A text mining case study on document clustering and topic extraction

Bag of Words

Representation of a corpus of documents in two files

vocabulary

Aardvark
Abacus
…
…
…
…
…
…
…
Zucchini

Document-word-frequencies

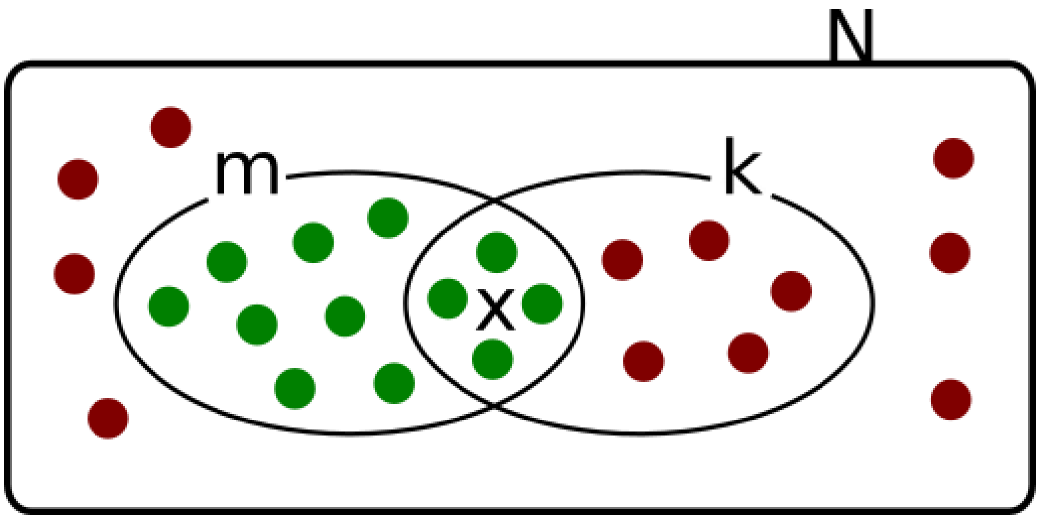
Doc Idx	Word Idx	Count
1	126	3
1	133	6
…		
…		
…		
…		
…		
…		
…		

DTM: Document-Term Matrix

Sparse matrix: most values are 0

	aah	aaheed	aaron	abacus	...	zucchini
Document 1	0	0	0	0	...	0
Document 2	0	0	0	1		1
...						
Document n	0	0	0	2	...	0

Term Enrichment Analysis and Hypergeometric Distribution for significant Topic Extraction



The hypergeometric test can be applied to associate

N = Total number of terms in corpus
m = counts of one particular term in corpus
k = terms in one particular group of documents (one cluster from K-Means)
x = counts of one particular term in cluster

What is the probability that the term has been observed x times in the cluster?

$$P(x \mid N, m, k) = \frac{\binom{m}{x}\binom{N-m}{k-x}}{\binom{N}{k}}$$

Weighting of the DTM

Absolute frequencies (counts) introduce a bias:

Larger documents will have larger values

TF-IDF : Term Frequency Inverse Document Frequency

Improves clustering and predictive performance by accounting for:

- Relative frequency of term across corpus
- Length of document

$$idf_t = \log_2\left(\frac{N}{df_t}\right)$$

N: Number of documents in corpus
df_t: Number of documents containing term *t*

Downweight terms appearing in many documents
multiplying the original DTM frequency by *idf*

$$tfidf_t = tf_{(t,d)} \times idf_t$$

SVD: Singular Value Decomposition

Dimensionality reduction:

DTM → Dense matrix with many fewer columns

The new (orthogonal) columns are linear combinations of columns in original DTM

Preserve as much as the variance structure of original DTM as possible

$$X' \approx UDV^T$$

U: Dense *d* by *s* (*d*: num documents, *s*: num reduced dimensions)

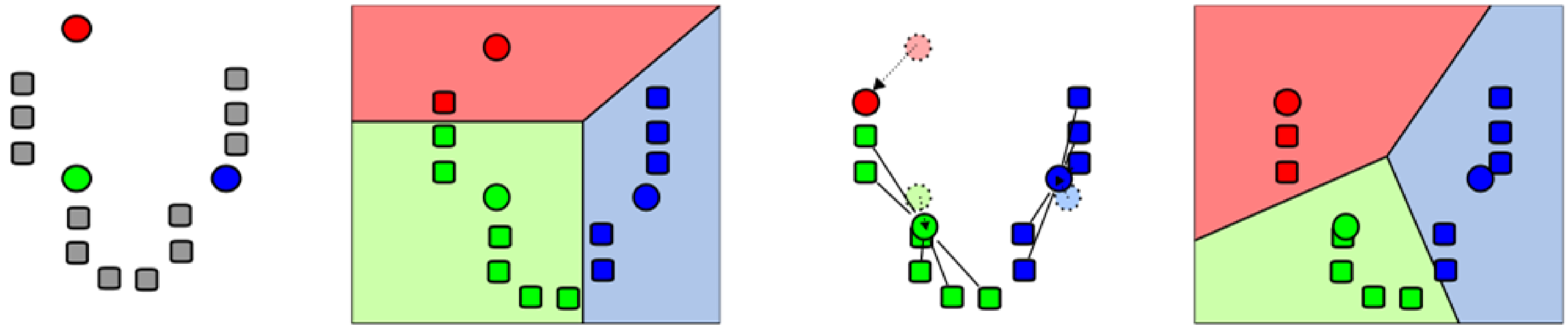
Dim-reduced representation of documents

D: Diagonal matrix with nonnegative entries

V: Dense *w* by *s* (*w*: num terms)

Dim-reduced representation of terms

The K-Means clustering algorithm



1 set ***k*** initial “means”

