

# Vector Space Embeddings and Data Maps for Cyber Defense

Benoit Hamelin, John Healy, Leland McInnes,  
Richard Millson and Aaron Smith

Tutte Institute for Mathematics and Computing



# What is cyber defense?

# Detect threats to

## Confidentiality

## Integrity

## Availability

# Gather forensic evidence of incidents

Requires data and  
infrastructure monitoring:  
telemetry

# Telemetry analysis:

- Know the infrastructure
- Retrace malicious activity

# Which “the telemetry”

Process command lines  
IP traffic flows  
Event logs  
Malware binaries  
Operating system events  
etc.



# Which “the telemetry”

**Process command lines**

IP traffic flows

Event logs

Malware binaries

Operating system events  
etc.

# ACME3 data

24 hosts, 15 days

~137,847 command lines

~31,000 unique command lines

c:\windows\system32\conhost.exe Oxfffffff -forcevl

c:\windows\system32\mousocoreworker.exe -embedding

c:\program files (x86)\microsoft\edgeupdate\microsoftedgeupdate.exe /ua /installsource scheduler

c:\windows\system32\svchost.exe -k netsvcs -p -s netsetupsvc

c:\windows\system32\taskhostw.exe

....

c:\programdata\microsoft\windows defender\platform\4.18.23090.2008-O\mpcmdrun.exe -idletask -taskname wdcleanup

c:\programdata\microsoft\windows defender\platform\4.18.23090.2008-O\mpcmdrun.exe -idletask -taskname wdverification

c:\windows\system32\taskhostw.exe keyroaming

c:\windows\system32\fontdrvhost.exe

c:\windows\system32\openssh\sshd.exe -r

c:\windows\system32\openssh\sshd.exe -y

c:\windows\system32\cmd.exe

C:\windows\system32\conhost.exe

....

<https://gdo168.llnl.gov/data/ACME-2023>

Key constraint: no label

Unsupervised analysis:  
examine patterns of similarity

Every approach is a lense

Every representation  
tells a story

*Lenses*  
*All ~~models~~ are wrong,*  
*Some are useful*

*George Box*

A useful lense  
provides insight and  
helps with labelling

Data to  
Vectors



Manifold  
Learning



Interactive  
Visualization



and



Supervised  
Cluster  
Inference



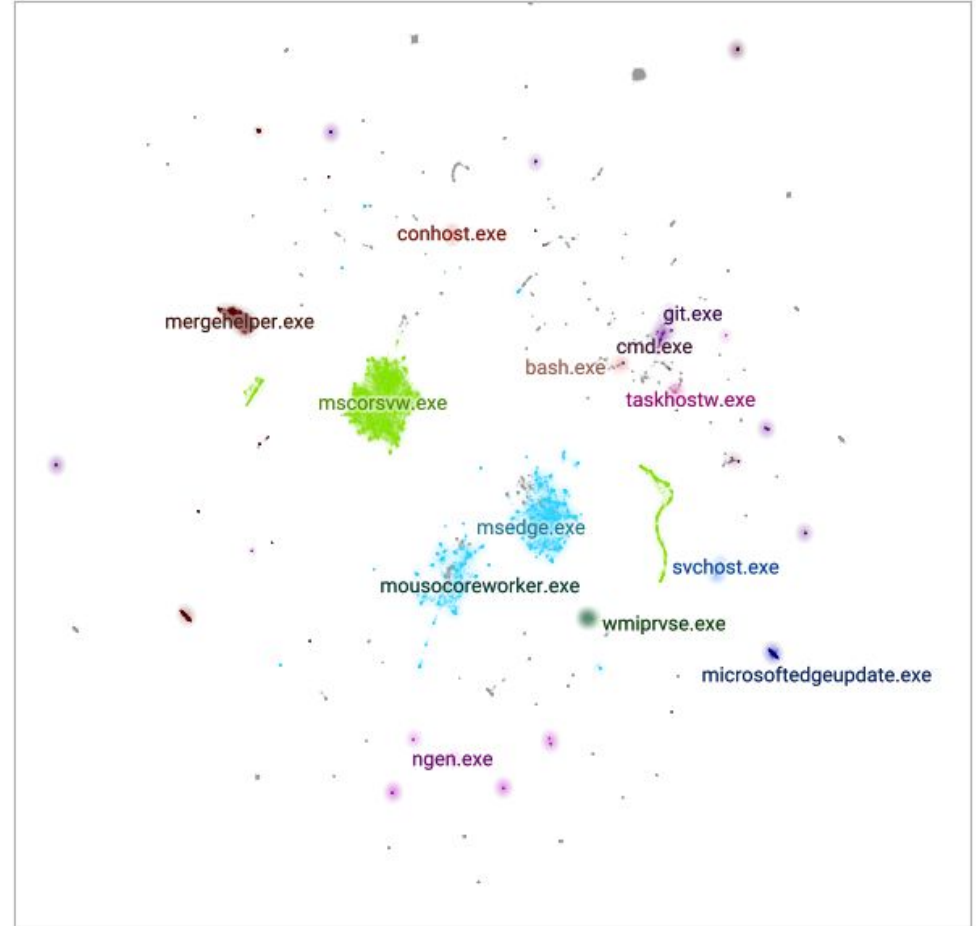


# Data map of processes



## Process instances

as distributions over a cloud of command line token cooccurrence vectors





Data to  
Vectors

Manifold  
Learning

Interactive  
Visualization

Supervised  
Cluster  
Inference

 VECTORIZERS

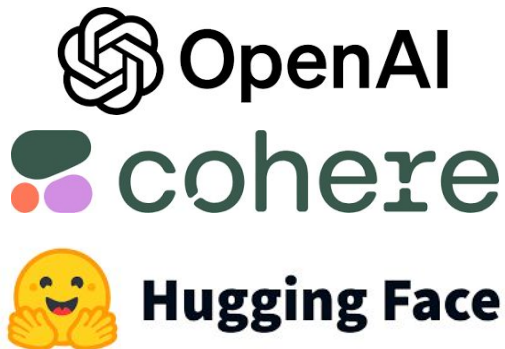
 UNIFORM MANIFOLD  
**UMAP**  
APPROXIMATION & PROJECTION

