
Clustering and Unsupervised Learning

CS 536: Machine Learning
Littman (Wu, TA)

K-Means Clustering

Slides from Andrew Moore (CMU).

Some Reading

Not sure what to suggest for K-Means and single-link hierarchical clustering.

Kleinberg (2002). “An impossibility theorem for clustering” in *NIPS-15*.

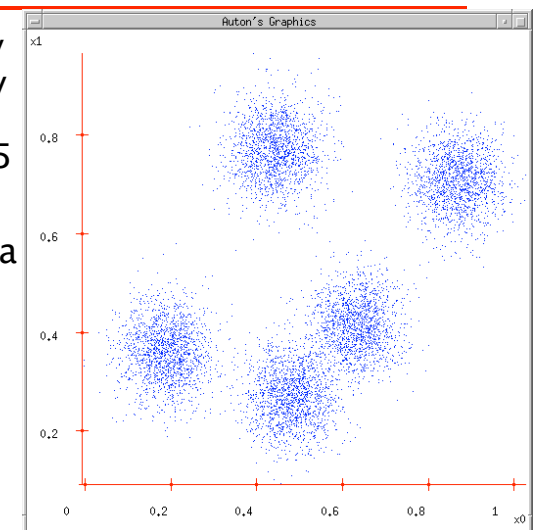
Landauer, Laham, and Foltz (1999).

“Learning human-like knowledge by singular value decomposition: A progress report” in *NIPS-10*.

Some Data

This could easily be modeled by a Gaussian Mixture (with 5 components)

But let’s look at a satisfying, friendly and infinitely popular alternative...

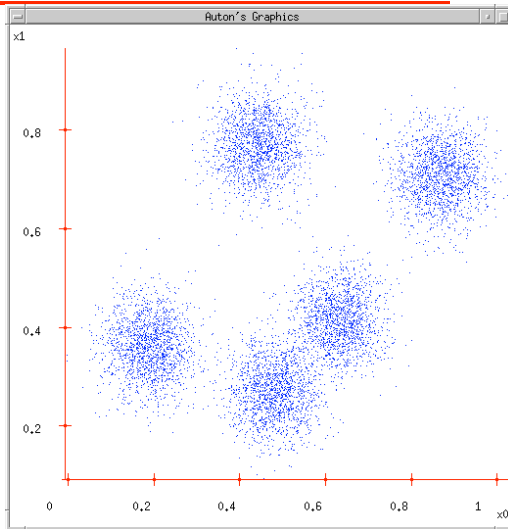


Lossy Compression

Want to transmit points, lossily, with only two bits per point.

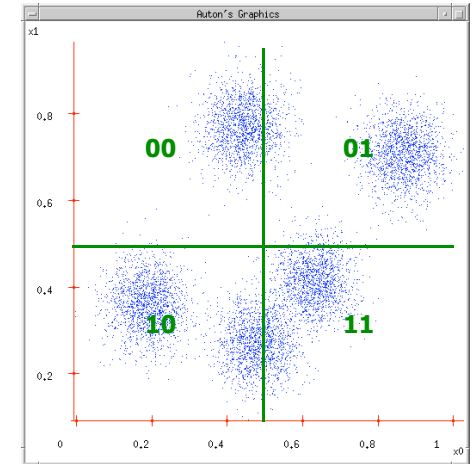
Loss = Sum Squared Error between decoded and original coordinates.

What encoder/decoder will lose the least information?



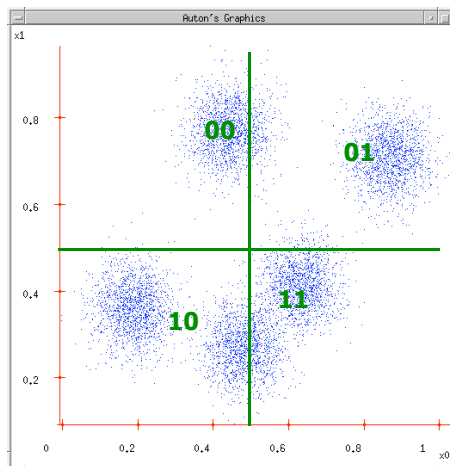
First Idea

Break into a grid, decode each bit-pair as the middle of each grid-cell.
Any better ideas?



Second Idea

Break into a grid, decode each bit-pair as the centroid of all data in that grid-cell.
Other ideas?

