INSTITUT TUTTE INSTITUTE

# Hypergraph exploration via vectorization

Valérie Poulin
vpoulin@gmail.com

Leland McInnes, John Healy,
Colin Wear, Benoit Hamelin

- Quickly: text vectorization
- Vertex/hyperedge vectorization
- Demo

INSTITUT TUTTE INSTITUTE

# What this is based of:

- Data to Vectors — VECTORIZERS
- Fast Vector Search — PyNNDescent
- Vectors to Maps — UMAP (Uniform Manifold Approximation & Projection)
- Map Annotation — HDBSCAN
- Map Exploration — TNT (This Not That)

# WARNINGS

- All 2-D vectors are obtained via UMAP
- We use * to indicate that some details are missing
- More details can be found here:
  https://github.com/vpoulin/Hypergraph-Vectorization-recipes/blob/master/notebooks/recipes-03-joint-annotated.ipynb

INSTITUT TUTTE INSTITUTE

# Text vectorization via... counting!

Think word = vertex as we go.

INSTITUT TUTTE INSTITUTE

# Simple Document Vectors

# The “bag-of-words” approach: Discard order and count how often each word occurs

# Bag of Words

| | *a* | *bear* | *big* | *can* | *eat* | *frog* | … | *zoo* |
|---|---|---|---|---|---|---|---|---|
| $d_1$ | 3 | 1 | 1 | 1 | 1 | 1 | … | 1 |
| $d_2$ | 2 | 0 | 0 | 0 | 1 | 1 | … | 0 |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $d_n$ | 1 | 0 | 2 | 0 | 0 | 0 | … | 1 |

Not all words are created equal!

Perform word weighting with an information gain measure

# Information Weight

$$\mathsf{Info}(t) = \sum_{d \in D} P_t(d) \log\left(\frac{P_t(d)}{Q_t(d)}\right)$$

where

$$P_t(d) = \frac{f_{t,d}}{\sum_{d \in D} f_{t,d}} \qquad Q_t(d) = \frac{|d|}{\sum_{d'} |d'|}$$

# Weighted Bag of Words

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| *w(t) =* | *0.001* | *0.15* | *0.05* | *0.02* | *0.1* | *0.17* | | *0.2* |
| | *a* | *bear* | *big* | *can* | *eat* | *frog* | … | *zoo* |
| $d_1$ | .003 | .15 | .05 | .02 | .1 | .17 | … | .2 |
| $d_2$ | .002 | 0 | 0 | 0 | .1 | .17 | … | 0 |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $d_n$ | .001 | 0 | .1 | 0 | 0 | 0 | … | .2 |

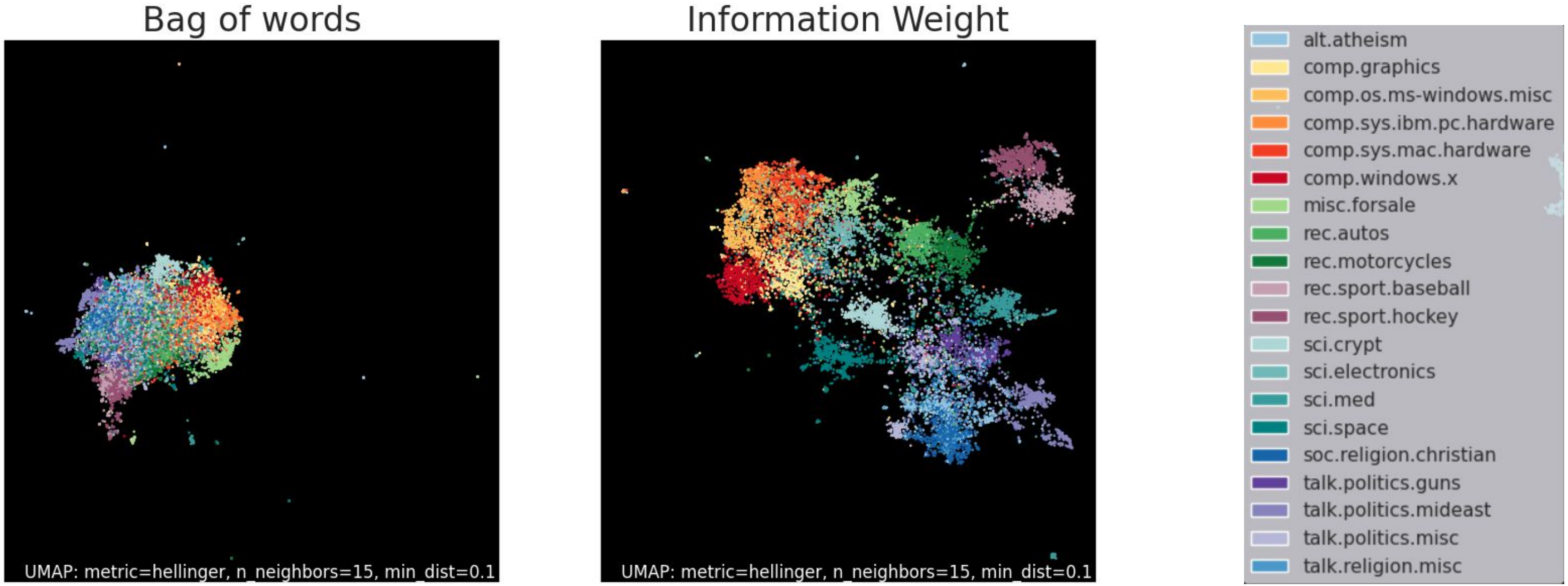
INSTITUT TUTTE INSTITUTE

# 20 Newsgroup Dataset

## (NNTP Newsgroup posts from 1990s)

http://qwone.com/~jason/20Newsgroups/

- alt.atheism
- talk.religion.misc
- soc.religion.christian
- talk.politics.misc
- talk.politics.mideast
- talk.politics.guns
- comp.graphics
- comp.os.ms-windows.misc
- comp.sys.ibm.pc.hardware
- comp.sys.mac.hardware
- comp.windows.x
- sci.crypt
- sci.electronics
- sci.med
- sci.space
- rec.sport.baseball
- rec.sport.hockey
- misc.forsale
- rec.autos
- rec.motorcycles


Bag of words
Information Weight
UMAP: metric=hellinger, n_neighbors=15, min_dist=0.1
UMAP: metric=hellinger, n_neighbors=15, min_dist=0.1
alt.atheism
comp.graphics
comp.os.ms-windows.misc
comp.sys.ibm.pc.hardware
comp.sys.mac.hardware
comp.windows.x
misc.forsale
rec.autos
rec.motorcycles
rec.sport.baseball
rec.sport.hockey
sci.crypt
sci.electronics
sci.med
sci.space
soc.religion.christian
talk.politics.guns
talk.politics.mideast
talk.politics.misc
talk.religion.misc

# Downside: all words are **equidistant**

| | *a* | *bear* | *big* | *can* | *eat* | *frog* | … | *zoo* |
|---|---|---|---|---|---|---|---|---|
| $d_1$ | 3 | 1 | 1 | 1 | 1 | 1 | … | 1 |
| $d_2$ | 2 | 0 | 0 | 0 | 1 | 1 | … | 0 |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $d_n$ | 1 | 0 | 2 | 0 | 0 | 0 | … | 1 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| *frog* | 0 | 0 | 0 | 0 | 0 | 1 | … | 0 |

*d( frog, toad ) = d( frog, car )*

# Better word vectors

Context before

Context after

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod

Target word

Window radius three

# Words not **equidistant**

| | *a* | *bear* | *big* | *can* | *eat* | *frog* | … | *zoo* |
|---|---|---|---|---|---|---|---|---|
| **frog** | 150 | 2 | 1 | 25 | 20 | 0 | … | 2 |
| **toad** | 125 | 1 | 5 | 20 | 17 | 19 | … | 0 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| **car** | 306 | 2 | 129 | 67 | 11 | 0 | … | 3 |

$$d(\,frog, toad\,) < d(\,frog, car\,)$$

Apply SVD* dimension reduction to the matrix of co-occurrence counts to get **word vectors**