 **TNT**  
THIS NOT THAT

and

 DATAMAPLOT

 scikit  
*learn*



```
[ 0.0021 0.0246 -0.0134 0.0343 ... -0.0103 -0.0008 0.0047 0.0363 ]
[ 0.0068 0.0098 -0.0079 -0.0006 ... -0.0018 0.0361 0.002 0.0045 ]
[ 0.021 0.0239 0.0024 0.0087 ... -0.0167 -0.0309 -0.0144 -0.0045 ]
[ -0.0082 0.0234 -0.0166 0.0078 ... 0.0375 -0.03 0.0367 -0.0141 ]
[ -0.0535 -0.002 -0.0141 -0.0198 ... -0.0071 -0.0074 0.0042 0.0034 ]
[ 0.0233 0.0273 -0.0139 0.0361 ... 0.0036 -0.0334 0.041 0.0253 ]
[ -0.002 0.0076 0.0048 -0.0299 ... 0.0057 0.0089 0.015 0.0169 ]
[ 0.0066 0.0171 -0.0076 -0.0234 ... -0.0065 -0.0082 -0.0097 0.0065 ]
[ -0.0112 0.0172 0.0051 -0.0432 ... 0.028 -0.0114 0.0481 -0.0051 ]
[ -0.0003 0.0093 -0.0221 0.0302 ... -0.015 -0.0136 0.0012 -0.0205 ]
[ -0.0183 0.011 0.0172 -0.0162 ... -0.0168 -0.003 0.0287 0.0069 ]
[ 0.0392 0.0106 -0.0003 0.0082 ... -0.0287 0.0081 0.0042 0.026 ]
[ 0.0155 0.0163 -0.0126 -0.0035 ... 0.0233 0.0009 -0.028 0.0179 ]
[ -0.0101 0.0118 -0.0006 -0.0225 ... -0.0283 -0.0076 0.0293 0.0114 ]
[ -0.0341 0.0303 -0.0015 -0.0195 ... -0.0459 -0.0501 0.0211 0.0116 ]
[ 0.0011 0.0125 0.0176 -0.0349 ... -0.0147 0.0016 0.035 0.0386 ]
[ -0.0329 0.0044 -0.0127 -0.0076 ... 0. -0.008 0.0692 -0.0044 ]
[ -0.0039 0.0214 -0.0231 0.0164 ... 0.0124 -0.005 0.0266 -0.0446 ]
[ -0.0173 0.0227 0.0188 -0.0349 ... -0.0235 -0.0243 0.004 -0.0072 ]
[ -0.0254 0.0279 -0.0279 -0.0003 ... -0.0189 -0.0109 0.032 0.0067 ]
[ -0.0196 0.0046 0.0117 -0.0074 ... -0.0047 0.0076 0.0297 0.0075 ]
[ -0.0255 0.0056 0.0052 0.0046 ... -0.0005 0.0038 0.0074 0.0015 ]
[ -0.004 0.0218 -0.024 0.0136 ... -0.0122 0.0158 0.001 0.0139 ]
[ -0.0319 0.0259 0.0051 0.0245 ... -0.0092 -0.0121 -0.0023 -0.026 ]
[ 0.0046 0.0147 -0.033 -0.0037 ... -0.0223 0.0213 0.0352 -0.0205 ]
[ -0.0097 0.0017 -0.0027 -0.0412 ... -0.0039 -0.0041 0.0132 0.0119 ]
[ -0.0043 0.0303 -0.0093 -0.0129 ... -0.0054 0.0176 0.0062 0.011 ]
[ 0.0032 -0.0213 -0.0307 -0.0157 ... -0.0327 -0.007 0.0099 -0.0053 ]
[ 0.0171 -0.009 0.0408 -0.0232 ... -0.0122 0.0043 -0.0433 0.0164 ]
[ -0.0029 0.0067 -0.0018 -0.0205 ... -0.017 0.0101 0.0083 -0.0049 ]
[ -0.0028 0.0106 0.0173 0.0014 ... -0.0097 -0.0186 0.0058 -0.0031 ]
[ -0.0089 0.0277 -0.014 -0.0308 ... 0.0001 0.0137 -0.0063 -0.0169 ]
[ -0.0099 -0.0112 -0.0366 -0.0169 ... -0.0178 0.0003 0.0122 -0.0082 ]
[ -0.0049 0.0123 -0.0125 -0.0267 ... -0.0284 -0.0124 -0.0069 -0.0086 ]
[ -0.0206 0.0202 0.0198 0.002 ... 0.0108 0.0031 0.0146 -0.0195 ]
[ 0.0132 -0.004 -0.0075 0.006 ... -0.0001 -0.0082 0.0078 -0.0091 ]
[ -0.0249 0.0234 0.0223 -0.0028 ... 0.0049 -0.0101 0.0240 -0.0184 ]
[ -0.0414 0.0088 -0.0256 0.0013 ... -0.0255 -0.0297 0.0186 0.0312 ]
[ 0.0086 0.0071 -0.0162 -0.0069 ... -0.0312 0.0254 0.0032 -0.0135 ]
[ -0.0015 0.0054 0.0004 -0.0261 ... 0.0167 -0.0128 -0.0093 0.0118 ]
[ 0.0141 -0.0039 0.0078 0.0028 ... -0.0111 -0.0412 -0.0066 -0.0232 ]
[ -0.0002 0.0219 -0.0078 0.0075 ... -0.0385 0.0036 -0.0074 -0.0108 ]
[ -0.0191 -0.003 -0.0061 -0.0098 ... -0.0053 -0.0144 0.0156 0.0071 ]
[ -0.0152 0.0059 -0. -0.0005 ... -0.0059 0.0006 -0.0383 0.018 ]
[ -0.0209 0.0095 -0.0229 -0.0163 ... -0.0211 0.0104 -0.0013 0.006 ]
[ 0.0083 0.0059 -0.0371 -0.0072 ... 0.0056 -0.0122 0.022 -0.0132 ]
[ -0.0393 0.0132 -0.0268 -0.0057 ... 0.0072 -0.0173 0.0238 0.0131 ]
[ -0.0213 -0.0035 0.0078 -0.0224 ... 0.0013 -0.0067 -0.0158 -0.0085 ]
[ -0.0371 -0.019 -0.0137 -0.0019 ... -0.0052 -0.037 0.0164 0.0238 ]
[ 0.0252 -0.017 0.0041 0.0001 ... -0.0062 -0.0307 -0.0372 0.0002 ]
[ -0.0395 0.0227 -0.0322 -0.0335 ... -0.0503 0.0124 -0.021 -0.0249 ]
```



# VECTORIZERS

## Tools to vectorize other types of data

Process command lines  
IP Traffic Flows  
Event logs  
Malware Binaries  
etc...

# Text to token sequences

```
1 %%time
2 cmdlines_vec = vz.BytePairEncodingVectorizer(max_vocab_size=600).fit_transform(cmdlines_u.tolist())
3 cmdlines_vec
```

CPU times: user 8.68 s, sys: 21.1 ms, total: 8.7 s

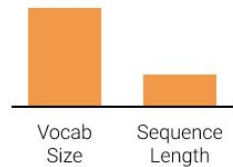
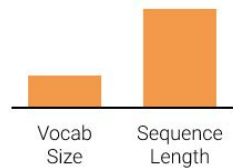
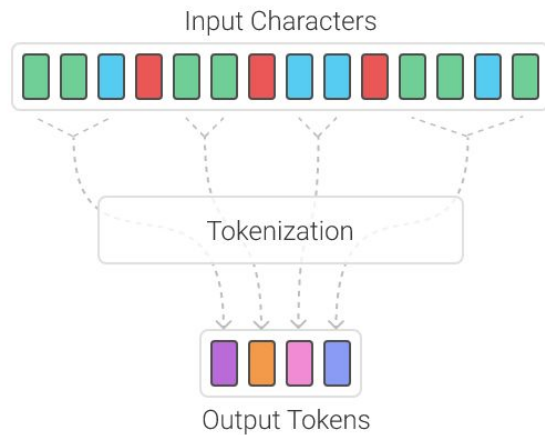
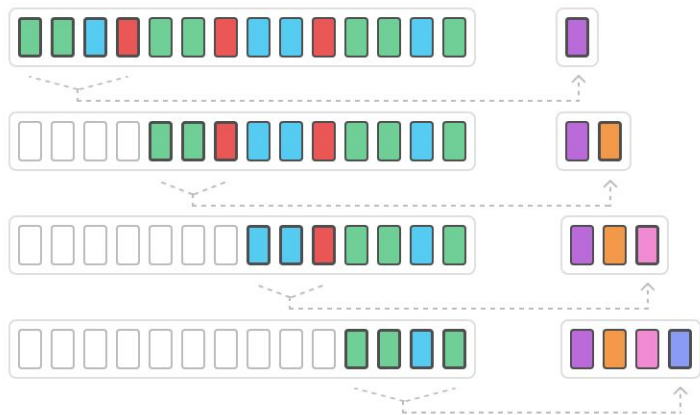
Wall time: 8.66 s

<31046x574 sparse matrix of type '<class 'numpy.float32'>'  
with 823621 stored elements in Compressed Sparse Row format>

```
1 vz_bpe = vz.BytePairEncodingVectorizer(max_vocab_size=600).fit(cmdlines_u.tolist())
2 [longest for longest in sorted(vz_bpe.tokens_, key=len, reverse=True)][75:100]
```

```
['\\v4.0.30319\\mcsorsvw.exe -',
 ' --disable-gpu-compositing',
 ' --enable-main-frame-before',
 'c:\\windows\\microsoft.net\\',
 ' --lang=en-us --service-s',
 ' --time-ticks-at-unix-epo',
 ' --mojo-platform-channel',
 ' --field-trial-handle=19',
 ' --device-scale-factor=1',
 'c:\\program files\\wintap',
 ' --time-ticks-at-unix-e',
 'c:\\windows\\microsoft.n',
 'tevent 0 -ngenprocess ',
 'c:\\program files (x86)',
 ' --field-trial-handle=',
 ' --lang=en-us --servic',
 ' --num-raster-threads=',
 ' --renderer-client-id=',
 ' --disable-gpu-composi',
 '"ngen worker process"',
 'c:\\program files (x86',
 'device-scale-factor=1',
 ' --launch-time-ticks=',
 ' --time-ticks-at-unix',
 ' --enable-main-frame-']
```





# Text to vectors: bag of words

# Parse command lines into tokens

cmdline	tokens	num
"c:\windows\system32\consent.exe" 8364 316 00000225c123a720	["c:\windows\system32\consent.exe", 8364, 316, 00000225c123a720]	4
"c:\windows\syswow64\msiexec.exe" -embedding dabb2a6d532491a1295e2c3d16e9fdfd	["c:\windows\syswow64\msiexec.exe", -embedding, dabb2a6d532491a1295e2c3d16e9fdfd]	3
"c:\windows\system32\consent.exe" 7040 288 0000019ec8c2a540	["c:\windows\system32\consent.exe", 7040, 288, 0000019ec8c2a540]	4
"c:\windows\system32\curl.exe" --help	["c:\windows\system32\curl.exe", --help]	2
"c:\windows\system32\consent.exe" 1640 318 00000204e5430040	["c:\windows\system32\consent.exe", 1640, 318, 00000204e5430040]	4
"c:\windows\system32\svchost.exe" -k wsappx -p -s appxsvc	["c:\windows\system32\svchost.exe", -k, wsappx, -p, -s, appxsvc]	6
"c:\windows\system32\ping.exe" acme-dc1.acme.com	["c:\windows\system32\ping.exe", acme-dc1.acme.com]	2
"c:\windows\system32\sihclient.exe" /cv sq9lrwm140ena5nxthiida.0.1	["c:\windows\system32\sihclient.exe", /cv, sq9lrwm140ena5nxthiida.0.1]	3
"c:\program files\git\usr\bin\ls.exe" -f --color=auto --show-control-chars 1277	["c:\program files\git\usr\bin\ls.exe", -f, --color=auto, --show-control-chars, 1277]	5
"c:\users\user10\downloads\visualstudiosetup.exe"	["c:\users\user10\downloads\visualstudiosetup.exe"]	1
"c:\windows\system32\svchost.exe" -k unistacksvcgroup	["c:\windows\system32\svchost.exe", -k, unistacksvcgroup]	3
"c:\windows\system32\sihclient.exe" /cv cxixvz5jlemri9ydvjhbea.0.1	["c:\windows\system32\sihclient.exe", /cv, cxixvz5jlemri9ydvjhbea.0.1]	3
"c:\windows\system32\systeminfo.exe" /?	["c:\windows\system32\systeminfo.exe", ?]	2
"c:\program files\git\usr\bin\ls.exe" -f --color=auto --show-control-chars 953	["c:\program files\git\usr\bin\ls.exe", -f, --color=auto, --show-control-chars, 953]	5
"c:\windows\system32\wbem\wmic.exe" product list brief	["c:\windows\system32\wbem\wmic.exe", product, list, brief]	4

```
%%time  
vz_ngram = vz.NgramVectorizer().fit(cmdlines_tokenized.tolist())  
cmdlines_tc = vz_ngram._train_matrix.tocsr()  
cmdlines_tc
```

