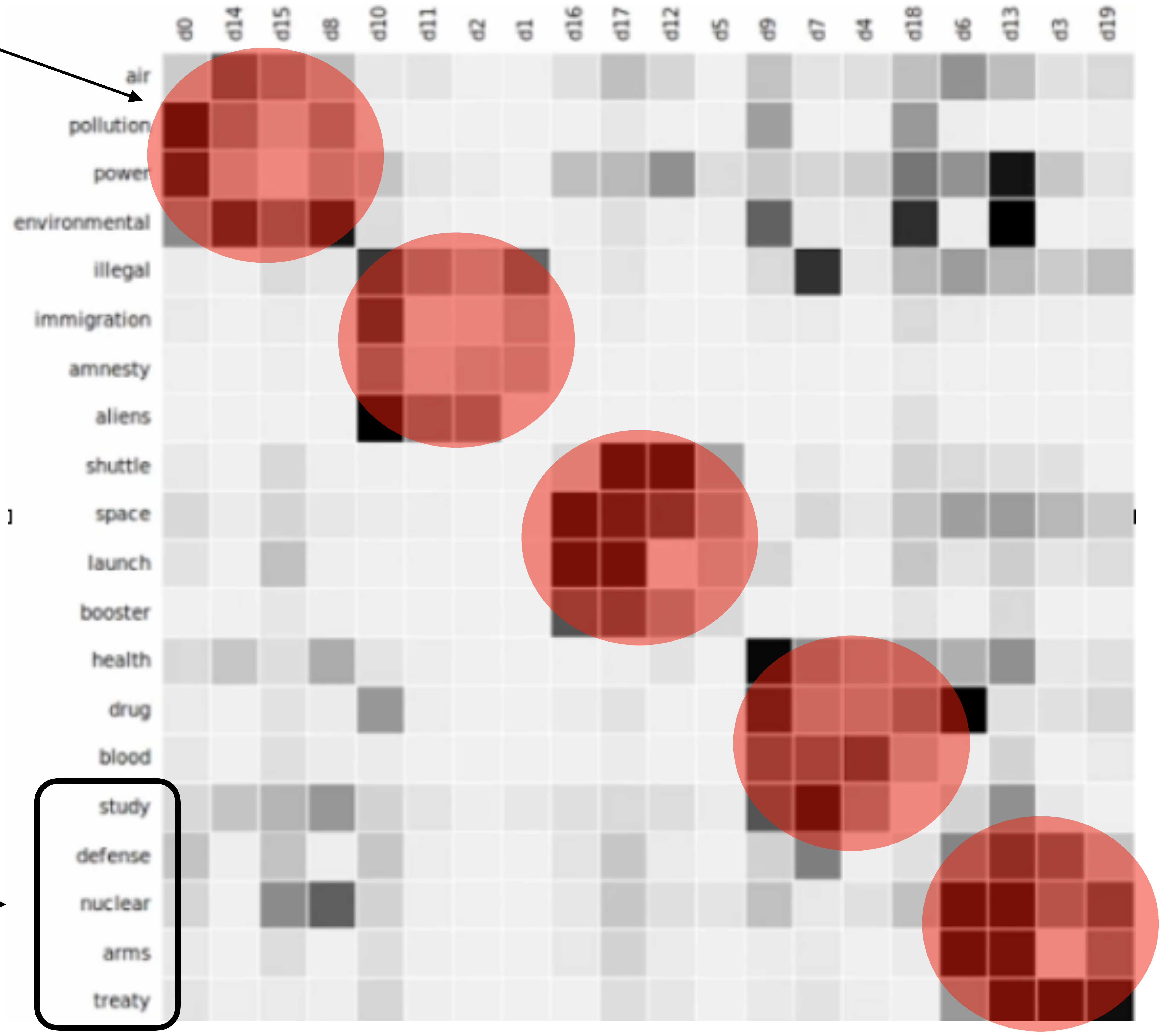
Intuitions



d0
d1
d2
d3
d4
d5
d6
d7
d8
d9
d10
d11
d12
d13
d14
d15
d16
d17
d18
d19
air
aliens
amnesty
arms
blood
booster
defense
drug
environmental
health
illegal
immigration
launch
nuclear
pollution
power
shuttle
space
study
treaty