INSTITUT TUTTE INSTITUTE

Documents
are (info-weighted) bags* of
word vectors

*Distributions over the word space

Documents

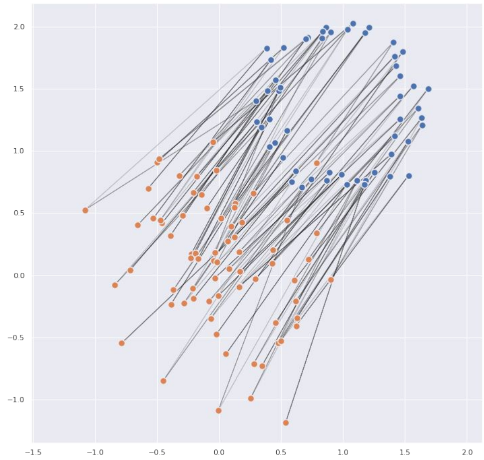
are finite distributions over

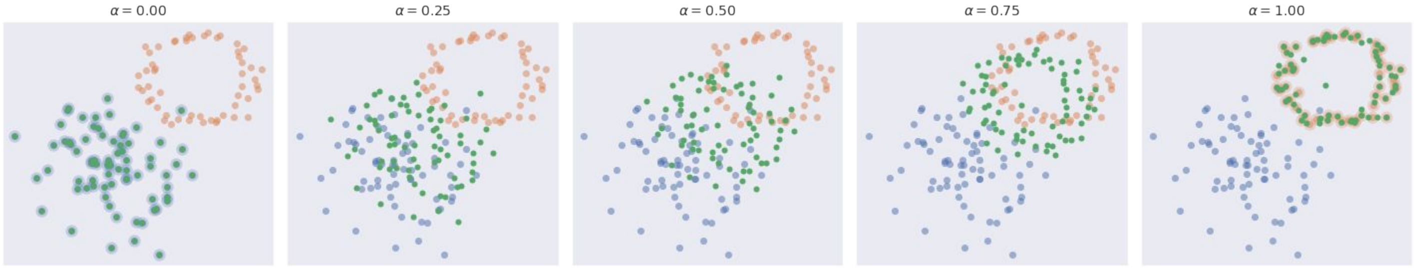
word vectors

# Documents are finite distributions over word vectors

Vectorize so to approximate Wasserstein distance

# Wasserstein: earth mover distance

INSTITUT TUTTE INSTITUTE

VECTORIZERS

Vectorize* documents so that

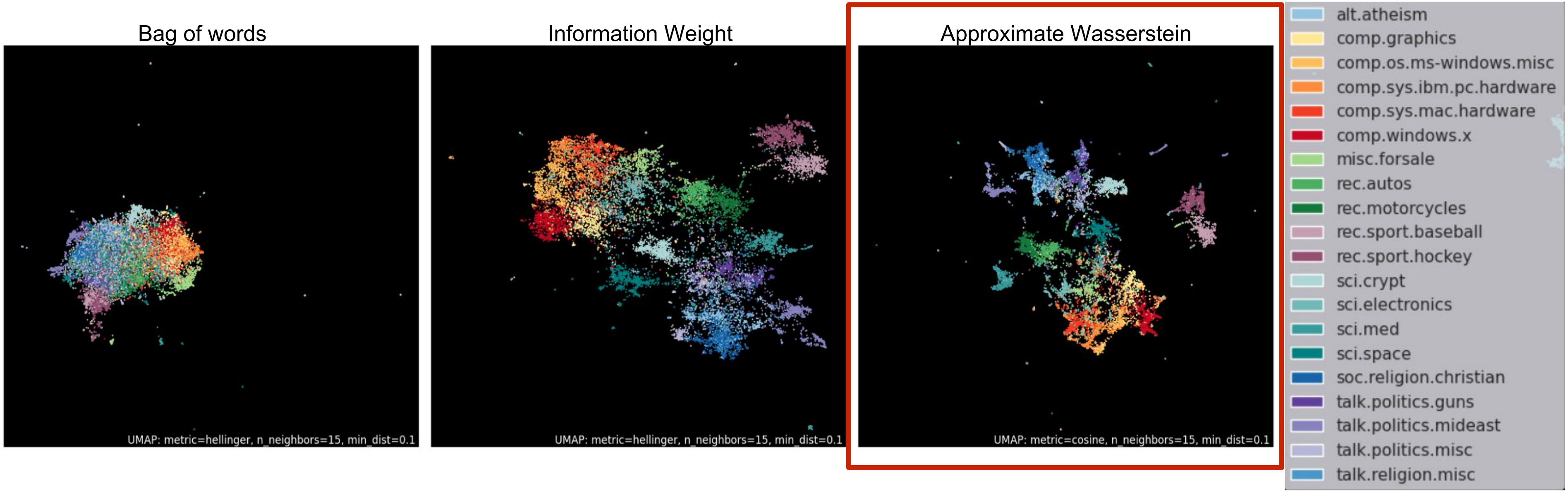
**distances between their respective vectors**

approximate the

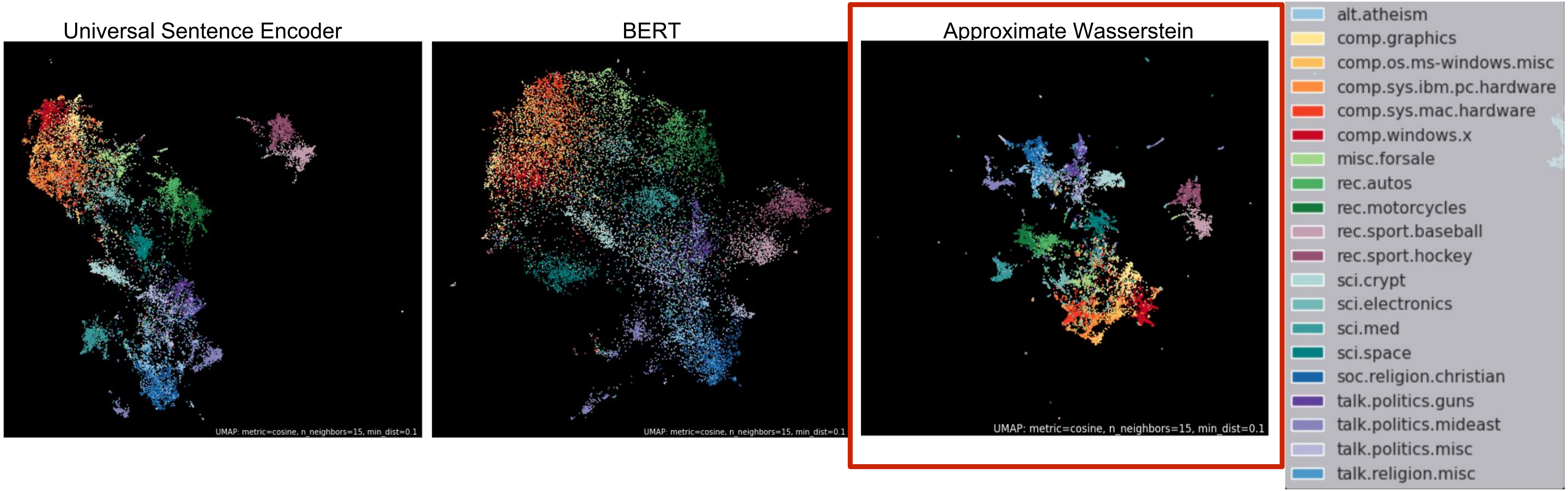
**Wasserstein distances between distributions**

*WASSERSTEIN EMBEDDING FOR GRAPH LEARNING
Soheil Kolouri∗ †, Navid Naderializadeh∗†, Gustavo K. Rohde‡ , & Heiko Hoffmann†, 2021

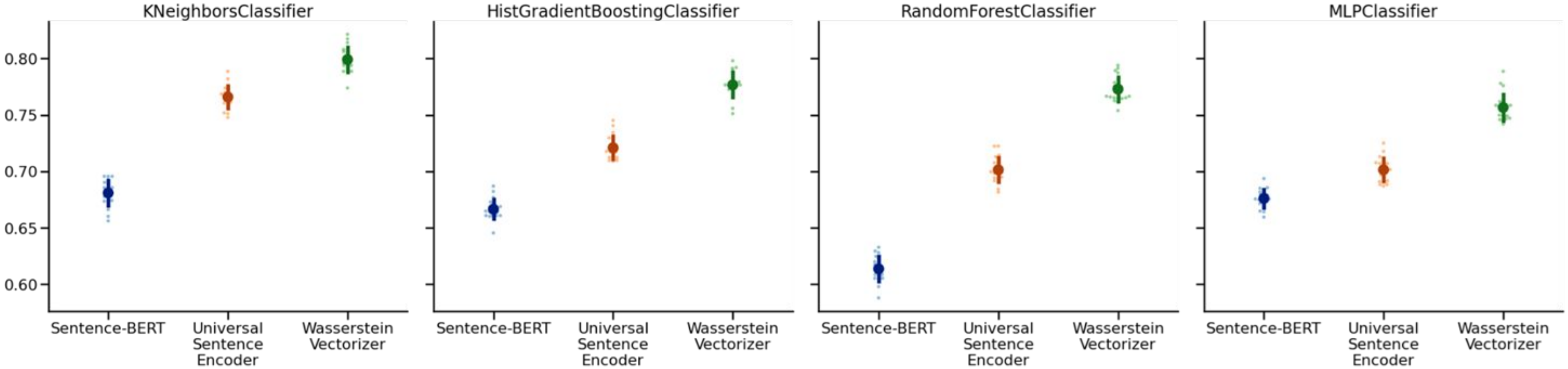
# Vectorizing via counting...