CPU times: user 771 ms, sys: 35.8 ms, total: 807 ms

Wall time: 812 ms

<31029x20887 sparse matrix of type '<class 'numpy.float32'>'  
with 236570 stored elements in Compressed Sparse Row format>

# Bag of words: vectorize by counting tokens

	tokens	"c:\windows\system32\svchost.exe"	-k	-p	-s	appxsvc	unistacksvcgrou	waasmedicsvc	wersvcgroup	wsappx	wusvcs
cmdline											
"c:\windows\system32\svchost.exe" -k unistacksvcgrou		1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
"c:\windows\system32\svchost.exe" -k wersvcgroup		1.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
"c:\windows\system32\svchost.exe" -k wsappx -p -s appxsvc		1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0
"c:\windows\system32\svchost.exe" -k wusvcs -p -s waasmedicsvc		1.0	1.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0

Information weight transform:  
downweight tokens with low  
similarity information contribution

```
cmdlines_iwt = vzt.InformationWeightTransformer().fit_transform(cmdlines_mnom)  
cmdlines_iwt
```

```
<31029x20907 sparse matrix of type '<class 'numpy.float64'>'  
  with 259822 stored elements in Compressed Sparse Row format>
```

## Classical TF-IDF

$$\text{IDF}(t) = \log \left( \frac{|D|}{|\{d \in D : t \in d\}|} \right)$$

## Information Weight

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

where

$$P_t(d) = \frac{f_{t,d}}{\sum_{d \in D} f_{t,d}}$$

$$Q(d) = \frac{|d|}{\sum_{d' \in D} |d'|}$$



Reduce dimensionality  
while preserving local  
similarity structure

```
%%time  
bagofwords_dmap = umap.UMAP(metric="hellinger").fit_transform(cmdlines_vec)  
bagofwords_dmap.shape
```

CPU times: user 1min 50s, sys: 3.08 s, total: 1min 53s

Wall time: 19.1 s

(31029, 2)

# Hellinger distance

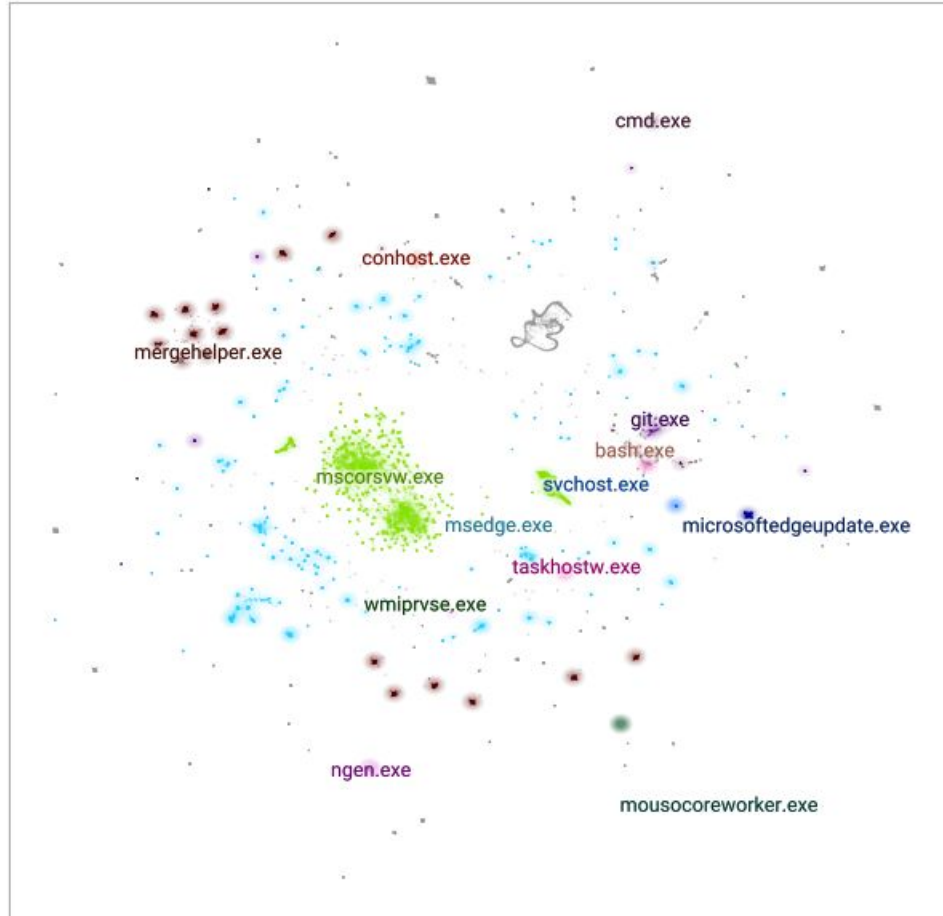
$$d(\mathbf{x}, \mathbf{y}) = \sqrt{1 - \sum_{i=1}^d \sqrt{\frac{x_i y_i}{\|\mathbf{x}\|_1 \|\mathbf{y}\|_1}}}$$

Approximated by  
cosine distance

$$d(\mathbf{x}, \mathbf{y}) = 1 - \sum_{i=1}^d \frac{x_i y_i}{\|\mathbf{x}\|_2 \|\mathbf{y}\|_2}$$

## Process instances

as bags of information-reweighted parsed command line tokens



# Document vectorization strategies

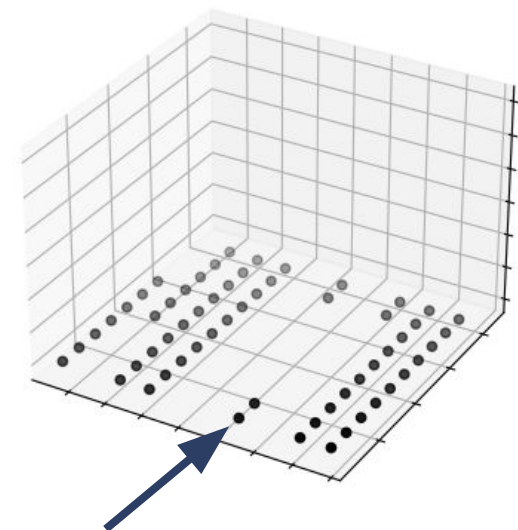
Bag of tokens

Discard order,  
count how  
often each  
word occurs

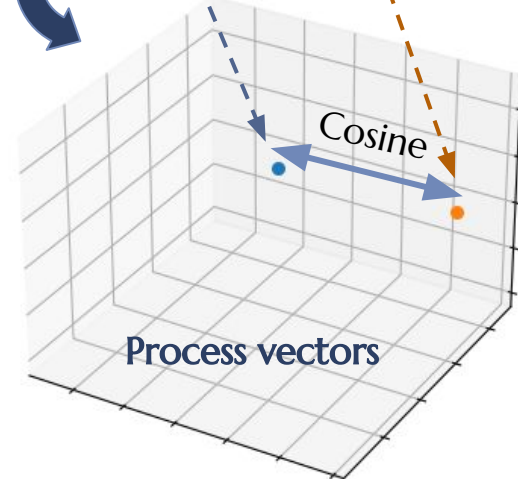
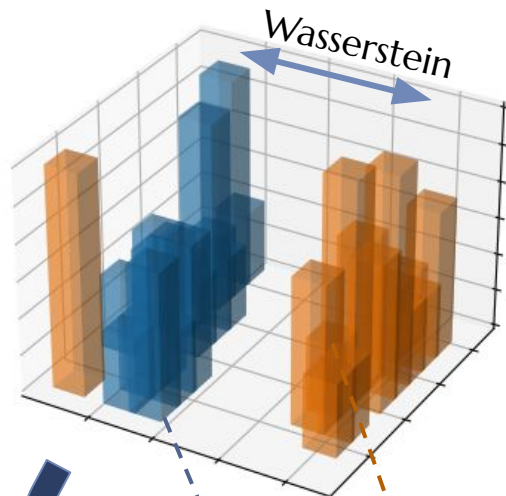
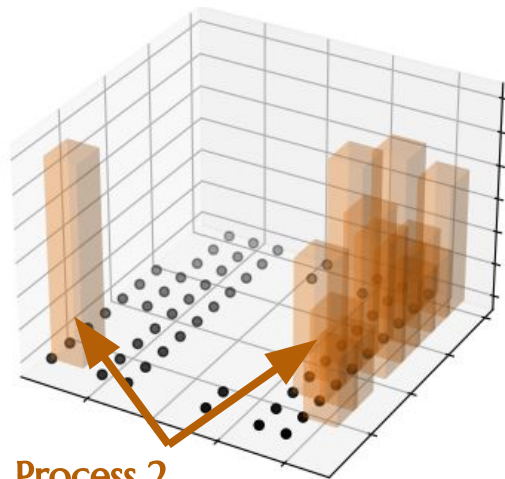
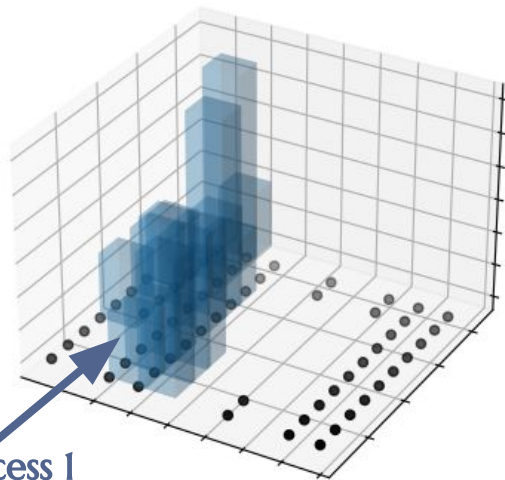
Sequence  
similarities?