2 assign each data point to its nearest mean

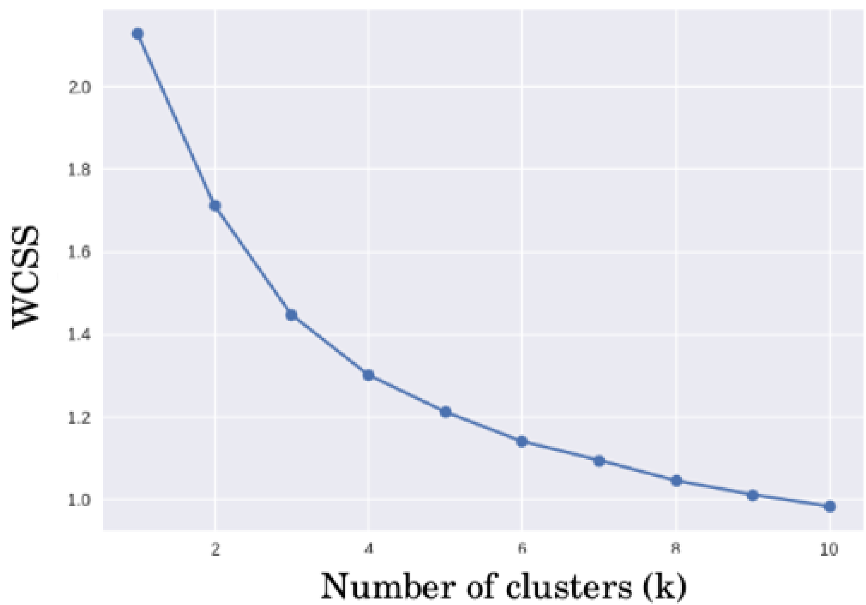
3 the centroid of each of the k clusters becomes the new mean

4 Iterate over steps 2 and 3 until convergence (the distance the new centroids are moved is below a threshold)

The centroids are chosen so that WCSS Within Cluster Sum of Squares aka Inertia is minimized in every cluster

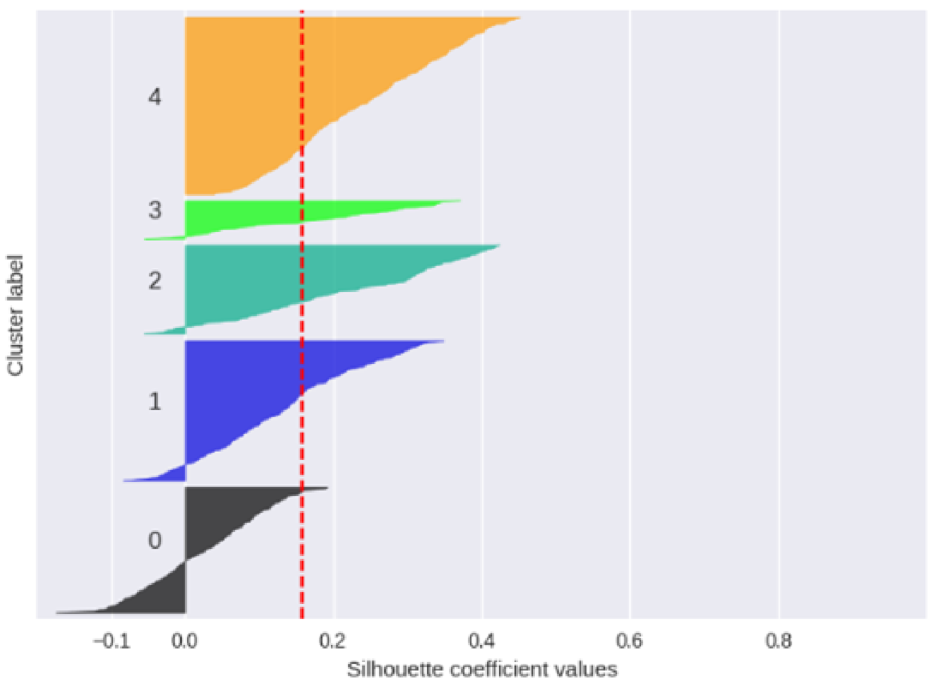
Finding an optimal number of clusters for K-Means

The “elbow” method



Find a point where the rate of decrease in WCSS sharply shifts

Silhouette plots



Silhouette plots

$$s=\frac{b-a}{max(a,b)}$$

a: mean distance between a sample and all other points in the same class
b: mean distance between a sample and all other points in the next nearest cluster

KOS Blog entries

A group blog and internet forum focused on liberal American politics

orig source: dailykos.com

D = 3430
W = 6906
N = 353160

KOS dataset

SVD/LSA

```
#kos W/5 -> 83%
n_components = W/5

svd = TruncatedSVD(n_components)
# DTM_tfidf results are normalized. Since LSA/SVD results are
# not normalized, we have to redo the normalization:
normalizer = Normalizer(copy=False)
lsa = make_pipeline(svd, normalizer)

%time \
DTM_tfidf_lsa = lsa.fit_transform(DTM_tfidf)

print("Explained variance of the SVD step: {}%".format(int(svd.explained_variance_ratio_.sum() * 100)))
```

```
CPU times: user 32.2 s, sys: 13.9 s, total: 46.1 s
Wall time: 10.4 s
Explained variance of the SVD step: 83%
```

For the kos dataset, LSA (n_components = W/5 explaining 83% of the original variance) reduces the size of the DTM from shape (3430, 6906) to (3430, 1381)

$$X' \approx UDV^T$$

U: Dense *d* by *s* (*d*: num documents, *s*: num reduced dimensions)
Dim-reduced representation of documents