# ...compared against big NNs.



# Evaluate on classification task



# Vertex and hyperedge embedding

INSTITUT TUTTE INSTITUTE

# Vectorizing vertices in hypergraphs vs. words in documents

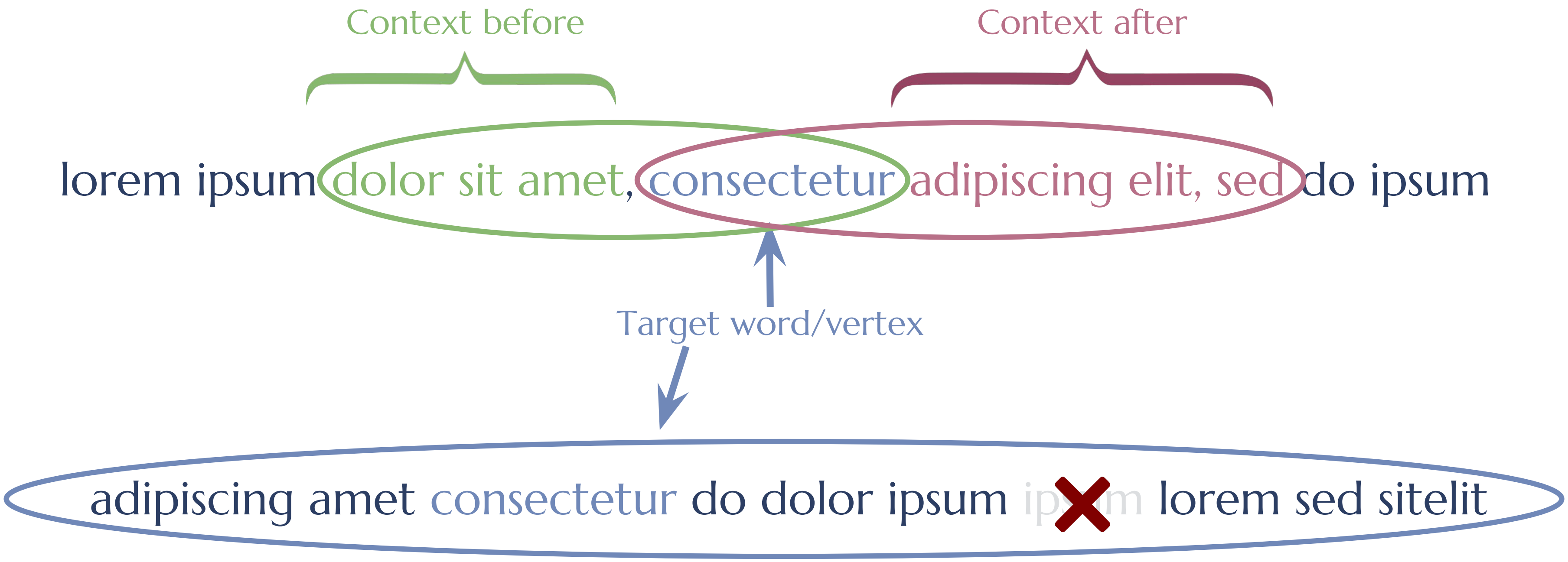
Context before

Context after

lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do ipsum

Target word/vertex

# Vectorizing hyperedges vs. documents



# Hypergraph: What's cooking?

Some summary statistics of the network are:

- number of nodes: 6,714 (ingredients)
- number of hyperedges: 39,774 (recipes)
- number of edge label categories: 20 (cuisine type)
- maximum hyperedge size: 65

**Task:** predict cuisine type of recipes - **hyperedge classification**

https://www.cs.cornell.edu/~arb/data/

# What's cooking? - the data

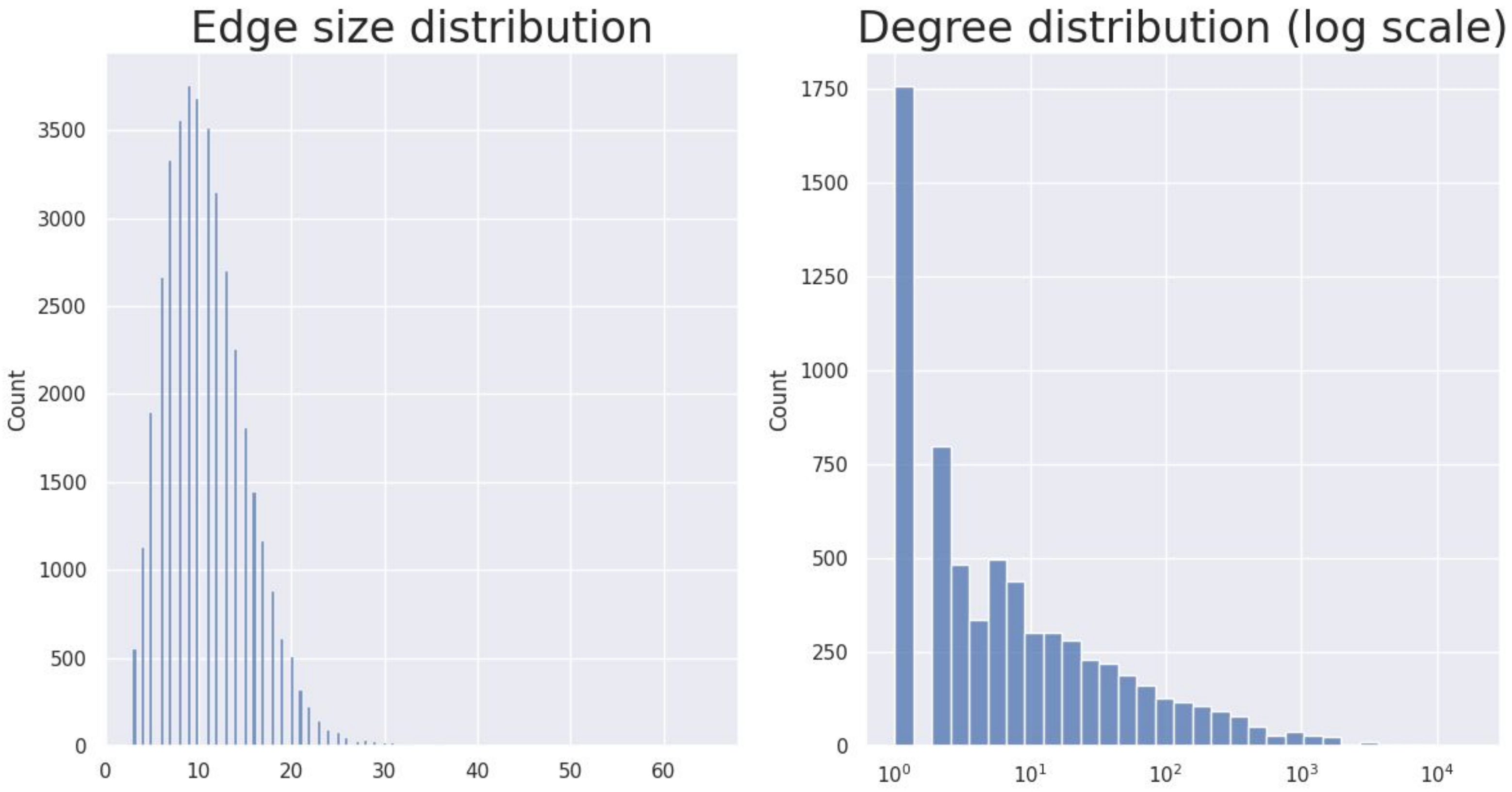

```
{
"id": 24717,
"cuisine": "indian",
"ingredients": [
    "tumeric",
    "vegetable stock",
    "tomatoes",
    "garam masala",
    "naan",
    "red lentils",
    "red chili peppers",
    "onions",
    "spinach",
    "sweet potatoes"
]
},
```