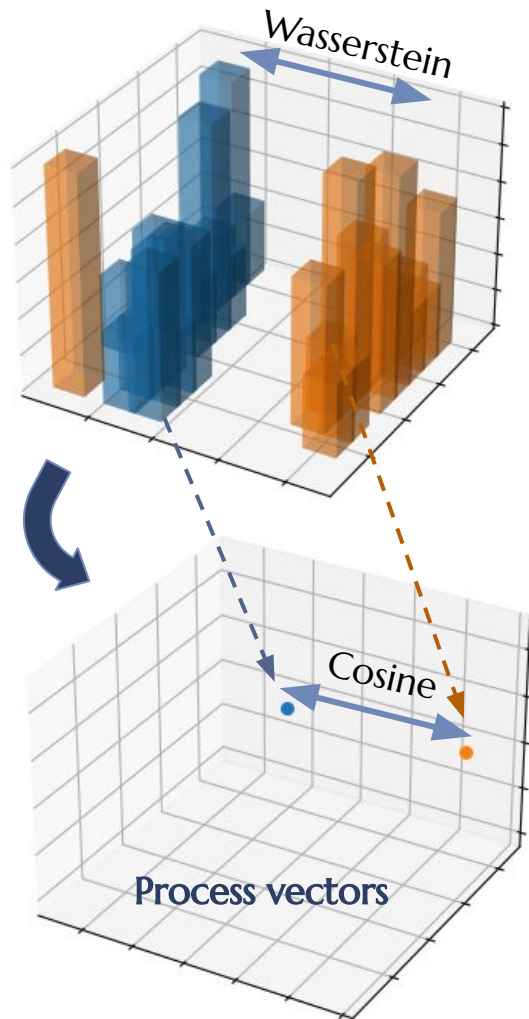
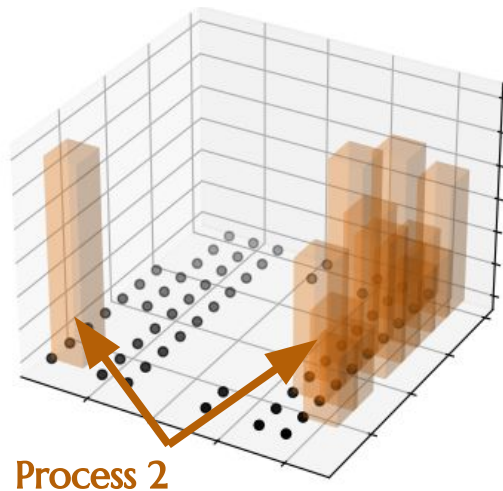
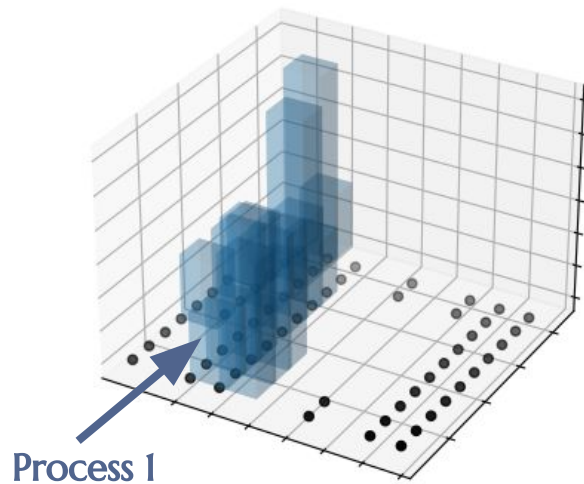
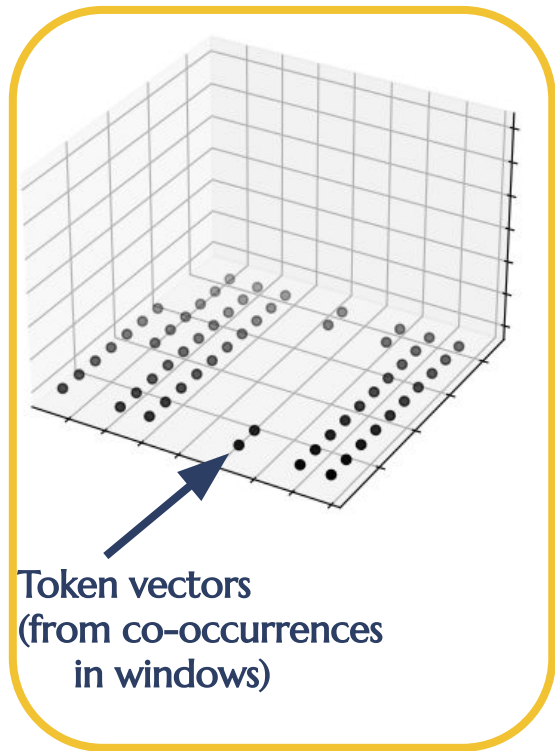
Optimal  
transport on  
**token vector**  
distributions

# Tokens to vectors: distributions on point clouds



Process 1







```
%%time  
vz_cooc = vz.TokenCooccurrenceVectorizer(n_threads=os.cpu_count(), n_iter=3)\  
    .fit(cmdlines_tokenized.tolist())  
cooc_vec = vz_cooc.reduce_dimension(512)  
cooc_vec.shape
```

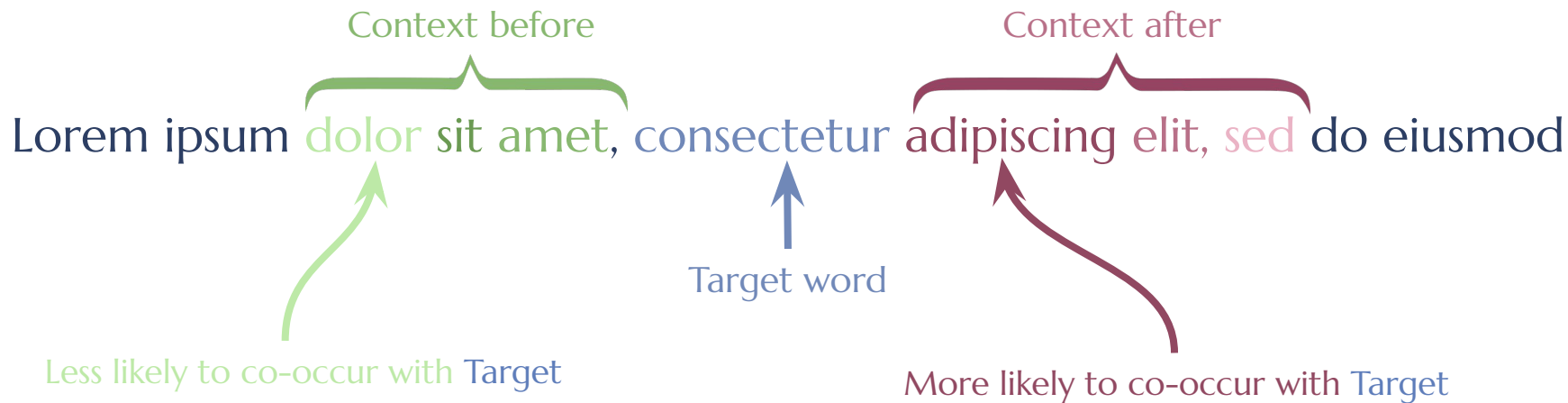
CPU times: user 2min 1s, sys: 7.18 s, total: 2min 8s

Wall time: 25.9 s

(20887, 512)