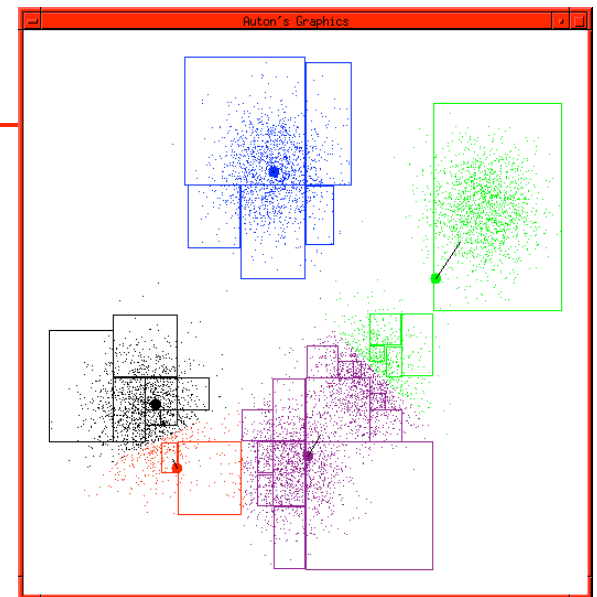
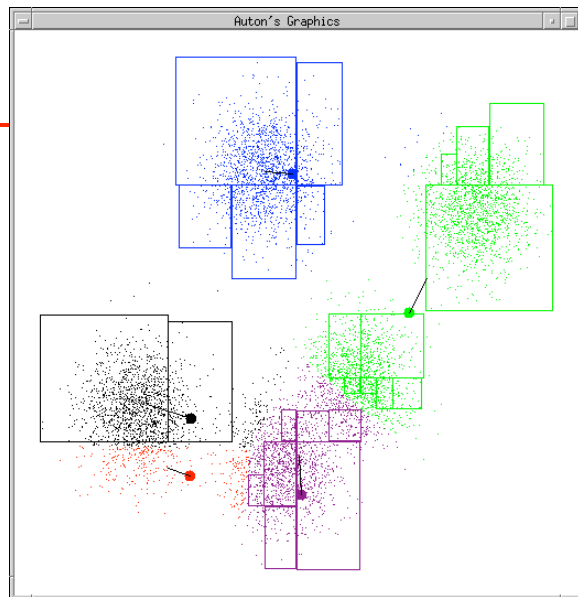
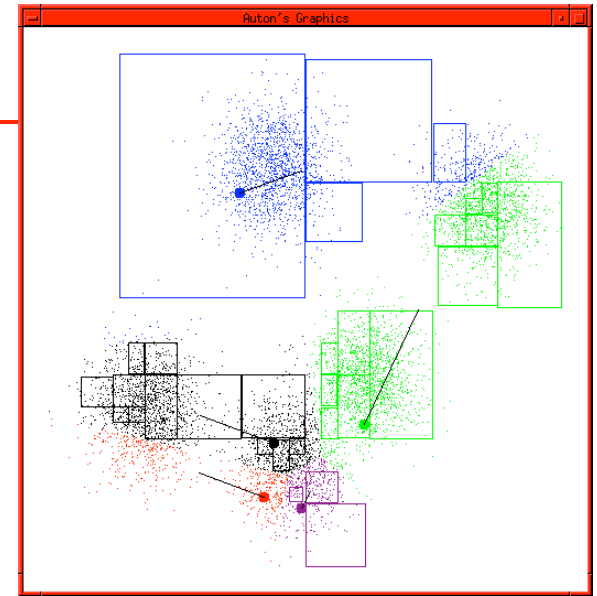
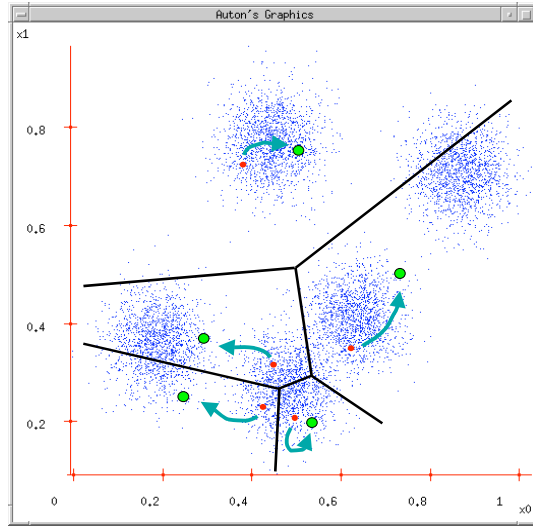