# Hyperedge: bad of ingredients

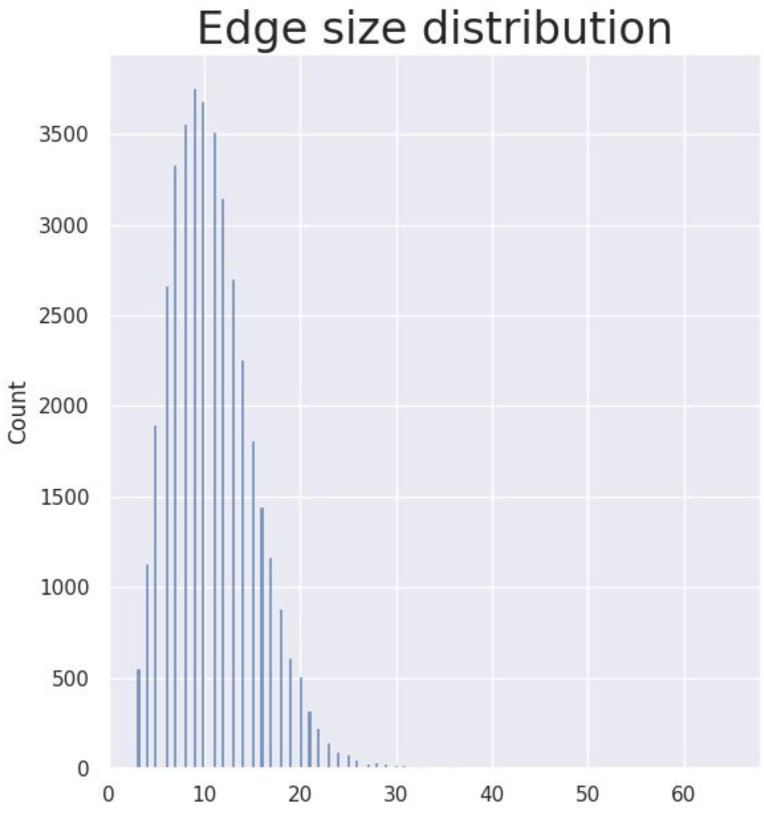
```
recipes = [
            ['water', 'vegetable oil', 'wheat', 'salt'],
            ['greek yogurt', 'lemon curd', 'confectioners sugar', 'raspberries'],
            ['italian seasoning', 'broiler-fryer chicken', 'mayonaise', 'zesty italian dressing'],
            ['sugar', 'hot chili', 'asian fish sauce', 'lime juice'],
            ['flour tortillas', 'cheese', 'breakfast sausages', 'large eggs'],
            ['yellow corn meal', 'boiling water', 'butter', 'fresh parmesan cheese', 'sea salt'],
            ['ice cubes', 'club soda', 'white rum', 'lime', 'turbinado'],
            ['lemon', 'pesto', 'salmon fillets', 'white wine'],
            ...
          ]
```

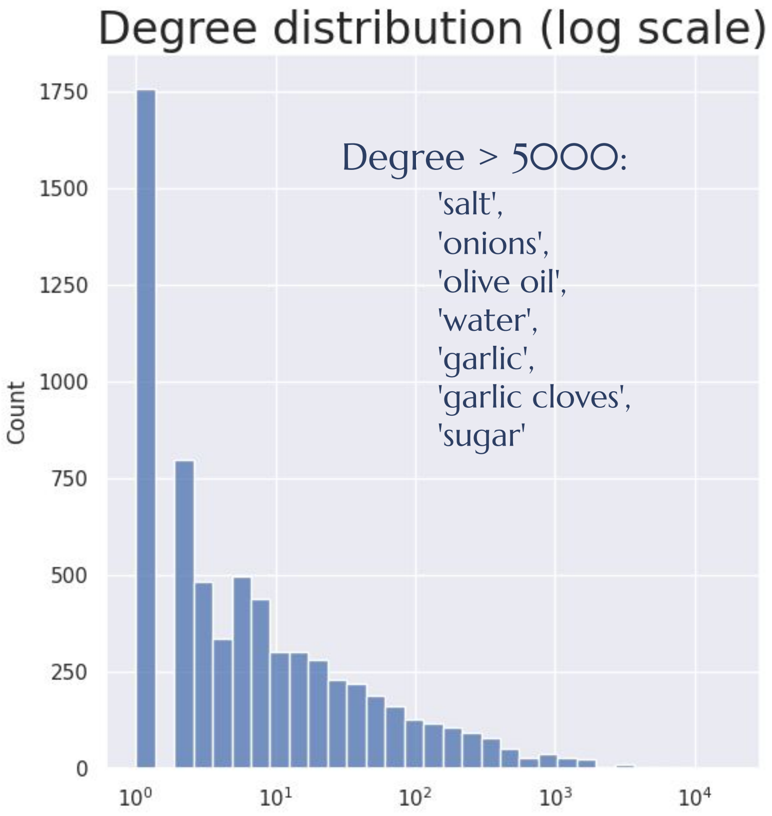
*Remove recipes of size less than 3

# What's cooking? - the data



# What's cooking? - the data



INSTITUT TUTTE INSTITUTE

asian.chinese
asian.filipino
asian.japanese
asian.korean
asian.thai
asian.vietnamese
american.brazilian
american.mexican
american.southern_us
others.greek
others.indian
others.moroccan
others.russian
europe.french
europe.italian
europe.spanish
islands.cajun_creole
islands.jamaican
english.british
english.irish

# Vectorizing hyperedges

# Hyperedge: **bag of vertices**

$H^T$

| | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $v_5$ | $v_6$ | $\ldots$ | $v_n$ |
|---|---|---|---|---|---|---|---|---|
| $e_1$ | 1 | 1 | 0 | 0 | 1 | 0 | … | 1 |
| $e_2$ | 0 | 0 | 0 | 1 | 0 | 0 | … | 0 |
| $\vdots$ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| $e_m$ | 0 | 0 | 1 | 0 | 1 | 1 | … | 1 |

This is really just using rows of the incidence matrix.

```python
from vectorizers.transformers import InformationWeightTransformer

incidence_matrix = vectorizers.NgramVectorizer(
    ).fit_transform(recipes)
```

# Hyperedge: information

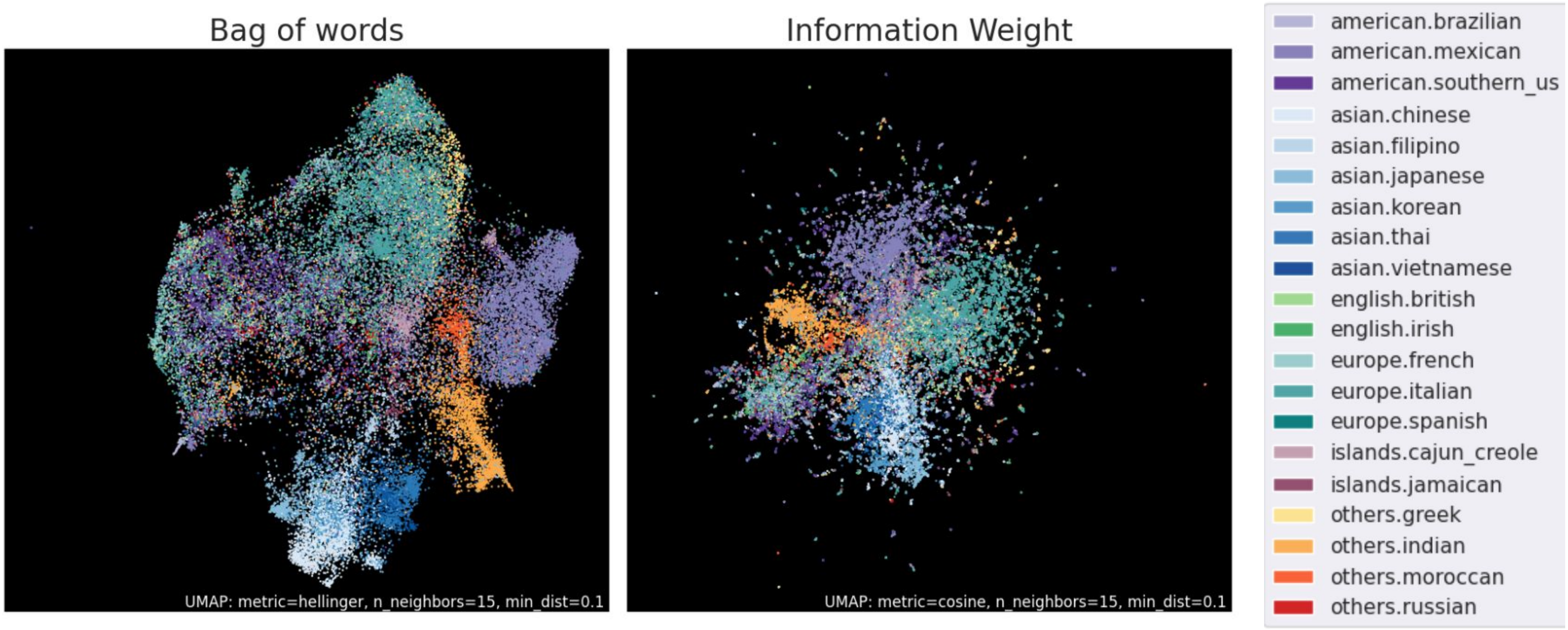
$H^T D_w$

| | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $v_5$ | $v_6$ | $\dots$ | $v_n$ |
|---|---|---|---|---|---|---|---|---|
| $e_1$ | $w_1$ | $w_2$ | 0 | 0 | $w_5$ | 0 | $\dots$ | $w_n$ |
| $e_2$ | 0 | 0 | 0 | $w_4$ | 0 | 0 | $\dots$ | 0 |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $e_m$ | 0 | 0 | $w_3$ | 0 | $w_5$ | $w_6$ | $\dots$ | $w_n$ |

This is rows of the column-weighted incidence matrix.

$$w_i = C - \frac{1}{\deg(v_i)} \sum_{e:\, v_i \in e} \log\left(|e| \cdot \deg(v_i)\right)$$

```python
from vectorizers.transformers import InformationWeightTransformer

info_incidence = InformationWeightTransformer().fit_transform(
    incidence_matrix
)
```