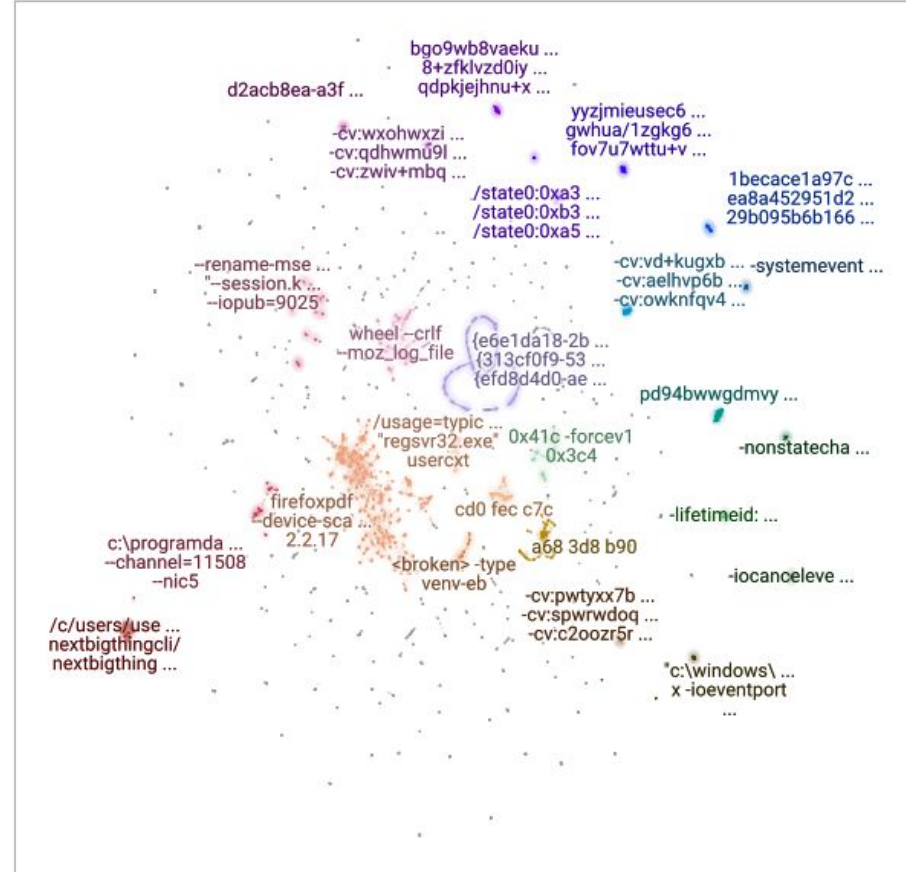
# Tokens aren't independent

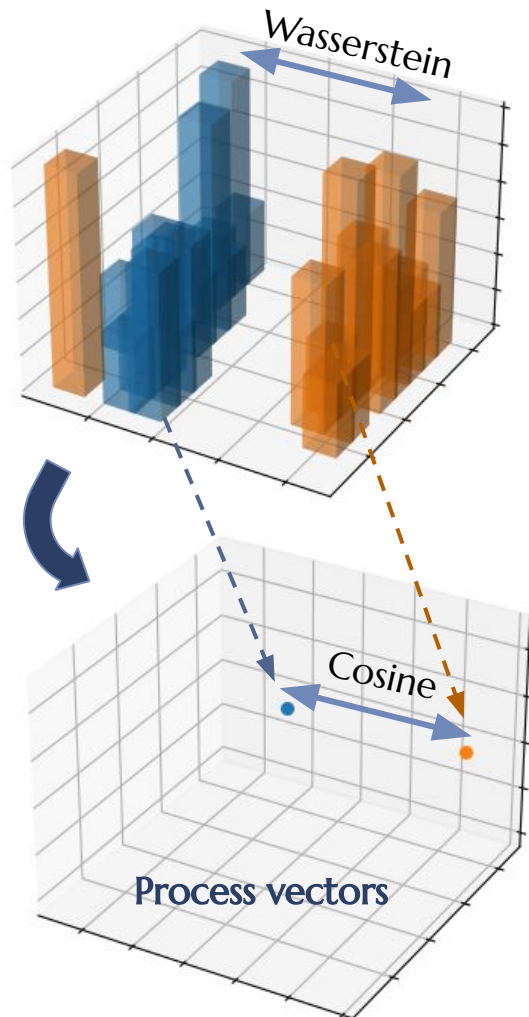
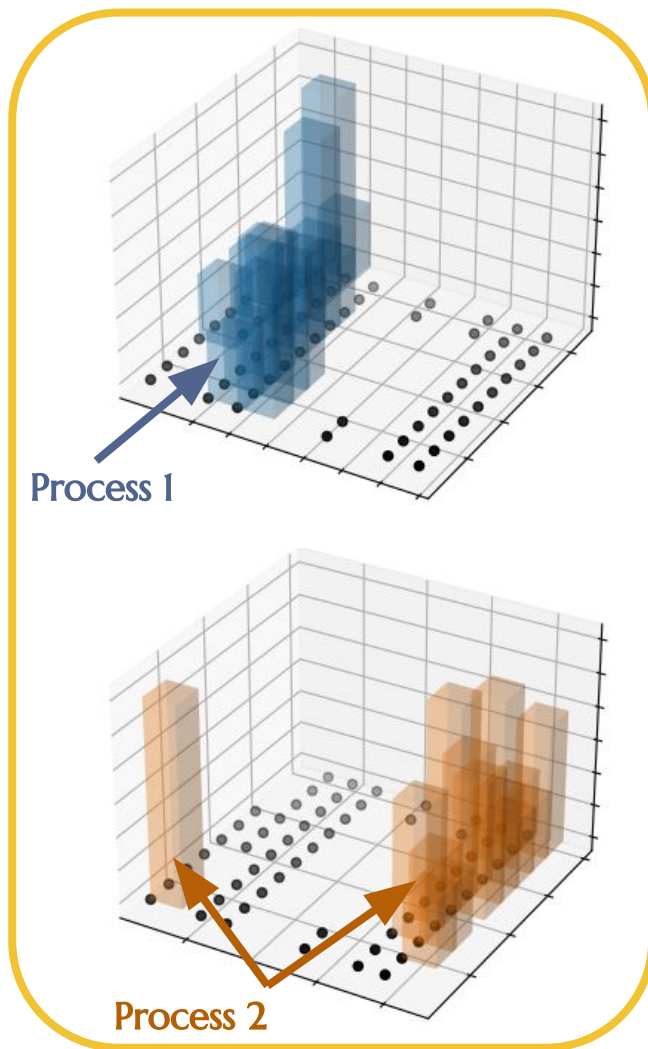
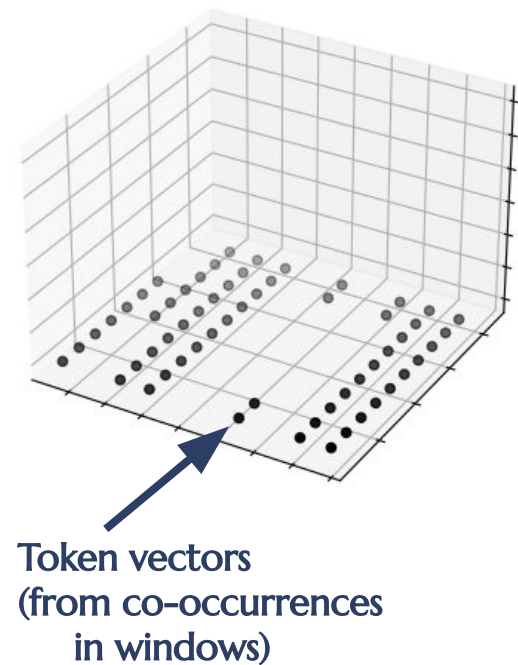
Count local cooccurrences  
of tokens to generate  
vector representation of  
tokens



## Token vectors

composed from cooccurrence with other tokens across command lines





# Wasserstein vectorization

## Via linear optimal transport

WASSERSTEIN EMBEDDING FOR GRAPH LEARNING

Soheil Kolouri\* †, Navid Naderializadeh\*†, Gustavo K. Rohde‡ , & Heiko Hoffmann†, 2021

```
%%time  
cmdlines_wass = vz.WassersteinVectorizer().fit_transform(cmdlines_iwt, vectors=cooc_vec)  
cmdlines_wass.shape
```

CPU times: user 1min 33s, sys: 5.9 s, total: 1min 39s

Wall time: 17 s

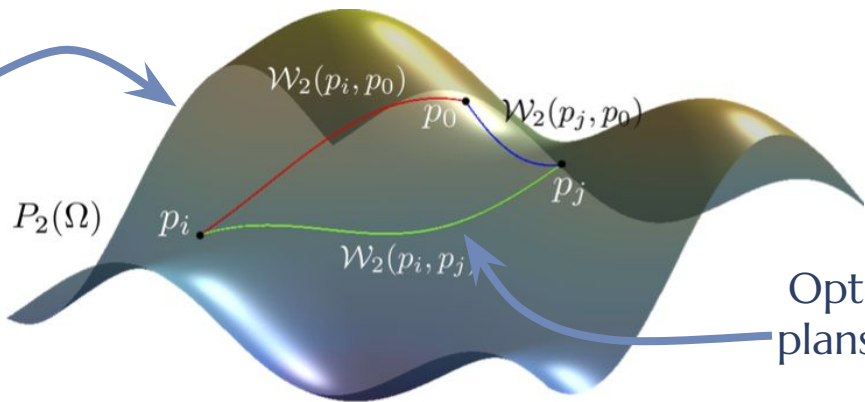
(31029, 128)