Intuitions

Can effectively be reduced



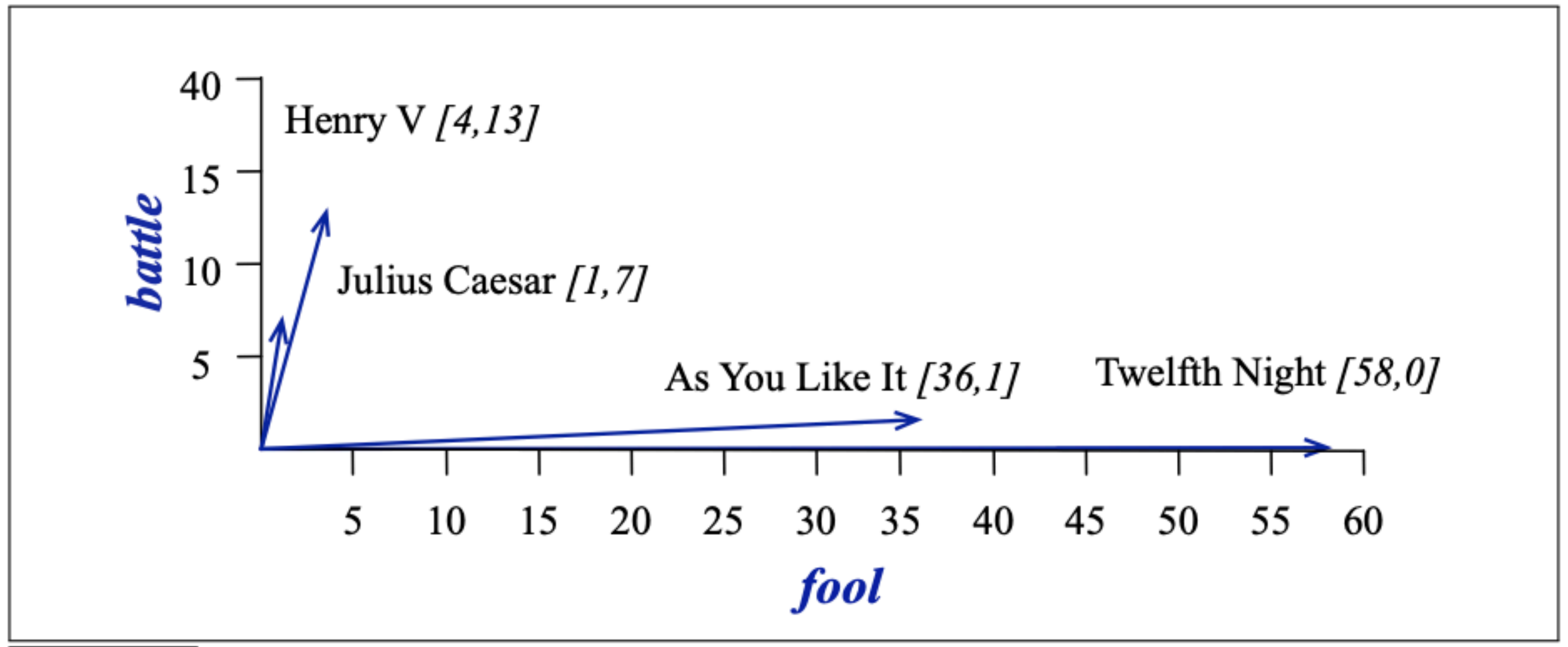
Topical words

Sources & Notes

Using compressions of count vectors have been a used for a while in humanities to get an overview of a corpus
Talk tomorrow by Marton will go into such approaches