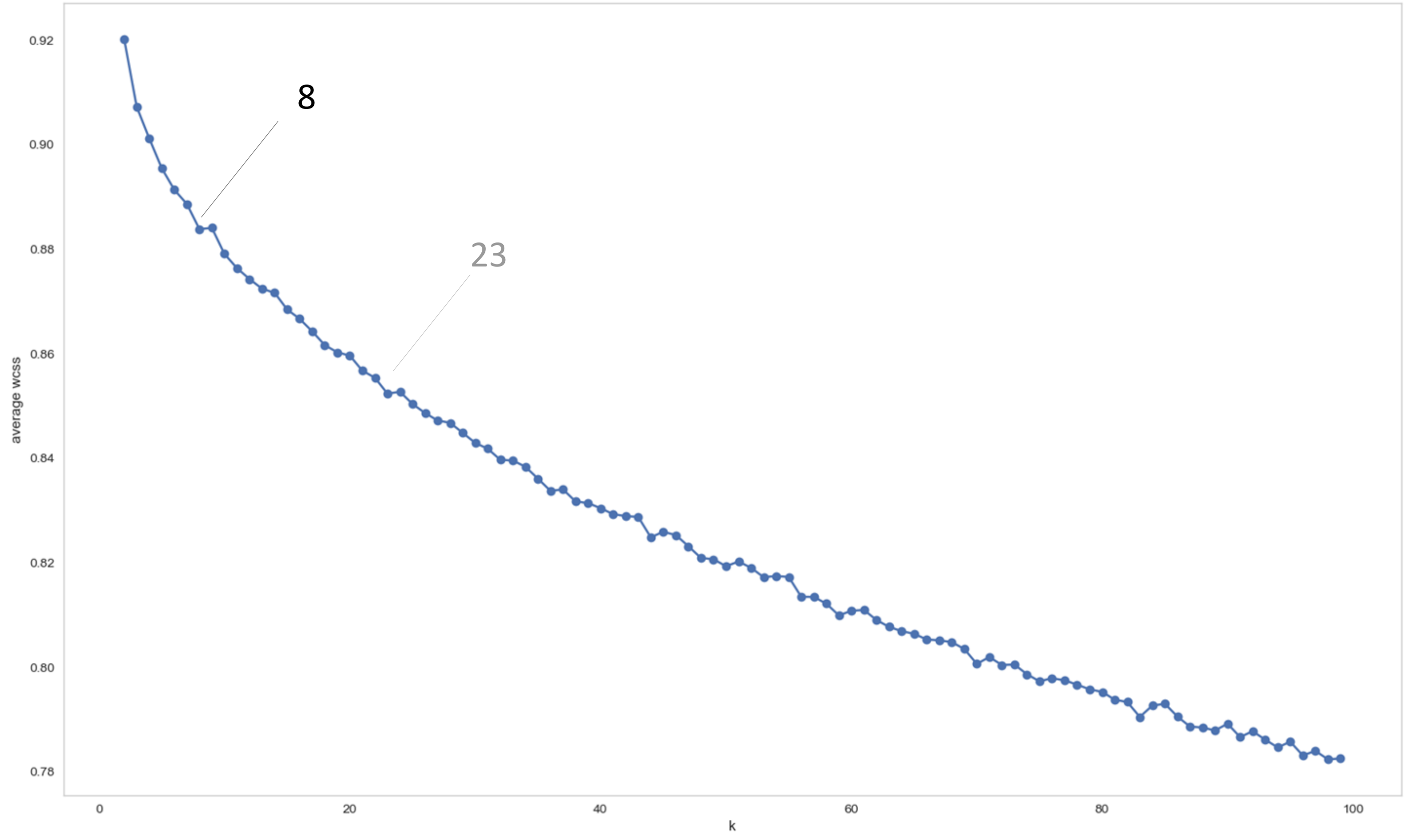
D: Diagonal matrix with nonnegative entries

V: Dense *w* by *s* (*w*: num terms)
Dim-reduced representation of terms

KOS dataset

Estimate k for KMeans

range(k) = [2,100]



'Elbow' method does not reveal an optimal k.

Clusters might not show a real good separation

Topic inter-relation

KOS dataset

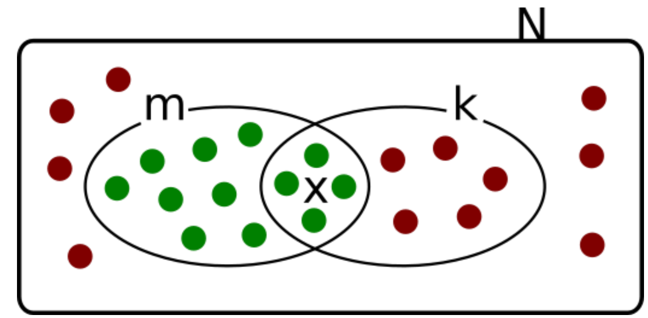
K-Means k= 8

Cluster 0	iraq	war	bush	iraqi	military
Cluster 1	dean	clark	edwards	kerry	lieberman
Cluster 2	party	nader	democratic	ballot	state
Cluster 3	bush	administration	president	kerry	general
Cluster 4	november	account	electoral	turnout	governor
Cluster 5	kerry	bush	poll	percent	voters
Cluster 6	november	voting	account	electoral	governor
Cluster 7	house	senate	race	elections	district