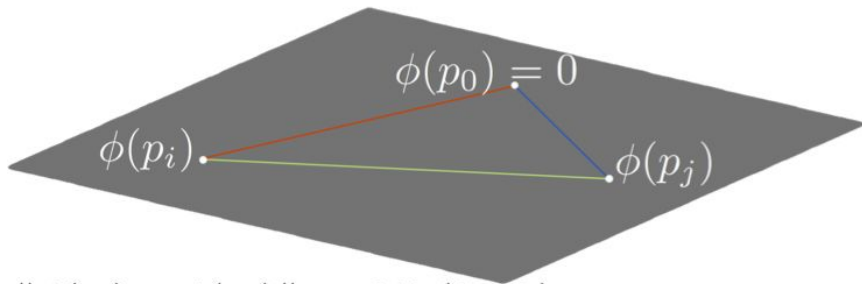
Each process is  
a distribution  
over a  
point cloud  
(of token cooccurrence vectors)

# Theory

Manifold of  
probability  
distributions



Optimal transport  
plans are geodesics

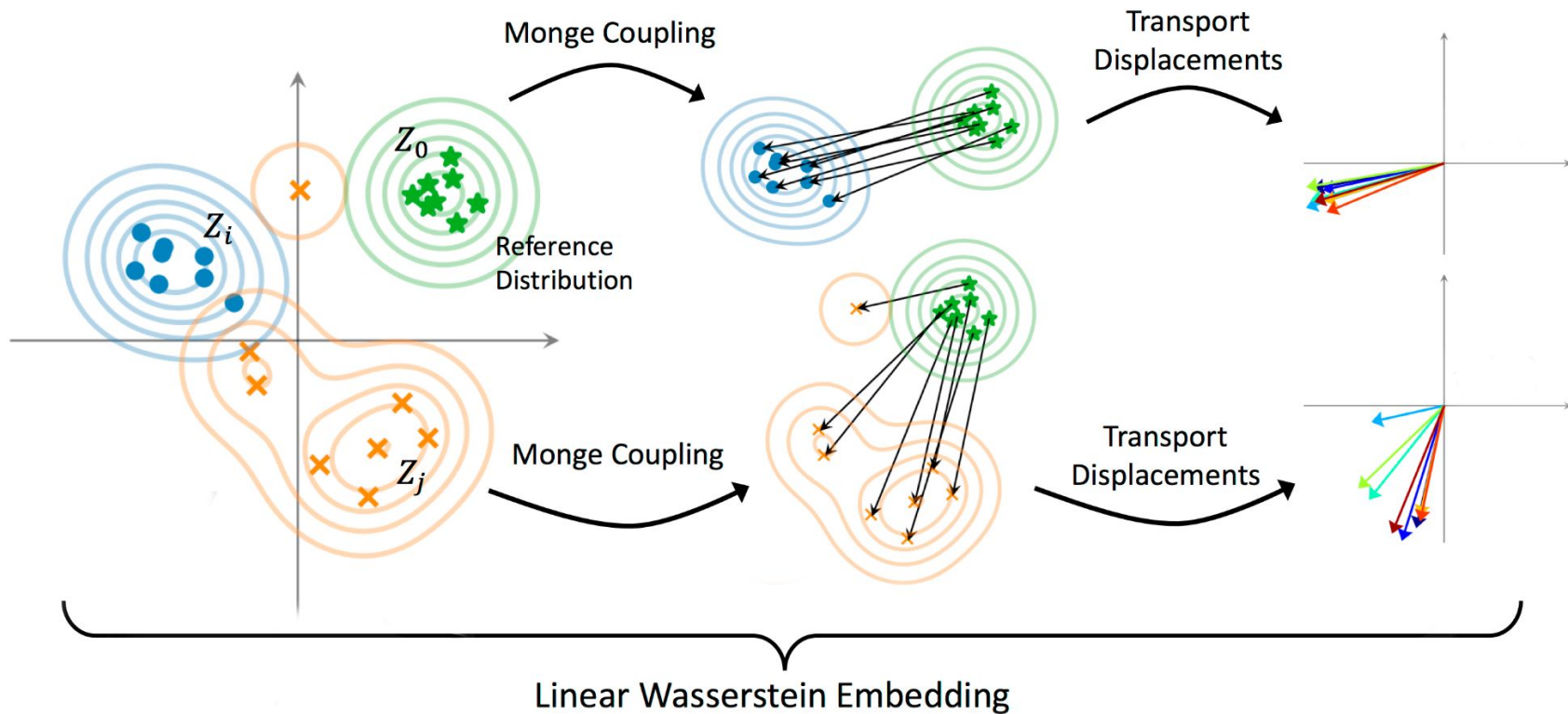


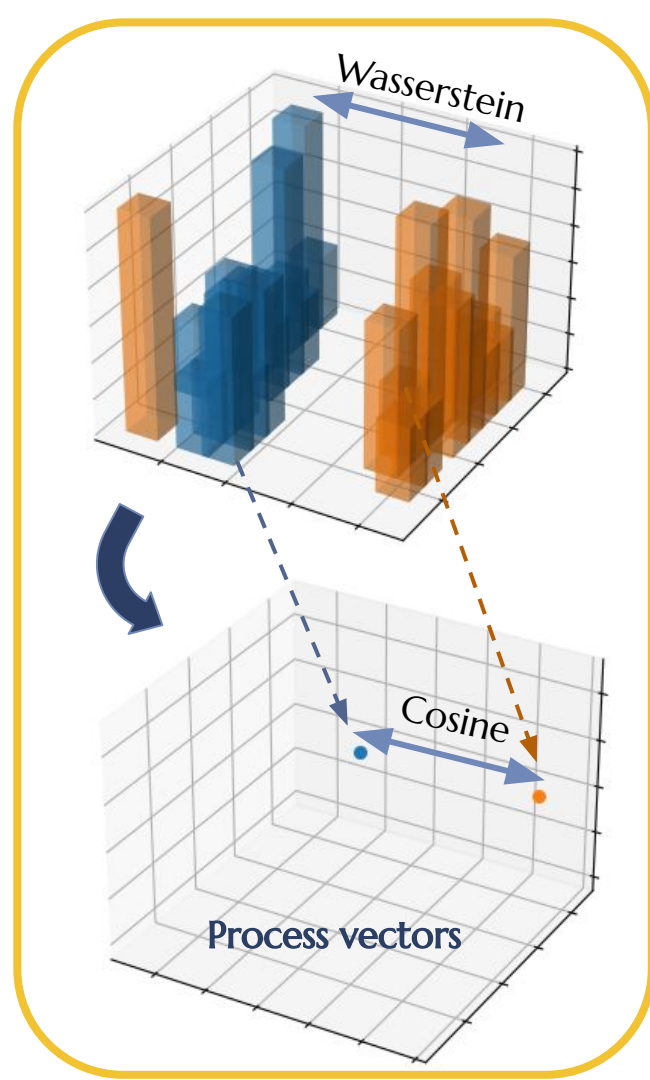
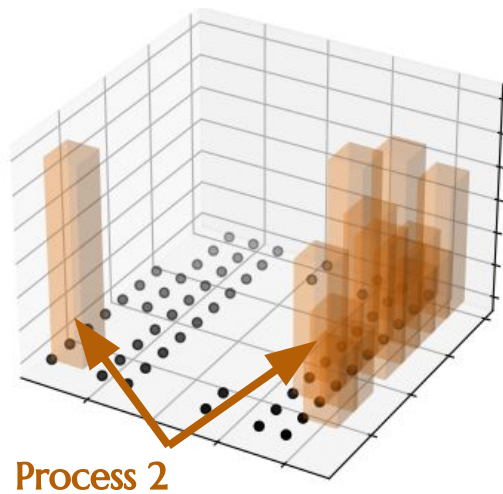
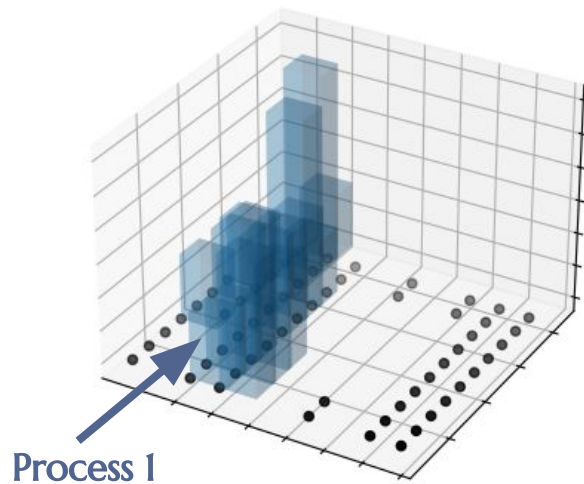
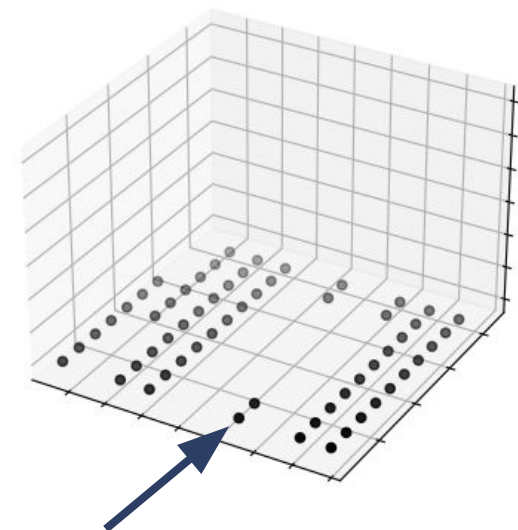
Tangent space of  
the manifold at  $p_0$