Aggregating word vectors

- Documents representations
  - aggregate word representations



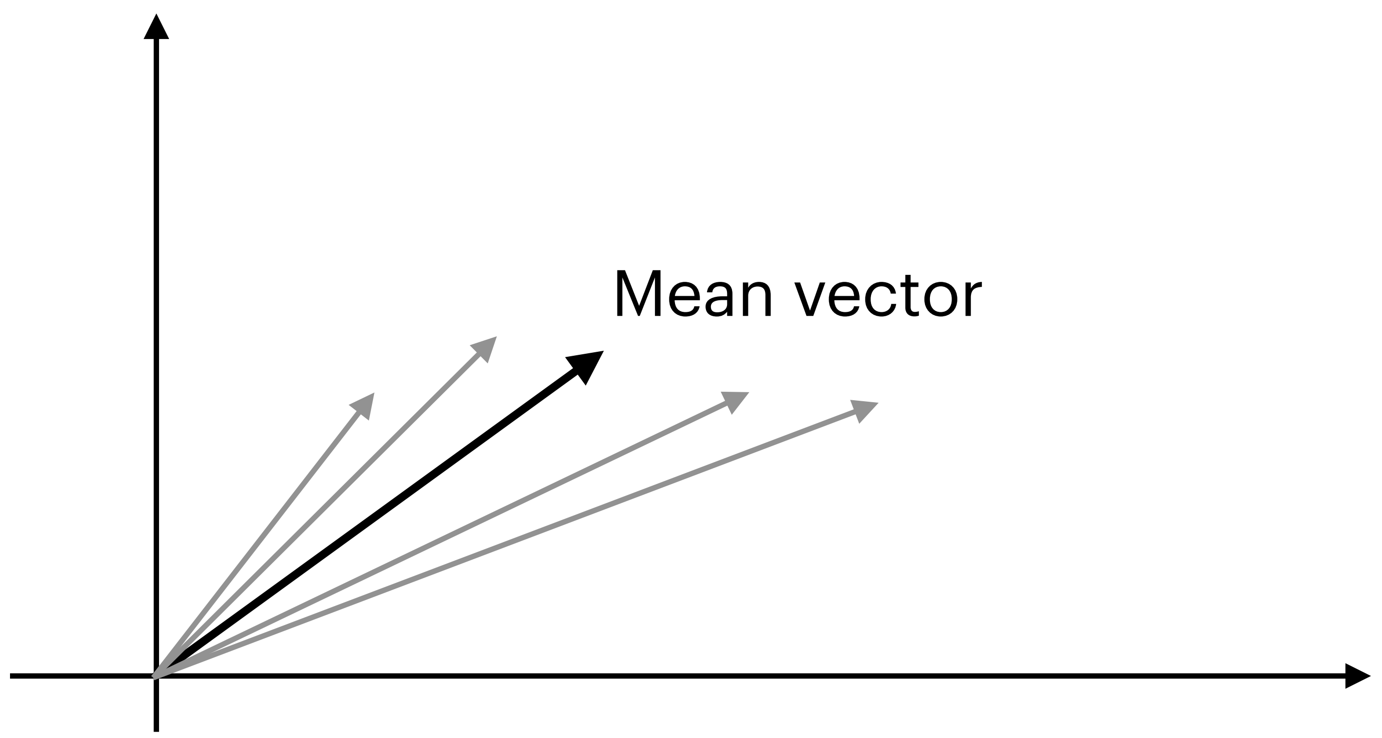
	W1	W2	...
Battle	0	1	
Good	0	0	
fool	0	0	
wit	1	0	