Bag of words
Information Weight
UMAP: metric=hellinger, n_neighbors=15, min_dist=0.1
UMAP: metric=cosine, n_neighbors=15, min_dist=0.1
american.brazilian
american.mexican
american.southern_us
asian.chinese
asian.filipino
asian.japanese
asian.korean
asian.thai
asian.vietnamese
english.british
english.irish
europe.french
europe.italian
europe.spanish
islands.cajun_creole
islands.jamaican
others.greek
others.indian
others.moroccan
others.russian

# Downside: vertices are all **equidistant**

*d( milk, cream ) = d( milk, wasabi )*

# Steps for vectorizing **hyperedges**

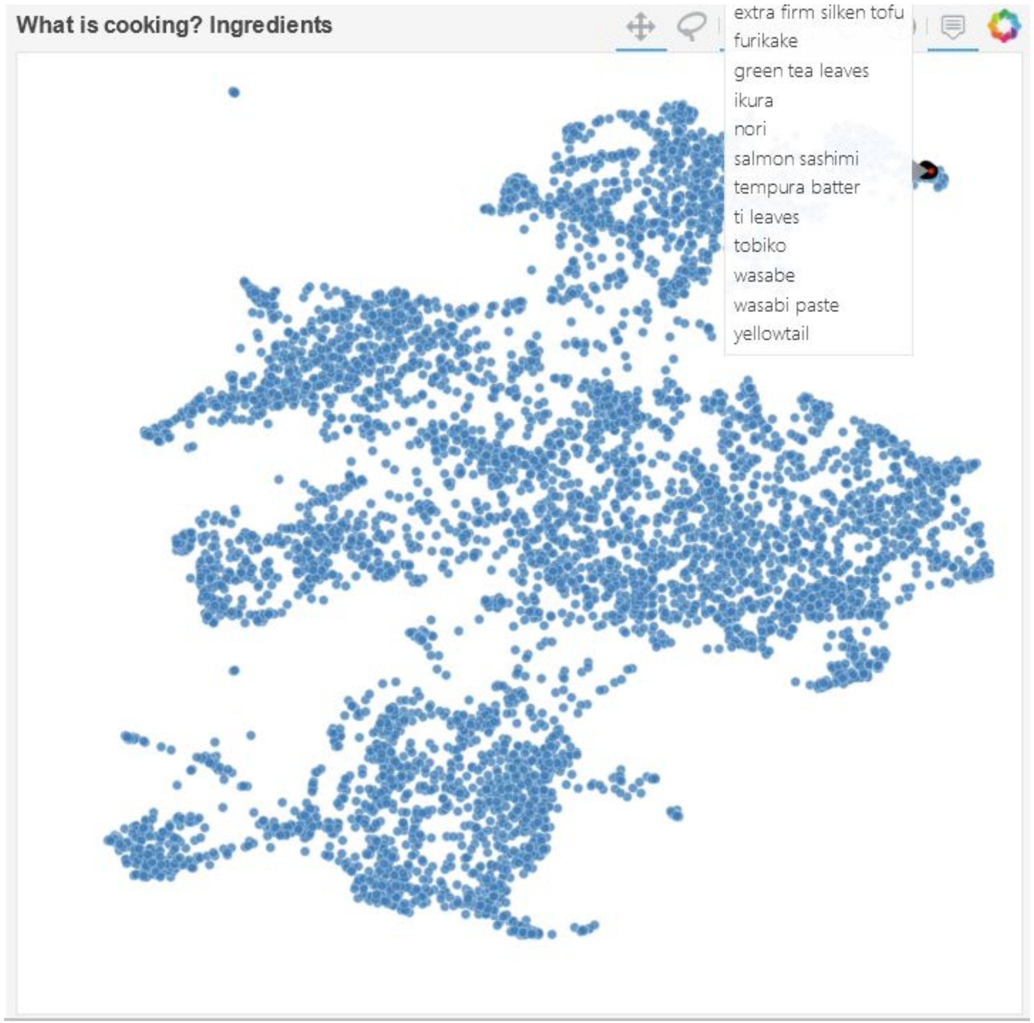
- Vectorize **vertices** using cooccurrences
- Vectorize **hyperedges**
  - Hyperedges are bags/distributions of vertex vectors
  - Vectorize so as to approximate Wasserstein distance between distributions

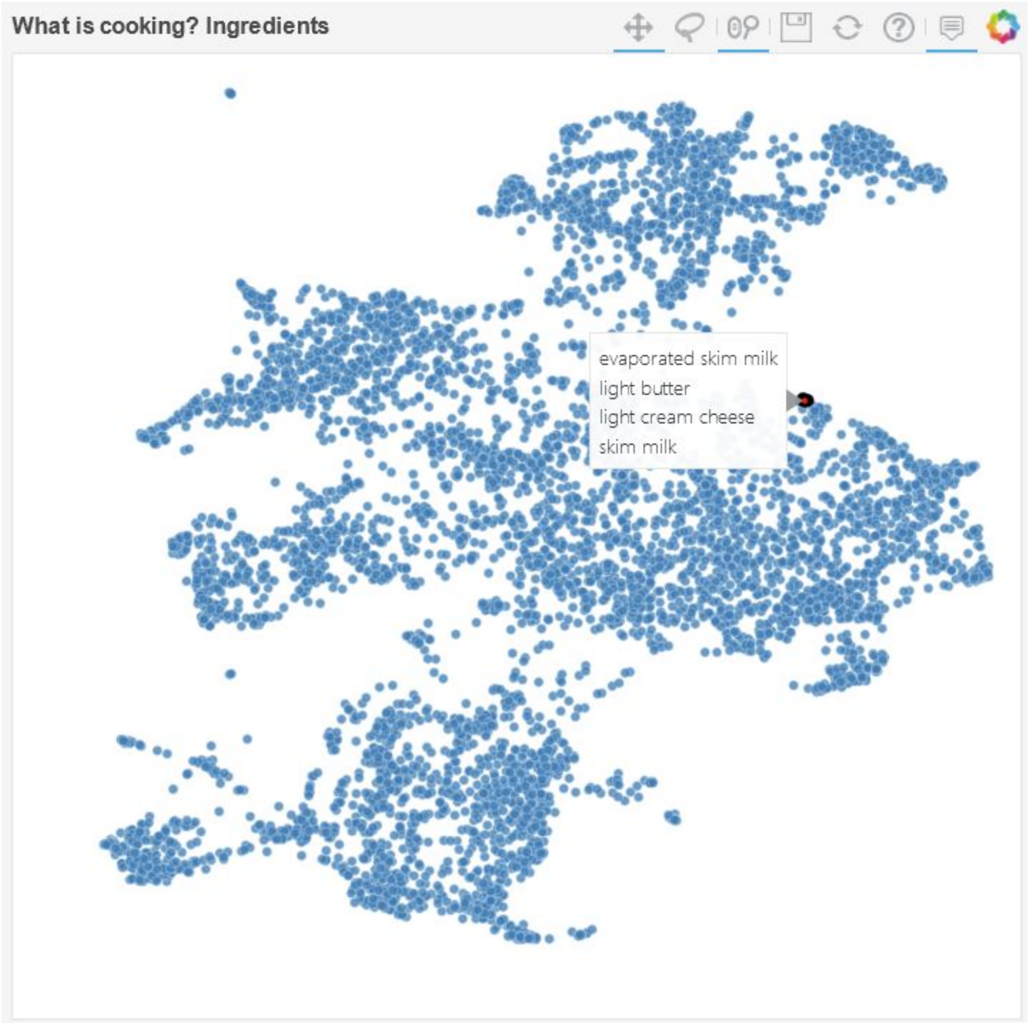
```
from vectorizers import WassersteinVectorizer

vertex_vectors =* 
```

$$HH^T - D_e$$

```
hyperedge_vectors = WassersteinVectorizer().fit_transform(
    info_incidence ,
    vectors = vertex_vectors
)
```


What is cooking? Ingredients
cayenne
cayenne pepper
green bell pepper
onion powder


What is cooking? Ingredients
extra firm silken tofu
furikake
green tea leaves
ikura
nori
salmon sashimi
tempura batter
ti leaves
tobiko
wasabe
wasabi paste
yellowtail


What is cooking? Ingredients
evaporated skim milk
light butter
light cream cheese
skim milk