$$\|\phi(p_i) - \phi(p_j)\|_2 \approx \mathcal{W}_2(I_i, I_j)$$

$$\|\phi(p_i) - \phi(p_0)\|_2 = \|\phi(p_i)\|_2 = \mathcal{W}_2(I_0, I_i)$$

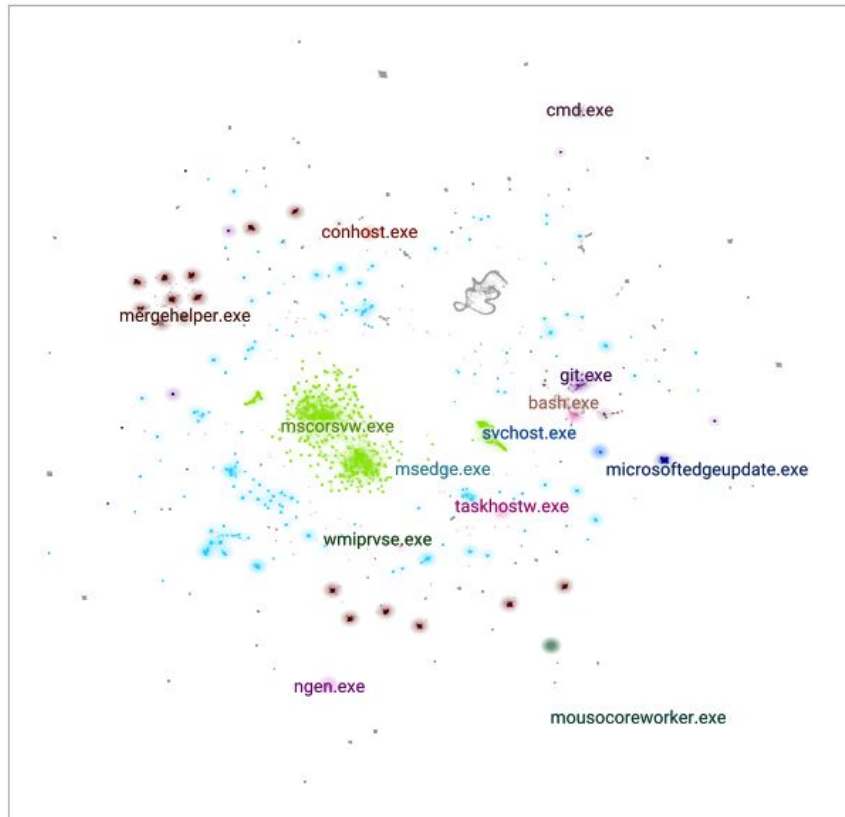
# Practice





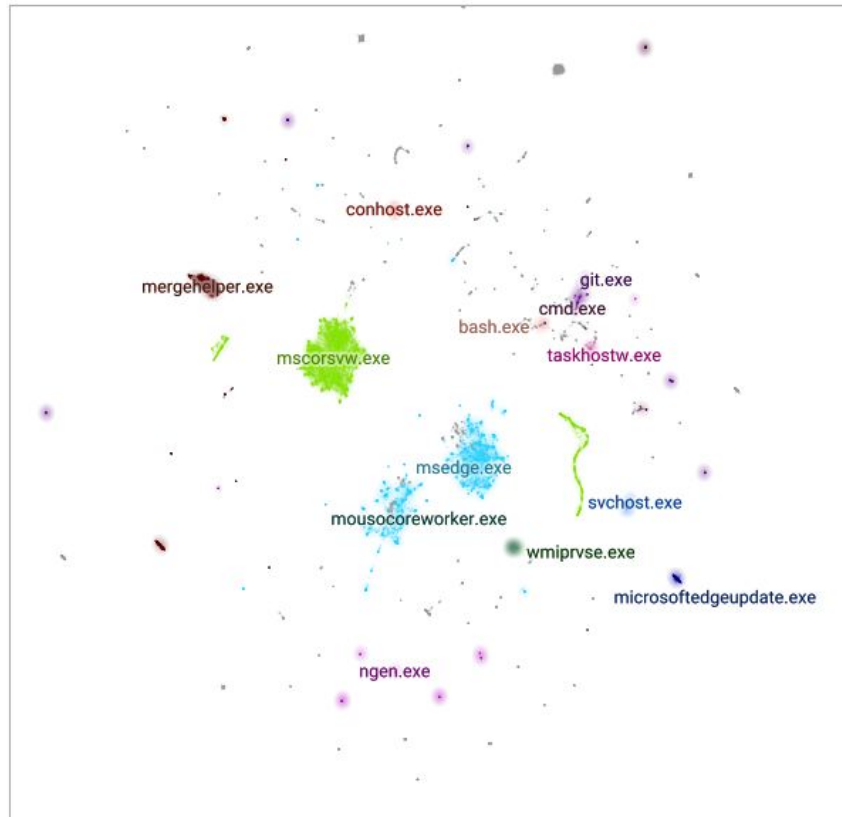
## Process instances

as bags of information-reweighted parsed command line tokens



## Process instances

as distributions over a cloud of command line token cooccurrence vectors





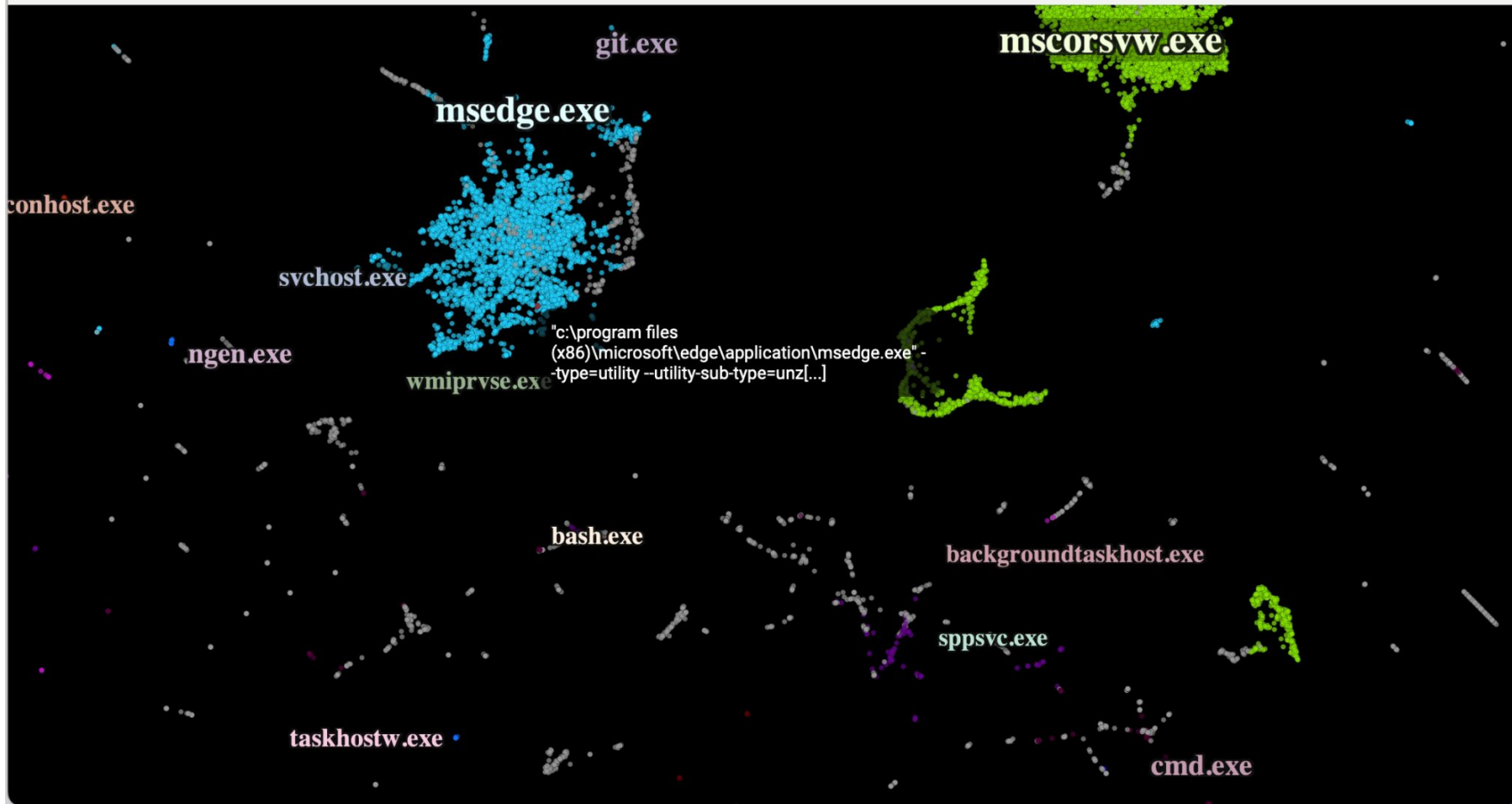


# DATA MAP PLOT

Easily build attractive  
static and simple  
interactive maps

```
plot_interactive = dmp.create_interactive_plot(  
    datamap|,  
    metadata_cmdlines["label"],  
    hover_text=metadata_cmdlines["hover_text"],  
    darkmode=True,  
    label_color_map=label_color_map,  
)  
plot_interactive.save("datamap.html")
```