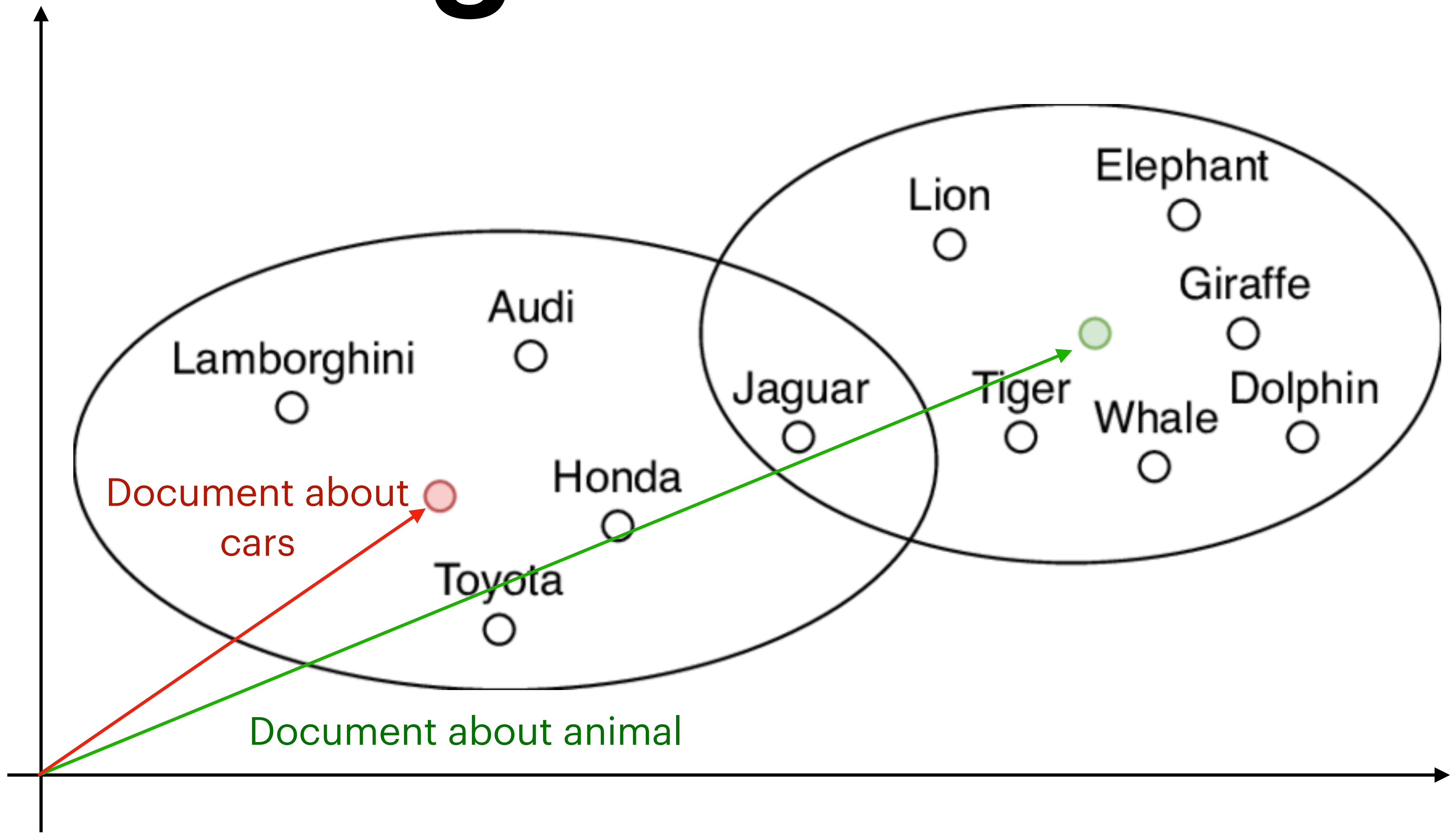
Sum →

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Aggregation word embeddings?

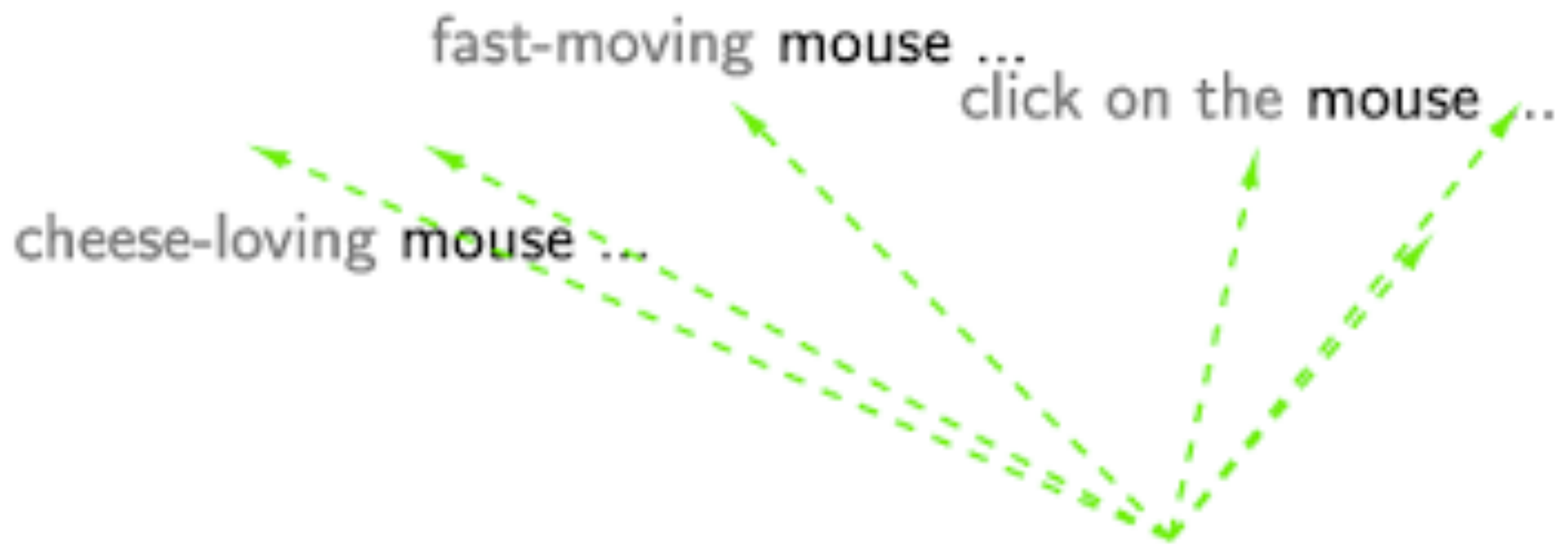
- What happens when we **aggregate word embeddings** using the **mean**?