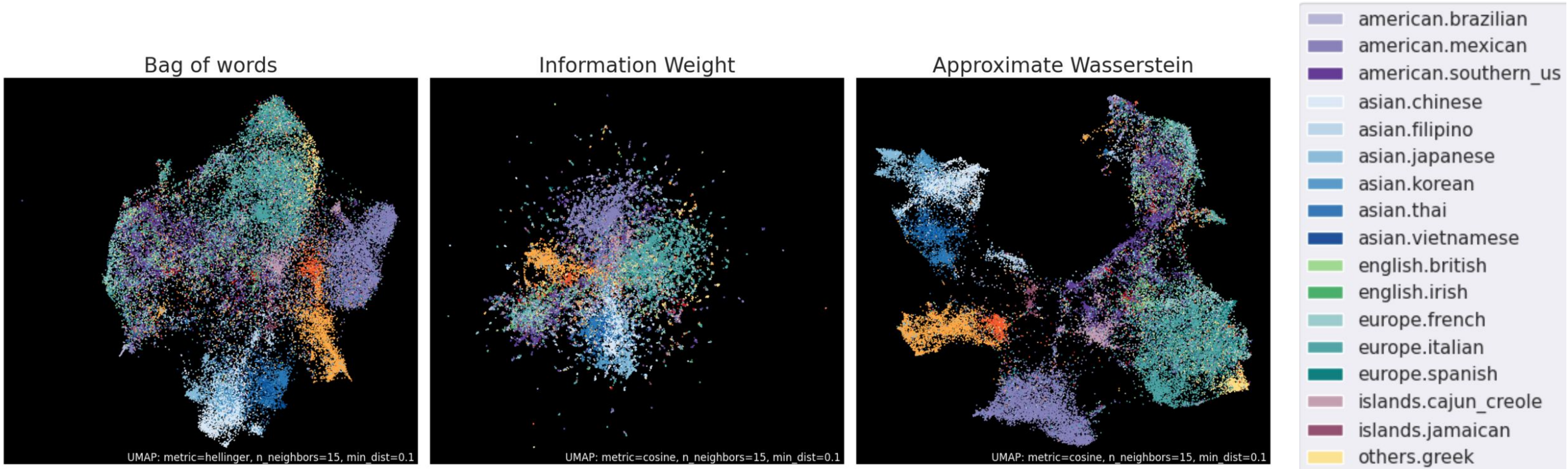

What is cooking? Ingredients
Bertolli Garlic Alfredo Sauce
Bertolli® Alfredo Sauce
Bertolli® Arrabbiata Sauce
Bertolli® Classico Olive Oil
bertolli organic tradit sauc
bertolli vineyard premium collect marinara with burgundi wine sauc
bertolli vodka sauc made with fresh cream
linguine, cook and drain

```
from vectorizers import WassersteinVectorizer

vertex_vectors =*
```

$$HH^T - D_e$$

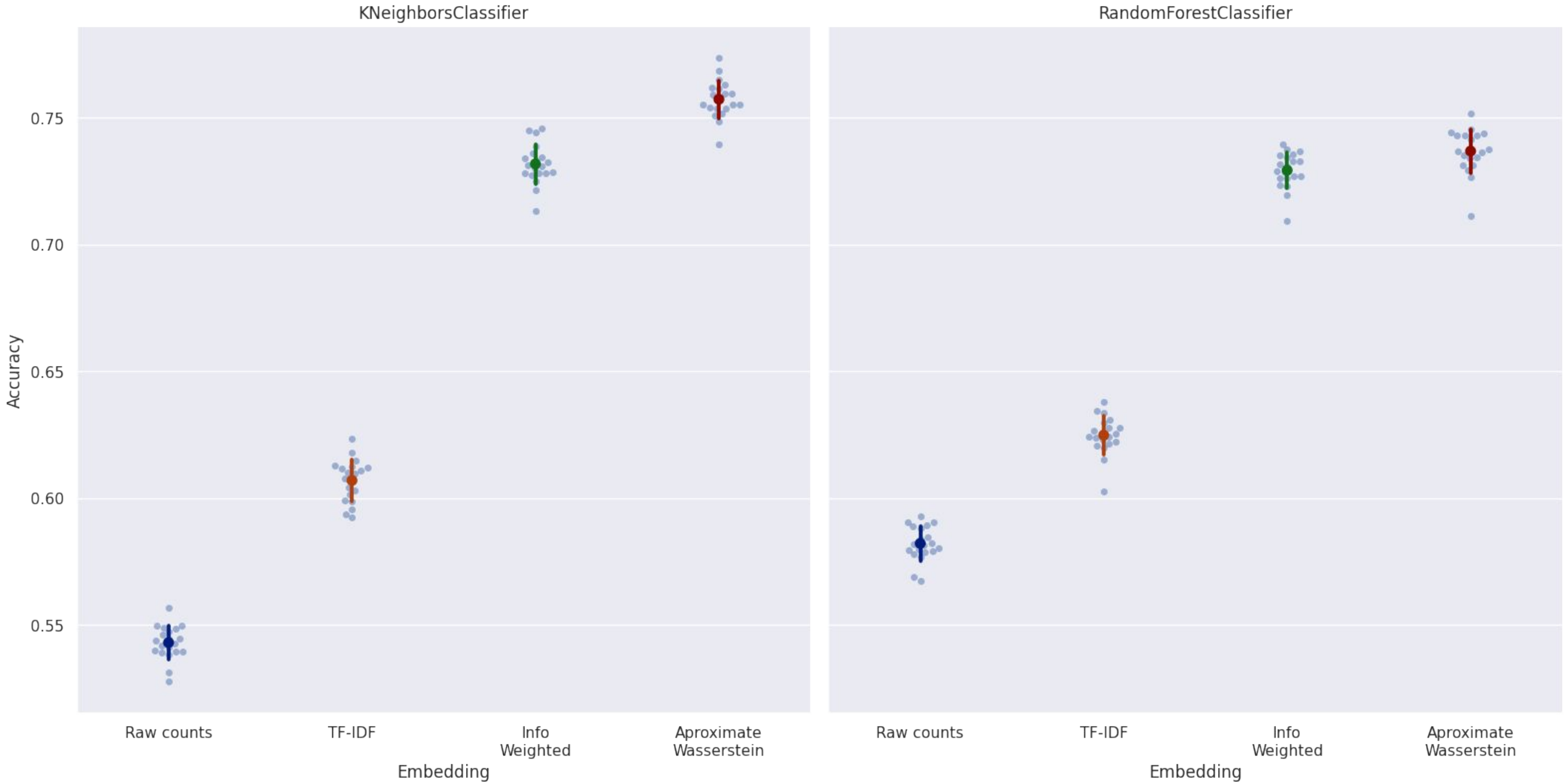
```
hyperedge_vectors = WassersteinVectorizer().fit_transform(
    info_incidence ,
    vectors = vertex_vectors
)
```

# Hypergraph: What's cooking?




| | | |
|---|---|---|
| Rows of incidence matrix. | Information weighted incidence | Hyperedge vectors come from Wasserstein distance between vertex-vector-distributions. |

# Hypergraph: What's cooking?



# Vectorizing hyperedges and vertices: Joint embedding

INSTITUT TUTTE INSTITUTE

Hyperedges are distributions of vertex vectors

Vertices are Dirac distributions on single vertex vector

```python
from vectorizers import WassersteinVectorizer

hyperedge_vertex_vectors = WassersteinVectorizer().fit_transform(
    vstack([info_incidence , identity_on_vertices]),
    vectors = vertex_vectors
)
```