Choose smaller values for k for
Broadly/loosely defined topics

KOS dataset

K-Means Cluster 1

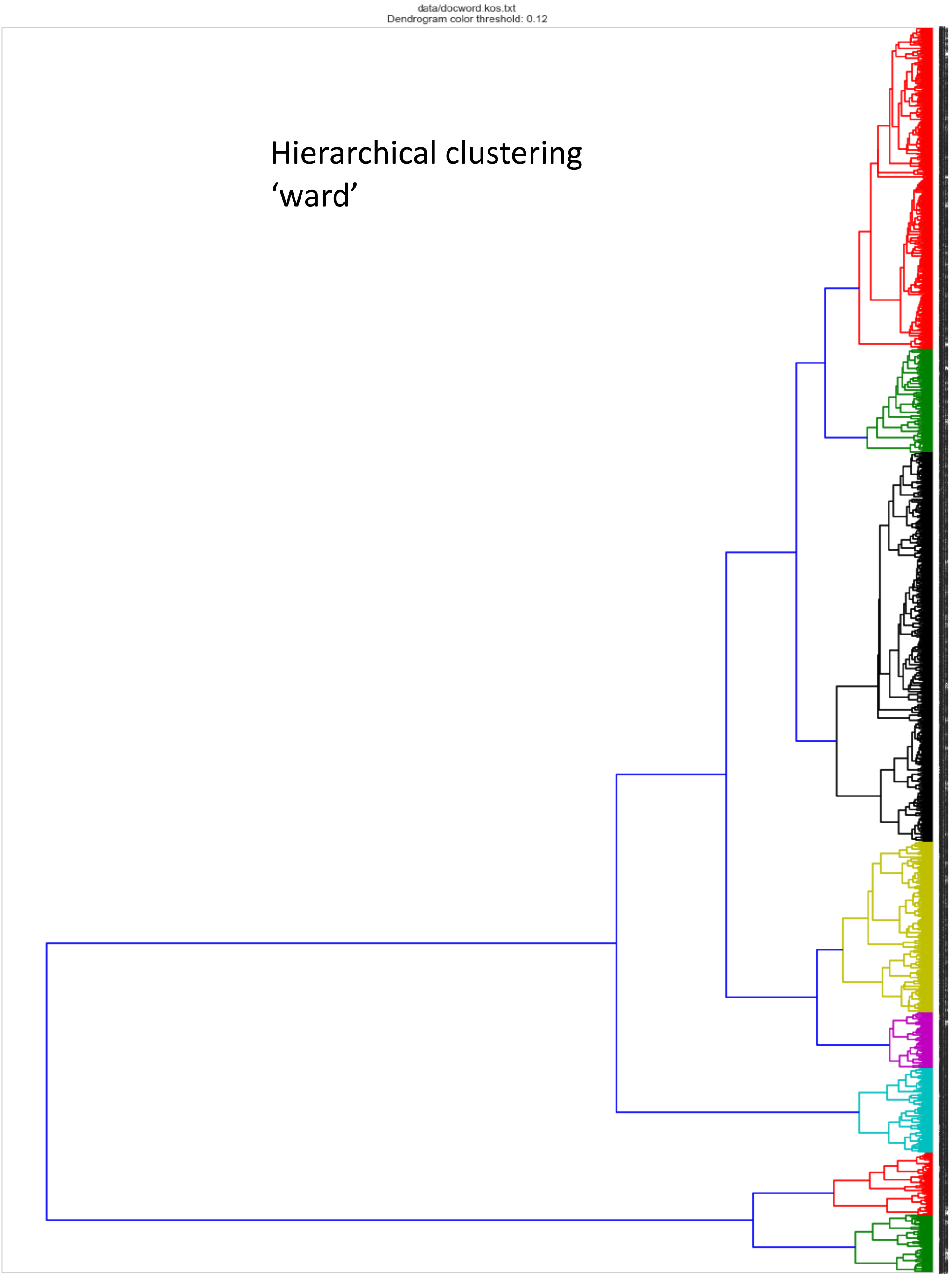


		x	k	m	N		
	word	**word_count_in_cluster**	**n_words_in_cluster**	**word_count_in_corpus**	**n_words_in_corpus**	**p-val**	**adj_pval(BH)**
2119	lieb	14	23087	14	353160	0.000000e+00	0.000000e+00
645	clark	625	23087	820	353160	0.000000e+00	0.000000e+00
942	dean	1278	23087	1798	353160	0.000000e+00	0.000000e+00
1193	edwards	681	23087	1127	353160	0.000000e+00	0.000000e+00
2120	lieberman	354	23087	459	353160	4.702023e-320	3.903619e-317
1589	gephardt	367	23087	530	353160	2.941865e-302	2.035280e-299
1935	iowa	351	23087	597	353160	8.080964e-252	4.792012e-249
2821	primary	520	23087	1528	353160	4.658350e-225	2.417102e-222
2015	kerry	886	23087	4679	353160	3.503594e-181	1.615935e-178
2038	kucinich	165	23087	214	353160	1.021201e-150	4.239004e-148

What is the probability to observe exactly x marked elements in the sample?

$$P(x \mid N, m, k) = \frac{\binom{m}{x}\binom{N-m}{k-x}}{\binom{N}{k}}$$

KOS dataset


data/docword.kos.txt
Dendrogram color threshold: 0.12
Hierarchical clustering
'ward'

KOS dataset

TSNE

t-distributed Stochastic Neighbor Embedding.

From Sklearn:
t-SNE is a tool to visualize high-dimensional data.
It converts similarities between data points
to joint probabilities and tries to minimize
the Kullback-Leibler divergence between the
joint probabilities of the low-dimensional embedding and the high-dim
t-SNE has a cost function that is not convex, i.e. with different initializat