# Easily build interactive map web applications

Built on top of the Panel library



# Panel












```

plot = tnt.BokehPlotPane(
    xy,
    labels=metadata_processes_short["description_topK"],
    hover_text=metadata_processes_short["description"],
    width=900,
    height=900,
    show_legend=False
)
summ_common = tnt.DataSummaryPane(FeatureCommonSummarizer())
summ_common.link_to_plot(plot)
summ_ts = tnt.PlotSummaryPane(tnt.summary.plot.TimeSeriesSummarizer(
    metadata_processes_short.assign(ones=1.),
    time_column="time_first",
    count_column="ones",
    freq="12H"
))
summ_ts.link_to_plot(plot)
editor = tnt.LabelEditorWidget(labels=metadata_processes_short["description_topK"])
editor.link_to_plot(plot)
pn.Row(editor, plot, pn.Column(summ_ts, summ_common, height=900))

```

[73]:

## Label Editor

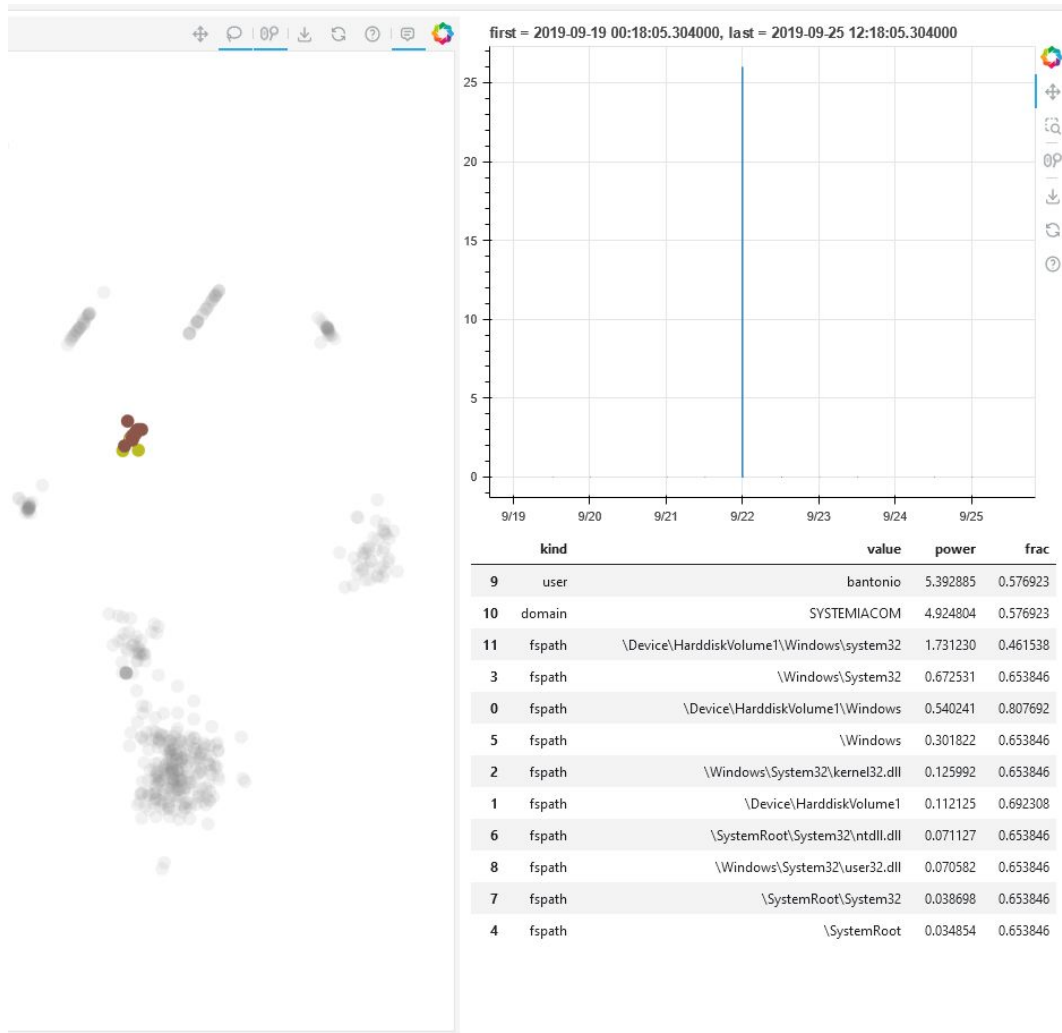
	cmd.exe /c netstat -n
	C:\Windows\system32\cmd.exe /c
	tasklist
	C:\Windows\SYSTEM32\cmd.exe /i
	\Device\HarddiskVolume1\Window
	\Device\HarddiskVolume1\Window
	NETSTAT.EXE
	\\?\C:\Windows\system32\conhost
	(other)
	netstat -n
	ping -w 2000 -n 2 127.0.0.1

New Label














Nothing to summarize.

Nothing to summarize



### Label Editor

	<code>cmd.exe /c netstat -n</code>
	<code>C:\Windows\system32\cmd.exe /c</code>
	<code>tasklist</code>
	<code>C:\Windows\SYSTEM32\cmd.exe /i</code>
	<code>\Device\HarddiskVolume1\Window</code>
	<code>\Device\HarddiskVolume1\Window</code>
	<code>NETSTAT.EXE</code>
	<code>\\?\C:\Windows\system32\conhost</code>
	<code>(other)</code>
	<code>netstat -n</code>
	<code>ping -w 2000 -n 2 127.0.0.1</code>
	<code>blah</code>

New Label





Data to  
Vectors



Manifold  
Learning



Interactive  
Visualization



and



Supervised  
Cluster  
Inference





Data to  
Vectors

Manifold  
Learning

Interactive  
Visualization

Supervised  
Cluster  
Inference

 VECTORIZERS

 UNIFORM MANIFOLD  
**UMAP**  
APPROXIMATION & PROJECTION

 **TNT**  
THIS NOT THAT

and

 DATAMAPLOT

 scikit  
*learn*

Data to  
Vectors

Manifold  
Learning

Interactive  
Visualization

Supervised  
Cluster  
Inference

 VECTORIZERS

 UNIFORM MANIFOLD  
**UMAP**  
APPROXIMATION & PROJECTION

 **TNT**  
THIS NOT THAT

and

 DATAMAPLOT

 scikit  
*learn*

Data to  
Vectors



Manifold  
Learning



Interactive  
Visualization



and



Supervised  
Cluster  
Inference



```
import vectorizers as vz, vectorizers.transformers as vzt, umap, fast_hdbscan, datamapplot













token_seqs = # Tokenize your data, yielding a list of lists of tokens.
vz_cooc = vz.TokenCooccurrenceVectorizer().fit(token_seqs)
vecs_cooc = vz_cooc.reduce_dimension(DIM_COOC) # Unsure? Use 128 🙋

token_counts = vz.NgramVectorizer().fit_transform(token_seqs)
distrib_iwt = vzt.InformationWeightTransformer().fit_transform(token_counts)
vecs = vz.WassersteinVectorizer().fit_transform(distrib_iwt, vectors=vecs_cooc)

datamap = umap.UMAP(metric="cosine").fit_transform(vecs)
labels = # Find clusters in data map and figure names for them.
datamapplot.create_interactive_plot(datamap, labels, ...)
```

The Tutte Institute for Mathematics and Computing (TIMC) is a government research institute focused on fundamental mathematics and computer science. Research work from the Institute that has been released as open source can be found here.

## Software from the Tutte Institute and its staff

 <p><a href="#">Vectorizers</a></p> <p>Tools for vectorizing sequence data</p>	 <p><a href="#">UMAP</a></p> <p>Dimension reduction and visualization</p>	 <p><a href="#">HDBSCAN</a></p> <p>Spatial clustering</p>	 <p><a href="#">PyNNDescent</a></p> <p>Approximate nearest neighbour search</p>	 <p><a href="#">ThisNotThat</a></p> <p>Interactive data map exploration and labeling</p>
 <p><a href="#">Ensemble Clustering for Graphs</a></p> <p>Graph clustering and community detection</p>	 <p><a href="#">Graph Partition Measures</a></p> <p>Graph partition evaluation</p>	 <p><a href="#">EnsTop</a></p> <p>Ensemble topic modeling</p>	 <p><a href="#">EasyData</a></p> <p>Data science reproducibility framework</p>	
 <p><a href="#">Glasbey</a></p> <p>Glasbey Algorithmic Categorical Colour Palettes</p>	 <p><a href="#">Fast HDBSCAN</a></p> <p>Fast Multicore HDBSCAN in Numba</p>	 <p><a href="#">DataMapPlot</a></p> <p>Presentation Ready Data Map Plots</p>		



<https://github.com/TutteInstitute/acme3-mapping>