INSTITUT TUTTE INSTITUTE

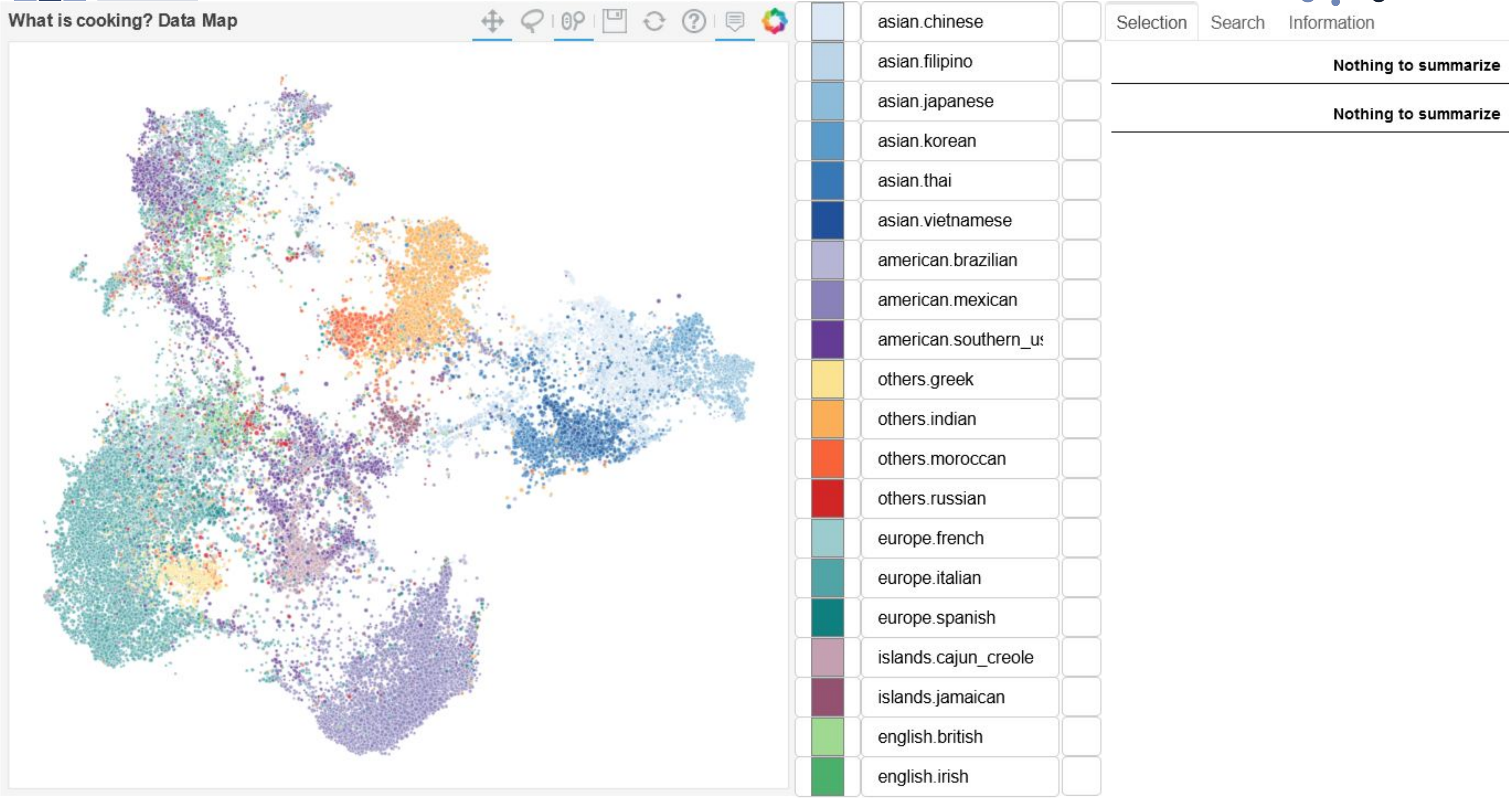
# Explore the results with

TNT

THIS NOT THAT

INSTITUT TUTTE

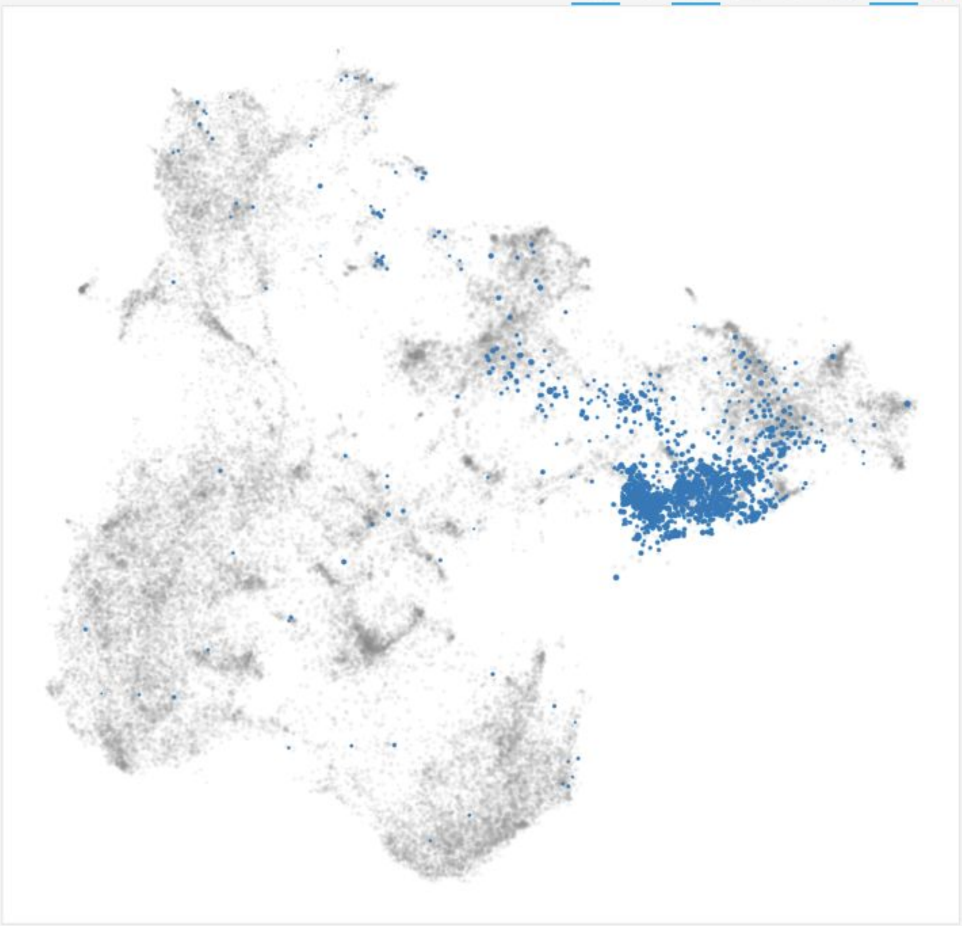
TNT THIS NOT THAT



INSTITUT TUTTE

TNT THIS NOT THAT

What is cooking? Data Map



- asian.chinese
- asian.filipino
- asian.japanese
- asian.korean
- asian.thai ✓
- asian.vietnamese
- american.brazilian
- american.mexican
- american.southern_us
- others.greek
- others.indian
- others.moroccan
- others.russian
- europe.french
- europe.italian
- europe.spanish
- islands.cajun_creole
- islands.jamaican
- english.british
- english.irish

Selection Search Information

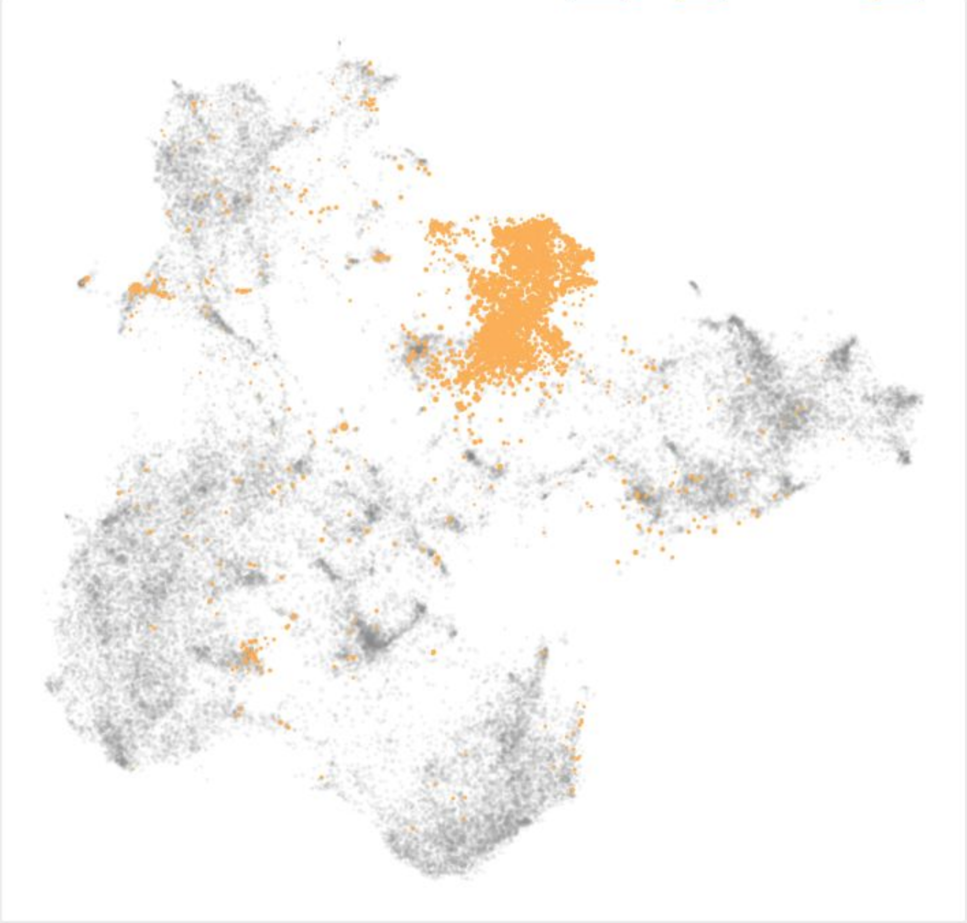
| | value |
|---|---|
| count | 1535 |

| | labels | distances |
|---|---|---|
| 0 | lemongrass | 0.301840 |
| 1 | thai basil | 0.306907 |
| 2 | lemon grass | 0.351924 |
| 3 | palm sugar | 0.364556 |
| 4 | kaffir lime leaves | 0.370222 |
| 5 | Thai fish sauce | 0.374322 |
| 6 | fish sauce | 0.379578 |
| 7 | thai chile | 0.392770 |
| 8 | asian fish sauce | 0.405768 |
| 9 | Thai red curry paste | 0.412965 |

INSTITUT TUTTE

TNT THIS NOT THAT

What is cooking? Data Map



- asian.chinese
- asian.filipino
- asian.japanese
- asian.korean
- asian.thai
- asian.vietnamese
- american.brazilian
- american.mexican
- american.southern_us
- others.greek
- others.indian ✓
- others.moroccan
- others.russian
- europe.french
- europe.italian
- europe.spanish
- islands.cajun_creole
- islands.jamaican
- english.british
- english.irish