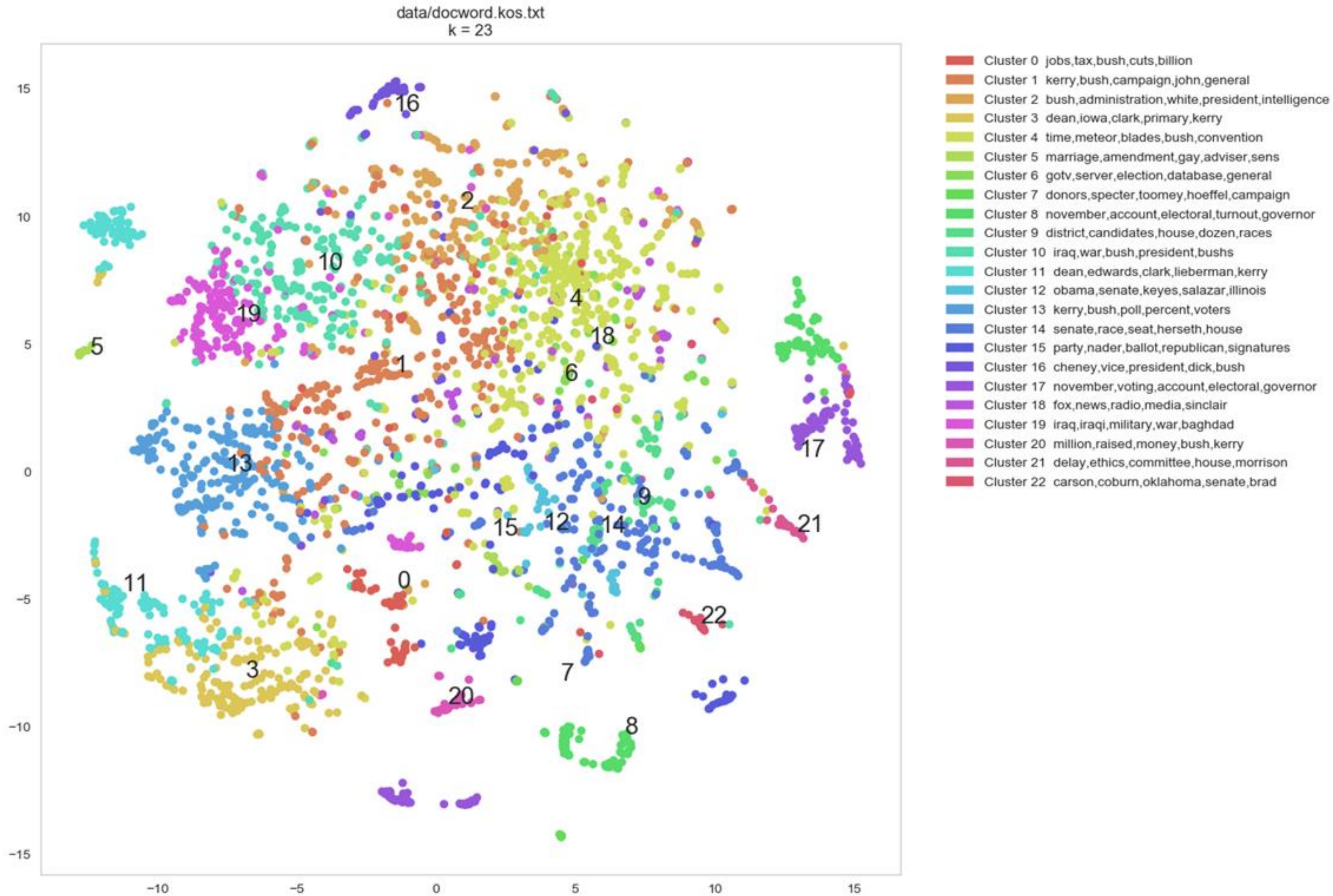
KOS dataset
K-Means k= 23

Cluster 0	jobs	tax	bush	cuts	billion
Cluster 1	kerry	bush	campaign	john	general
Cluster 2	bush	administration	white	president	intelligence
Cluster 3	dean	iowa	clark	primary	kerry
Cluster 4	time	meteor	blades	bush	convention
Cluster 5	marriage	amendment	gay	adviser	sens
Cluster 6	gotv	server	election	database	general
Cluster 7	donors	specter	toomey	hoeffel	campaign
Cluster 8	november	account	electoral	turnout	governor
Cluster 9	district	candidates	house	dozen	races
Cluster 10	iraq	war	bush	president	bushs
Cluster 11	dean	edwards	clark	lieberman	kerry
Cluster 12	obama	senate	keyes	salazar	illinois
Cluster 13	kerry	bush	poll	percent	voters
Cluster 14	senate	race	seat	herseth	house
Cluster 15	party	nader	ballot	republican	signatures
Cluster 16	cheney	vice	president	dick	bush
Cluster 17	november	voting	account	electoral	governor
Cluster 18	fox	news	radio	media	sinclair
Cluster 19	iraq	iraqi	military	war	baghdad
Cluster 20	million	raised	money	bush	kerry
Cluster 21	delay	ethics	committee	house	morrison
Cluster 22	carson	coburn	oklahoma	senate	brad