Aggregating **Contextual** word embeddings

- mean
- Used in many state-of-the-art retrieval systems
  - e.g. SentenceTransformers



Aggregation Methods

- Simple Aggregations
  - Mean
  - Sum
  - …
- **Model-based** Aggregation
  - Recurrent Neural Networks
  - Transformers/Attention*

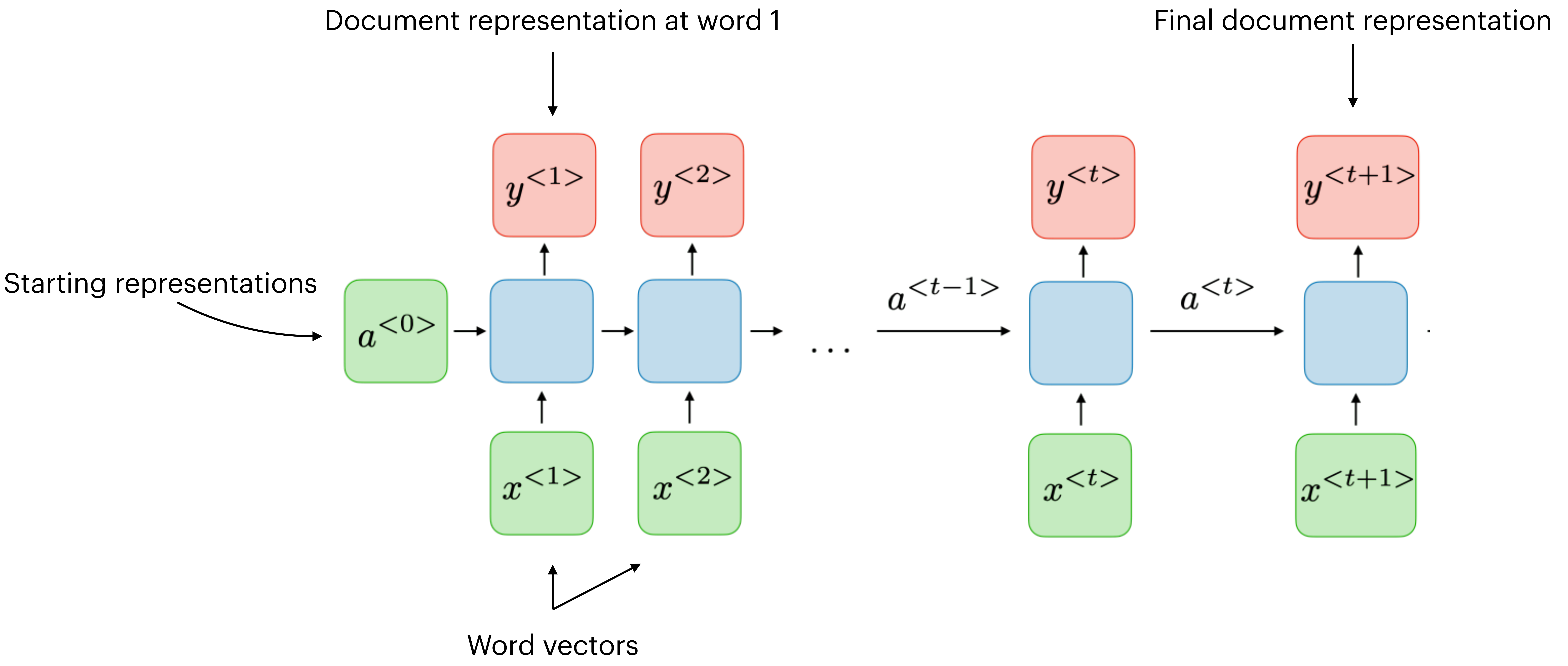