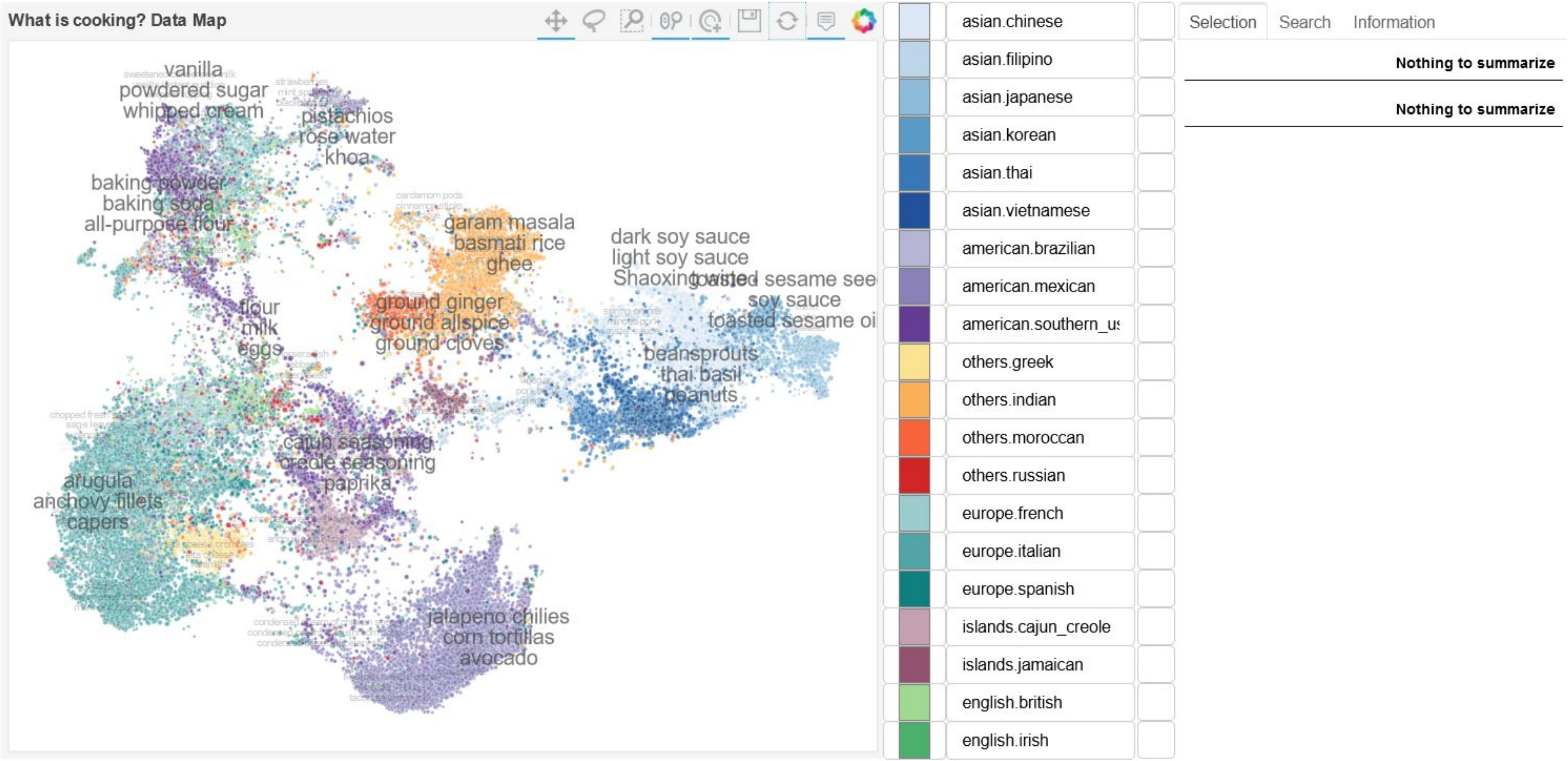
Selection | Search | Information

| | value |
|---|---|
| count | 2984 |

| | labels | distances |
|---|---|---|
| 0 | ghee | 0.297796 |
| 1 | ground turmeric | 0.354122 |
| 2 | garam masala | 0.361167 |
| 3 | basmati rice | 0.372548 |
| 4 | coriander powder | 0.389299 |
| 5 | tumeric | 0.396158 |
| 6 | cashew nuts | 0.411860 |
| 7 | yoghurt | 0.417711 |
| 8 | curry leaves | 0.428681 |
| 9 | cumin seed | 0.433862 |

# Automatic annotation:

1. Hierarchical clustering of hyperedges
2. Identify vertices closest to cluster centroid
3. Display on plot at the right resolution

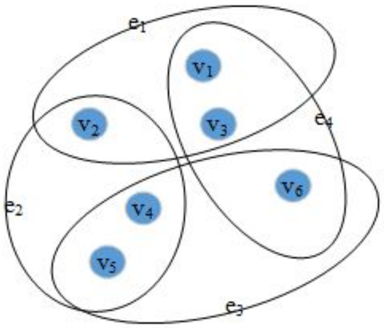


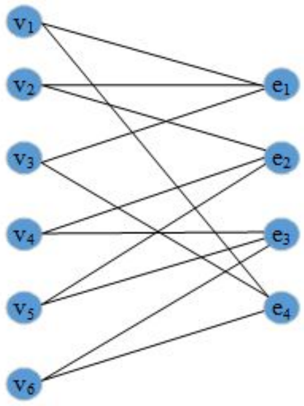
What is cooking? Data Map
vanilla
powdered sugar
whipped cream
pistachios
rose water
khoa
baking powder
baking soda
all-purpose flour
garam masala
basmati rice
ghee
dark soy sauce
light soy sauce
Shaoxing wine
toasted sesame see
soy sauce
toasted sesame oi
ground ginger
ground allspice
ground cloves
flour
milk
eggs
beansprouts
thai basil
peanuts
cajun seasoning
creole seasoning
paprika
arugula
anchovy fillets
capers
jalapeno chilies
corn tortillas
avocado
asian.chinese
asian.filipino
asian.japanese
asian.korean
asian.thai
asian.vietnamese
american.brazilian
american.mexican
american.southern_us
others.greek
others.indian
others.moroccan
others.russian
europe.french
europe.italian
europe.spanish
islands.cajun_creole
islands.jamaican
english.british
english.irish
Selection
Search
Information
Nothing to summarize
Nothing to summarize

# Final words

# Joint embedding based on the dual?

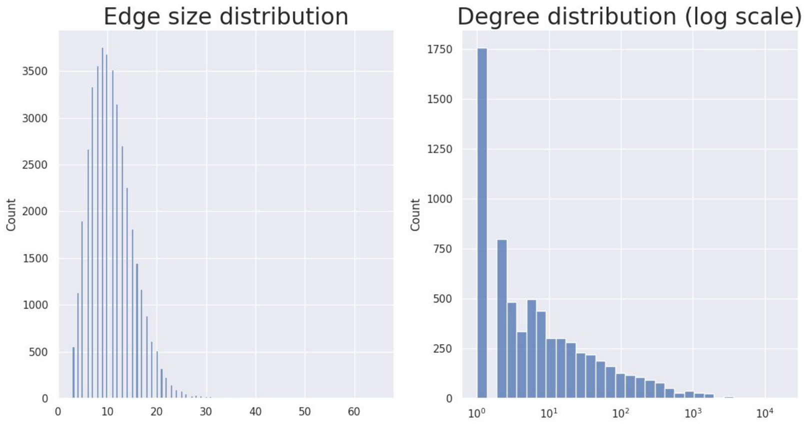
Hypergraph → Joint embedding of vertices and hyperedges




Dual hypergraph → Joint embedding of vertices and hyperedges

Hypergraph → Joint embedding of vertices and hyperedges




Select* the version (dual or not) that has a **non exponential** edge size distribution.

Dual hypergraph → Joint embedding of vertices and hyperedges

*From experiments, more to come.

# What is this representation good/not good for?

# Not good for pure hypergraph questions

- Paths: shortest, number of, centralities,...
- Hyperedge counting
- Motif finding

# Good for hypergraph mining

- Clustering, community finding, data partitioning
- Classifying
- Exploring

We have dropped edge ordering, vertex repetition but…

INSTITUT TUTTE INSTITUTE

Hypergraphs are **chosen** abstractions for problem solving.

Hypergraphs are **chosen** abstractions for problem solving. Often, we have:

- Timestamped events (vertices/edges)
- Edge-dependent vertex weights

INSTITUT TUTTE INSTITUTE

# Hypergraphs are **chosen** abstractions for problem solving.

# Often, we have:

- Timestamped events (vertices/edges)
- Edge-dependent vertex weights

VECTORIZERS **naturally deals with these cases**

Vectorizers:

Vectorizers.readthedocs.io

Vectorizers on hypergraphs

https://github.com/vpoulin/Hypergraph-Vectorization-recipes

Hypergraph datasets

https://www.cs.cornell.edu/~arb/data/