Choose ***larger*** values for k for more granular topics

KOS dataset

TSNE



NIPS full papers

Neural Information Processing Systems Conference

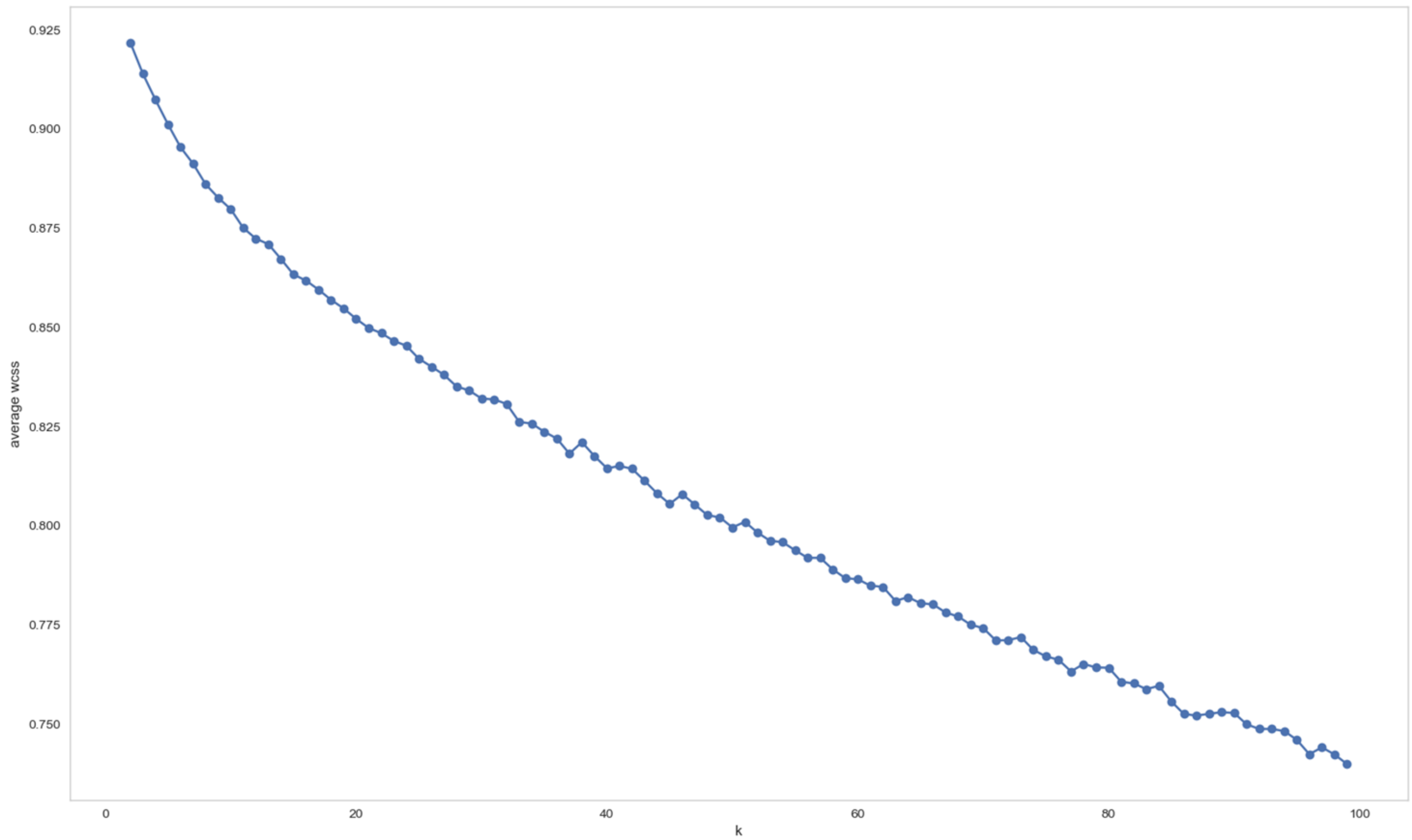
orig source: books.nips.cc

D = 1500
W = 12419
N = 1900000 (approx)

NIPS dataset

Estimate k for KMeans

range(k) = [2,100]



NIPS dataset

SVD/LSA

```
#nips W/20 -> 76%
n_components = W/20

svd = TruncatedSVD(n_components)
# DTM_tfidf results are normalized. Since LSA/SVD results are
# not normalized, we have to redo the normalization:
normalizer = Normalizer(copy=False)
lsa = make_pipeline(svd, normalizer)

%time \
DTM_tfidf_lsa = lsa.fit_transform(DTM_tfidf)

print("Explained variance of the SVD step: {}%".format(int(svd.explained_variance_ratio_.sum() * 100)))
```

```
CPU times: user 13.4 s, sys: 6.93 s, total: 20.4 s
Wall time: 6.07 s
Explained variance of the SVD step: 76%
```

For the NIPS dataset, LSA (n_components = W/20 explaining 76% of the original variance) reduces the size of the DTM from shape (1500, 12419) to (1500, 620)

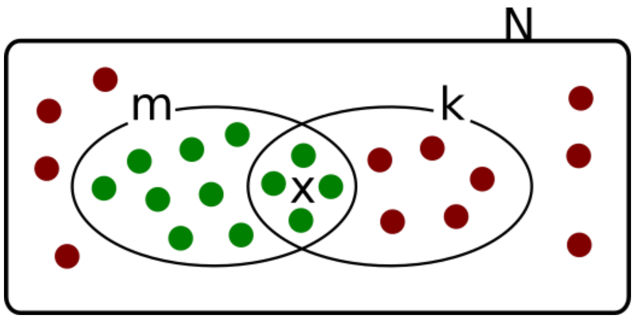
$$X' \approx UDV^T$$

U: Dense *d* by *s* (*d*: num documents, *s*: num reduced dimensions)
Dim-reduced representation of documents
D: Diagonal matrix with nonnegative entries
V: Dense *w* by *s* (*w*: num terms)
Dim-reduced representation of terms

NIPS dataset

K-Means k=15

Cluster 0	ica	blind	separation	signal	sources
Cluster 1	learning	error	function	network	weight
Cluster 2	network	attractor	memory	neuron	dynamic
Cluster 3	network	algorithm	data	model	learning
Cluster 4	model	data	gaussian	likelihood	bayesian
Cluster 5	cell	orientation	visual	cortical	model
Cluster 6	chip	circuit	analog	voltage	vlsi
Cluster 7	neuron	spike	firing	cell	synaptic
Cluster 8	kernel	svm	vector	function	support
Cluster 9	policy	action	learning	mdp	algorithm
Cluster 10	movement	eye	motor	model	head
Cluster 11	network	unit	hidden	input	weight
Cluster 12	image	object	images	model	network
Cluster 13	controller	control	learning	action	robot
Cluster 14	speech	word	recognition	hmm	speaker