We are just going for the ideas

Sources
& Notes

*We will get back to this one

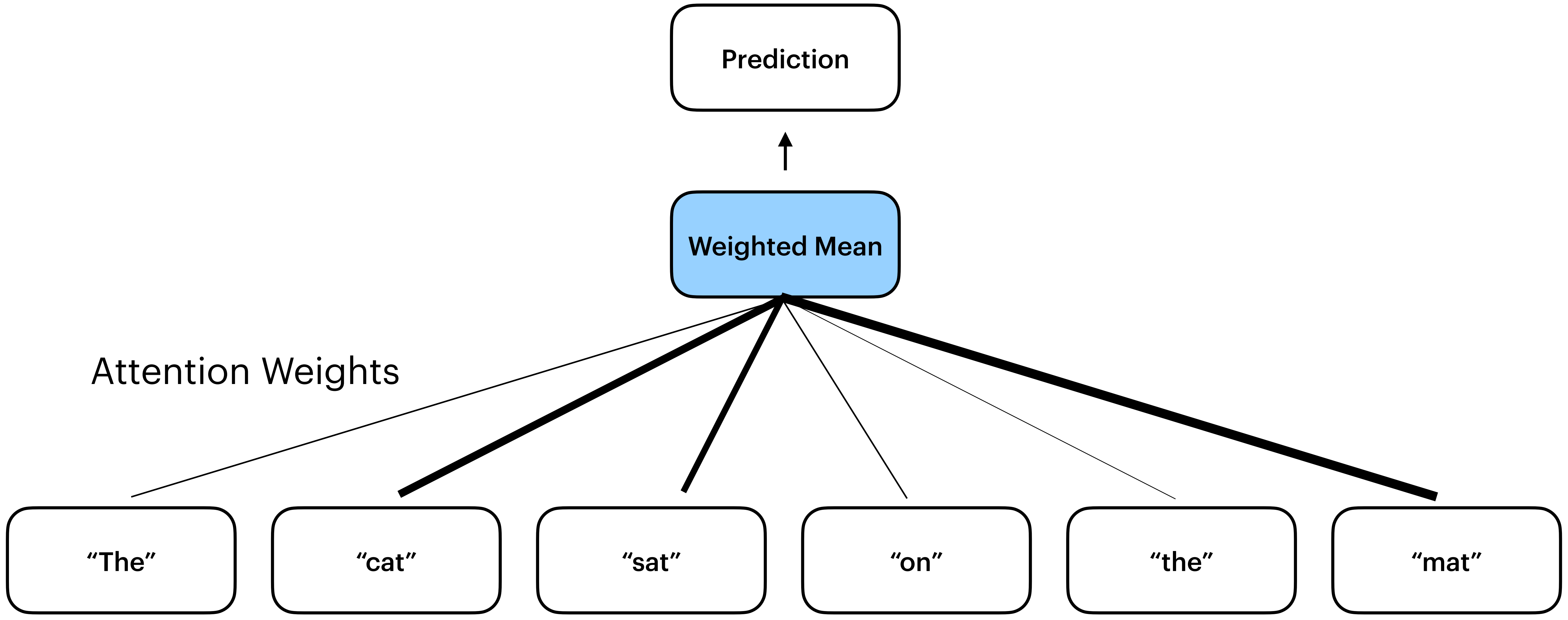
CENTER FOR
HUMANITIES
COMPUTING

Idea: Recurrent Neural Networks

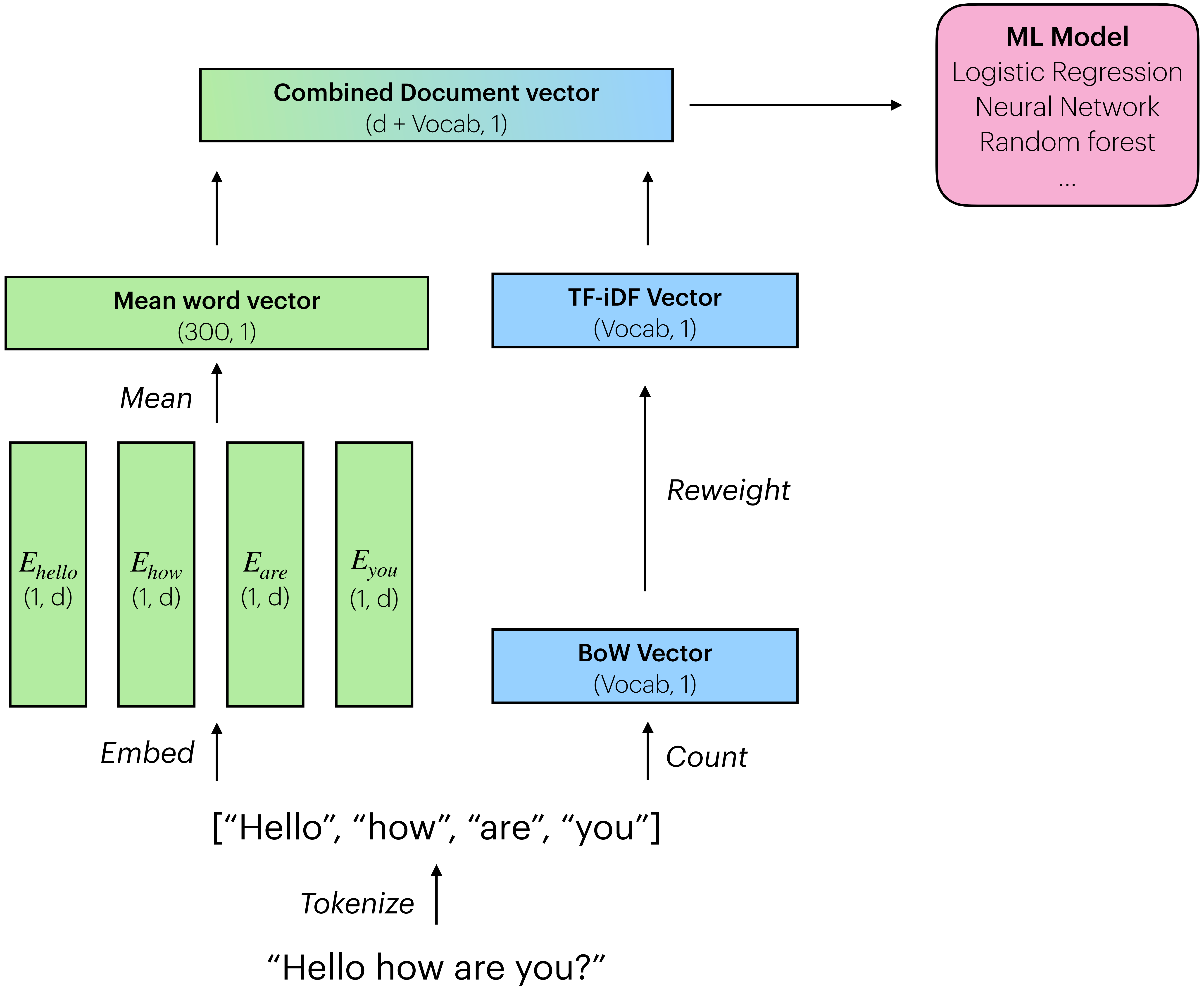


Idea: Transformers and Attention

- Less dependence
- Can run in **parallel**



Combining Vector representation



Any questions?

Next Class: Neural Networks

- Reading:
  - Focus on Ideas and how information flows through the network
  - We have tools for **Backpropergation** and **gradient descent**
  - Encourage you to see 3B1B video ch3-4 as well

Perspectives on Word Vectors
(missing from last class)