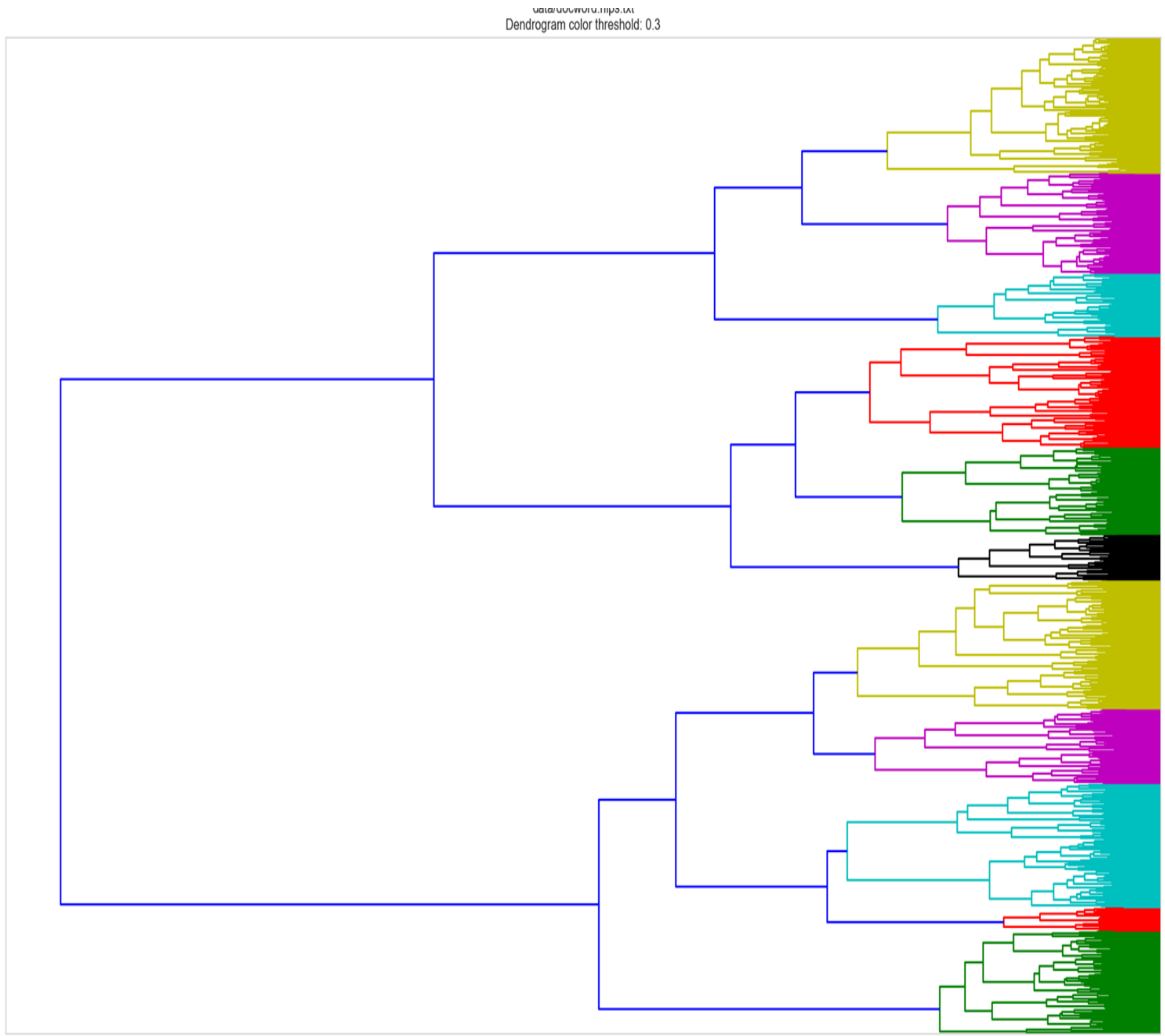


NIPS dataset

K-Means Cluster 0

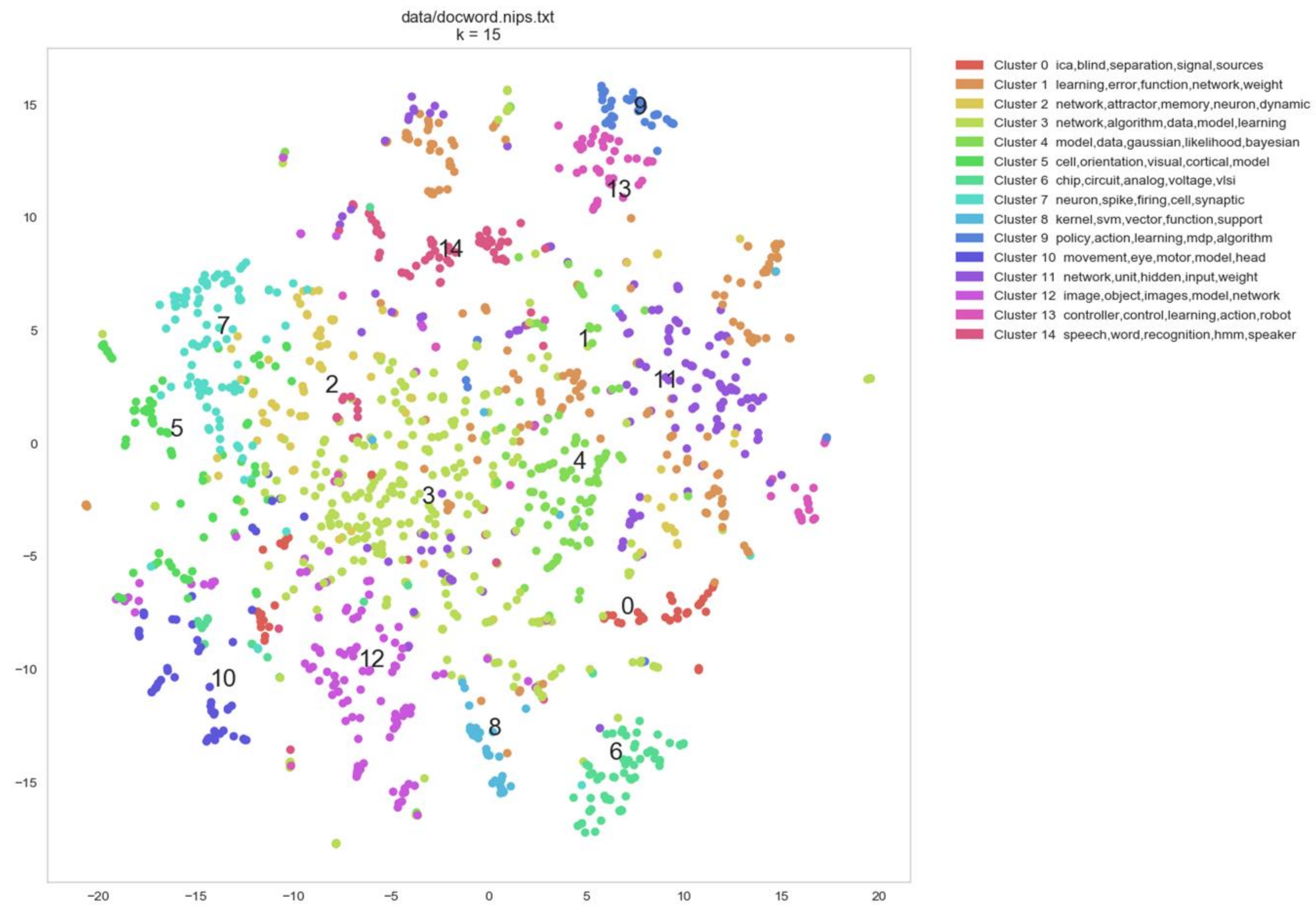
	x	k	m	N		
word	**word_count_in_cluster**	**n_words_in_cluster**	**word_count_in_corpus**	**n_words_in_corpus**	**p-val**	**adj_pval(BH)**
demixing	30	22289	30	746316	0.00E+00	0.00E+00
ica	341	22289	422	746316	0.00E+00	0.00E+00
blind	288	22289	355	746316	0.00E+00	0.00E+00
endmember	29	22289	29	746316	0.00E+00	0.00E+00
separation	361	22289	592	746316	0.00E+00	0.00E+00
equipartition	23	22289	23	746316	0.00E+00	0.00E+00
sources	327	22289	642	746316	0.00E+00	0.00E+00
signal	703	22289	4261	746316	1.09E-296	8.02E-295
sound	311	22289	653	746316	5.05E-287	3.30E-285
source	359	22289	1024	746316	1.06E-272	6.22E-271
eeg	223	22289	328	746316	1.86E-256	9.92E-255
component	523	22289	3311	746316	7.45E-212	3.65E-210
independent	374	22289	1966	746316	5.19E-180	2.35E-178
artifact	135	22289	176	746316	5.63E-169	2.37E-167
mixing	144	22289	256	746316	1.10E-148	4.31E-147
auditory	191	22289	651	746316	8.57E-130	3.15E-128
jutten	71	22289	74	746316	3.71E-107	1.28E-105
localization	115	22289	266	746316	7.34E-102	2.40E-100
spectral	132	22289	438	746316	2.94E-92	9.09E-91
deconvolution	64	22289	74	746316	5.94E-89	1.75E-87

NIPS dataset



NIPS dataset

TSNE



Enron emails

Collection of emails made public by the Federal Energy Regulatory Commission of the United States

orig source: www.cs.cmu.edu/~enron

D = 39861
W = 28102
N = 6400000 (approx)

Enron emails

SVD/LSA

```
#enron W/20 -> 74%
n_components = W/10

svd = TruncatedSVD(n_components)
# DTM_tfidf results are normalized. Since LSA/SVD results are
# not normalized, we have to redo the normalization:
normalizer = Normalizer(copy=False)
lsa = make_pipeline(svd, normalizer)

%time \
DTM_tfidf_lsa = lsa.fit_transform(DTM_tfidf)

print("Explained variance of the SVD step: {}%".format(int(svd.explained_variance_ratio_.sum() * 100)))
```

```
CPU times: user 11min 13s, sys: 1min 52s, total: 13min 6s
Wall time: 3min 38s
Explained variance of the SVD step: 74%
```

For the Enron dataset, LSA (n_components = W/10 explaining 74% of the original variance) reduces the size of the DTM from shape (39861, 28102) to (39861, 2810)

Enron emails

K-Means k= 10

Cluster 0	game	texas	longhorn	yard	fantasy
Cluster 1	click	online	market	free	link
Cluster 2	meeting	attend	agenda	discuss	scheduled
Cluster 3	request	resource	status	approval	report
Cluster 4	hourahead	final	variances	detected	schedulingiso
Cluster 5	ferc	customer	california	bill	market
Cluster 6	attached	report	review	gas	deal
Cluster 7	corp	contract	estoppel	enforceable	binding
Cluster 8	power	california	electricity	energy	company
Cluster 9	think	going	ill	office	hope

NYTimes articles

orig source: ldc.upenn.edu

D = 300,000
W = 102,660
N = 100,000,000 (approx)

PubMed Abstracts

orig source: www.pubmed.gov

D = 8,200,000
W = 141,043
N = 730,000,000 (approx)

Very large datasets → Huge Document-Term Matrices

Approach
(Just started doing this)

- Save DTM matrix to binary format

```
np.savez('DTM_kos.npz', data=DTM.data, indices=DTM.indices, indptr= DTM.indptr, shape=DTM.shape)
```

- Load DTM from binary file
- Use pyspark.mllib.linalg.distributed and its computeSVD function

```
npz_rdd = sc.binaryFiles("DTM_kos.npz")
local_bytes = npz_rdd.collect()[0][1]
local_np_obj = np.load(StringIO(local_bytes))
```

```
rows = sc.parallelize(local_np_obj)

mat = RowMatrix(rows)

# Compute the top 5 singular values and corresponding singular vectors.
svd = mat.computeSVD(5, computeU=True)
U = svd.U       # The U factor is a RowMatrix.
s = svd.s       # The singular values are stored in a local dense vector.
V = svd.V       # The V factor is a local dense matrix.
```