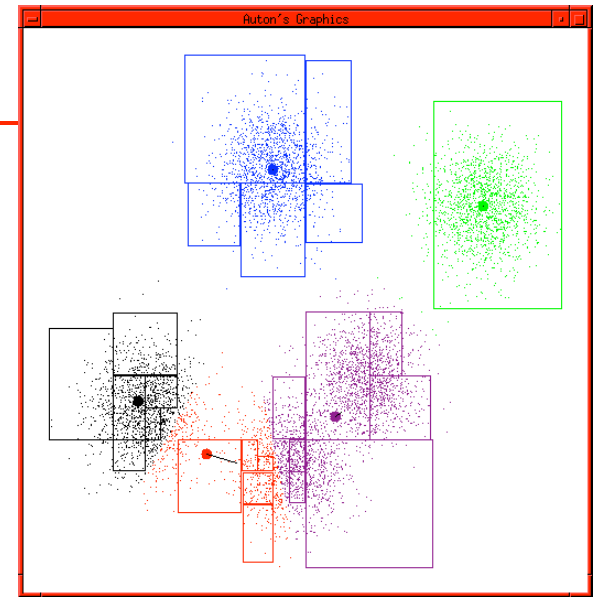
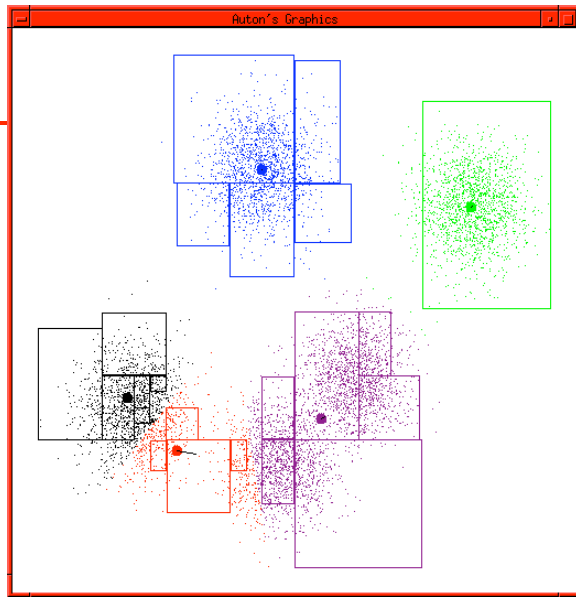
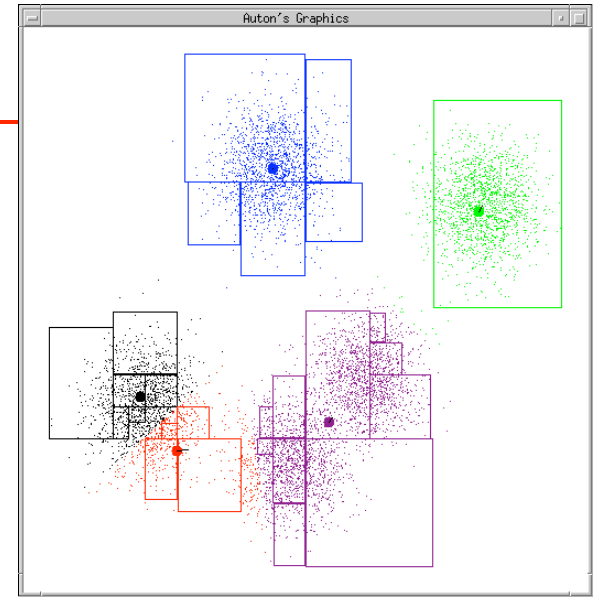
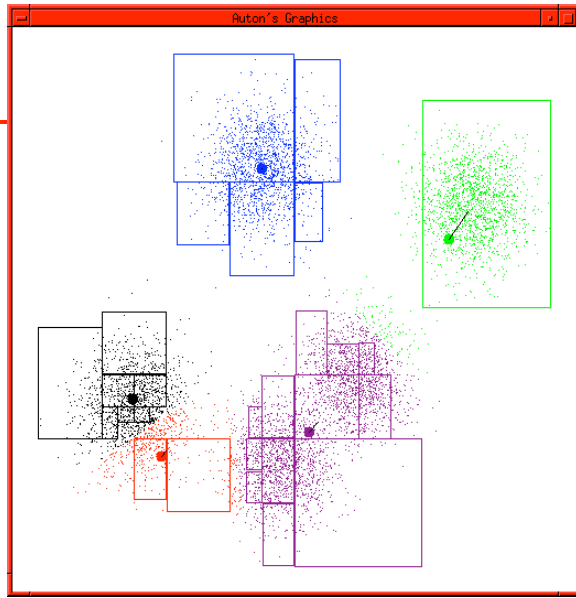
K-Means

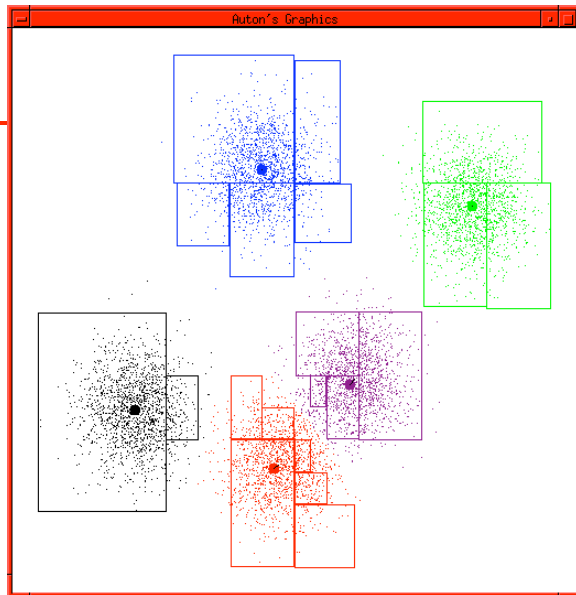
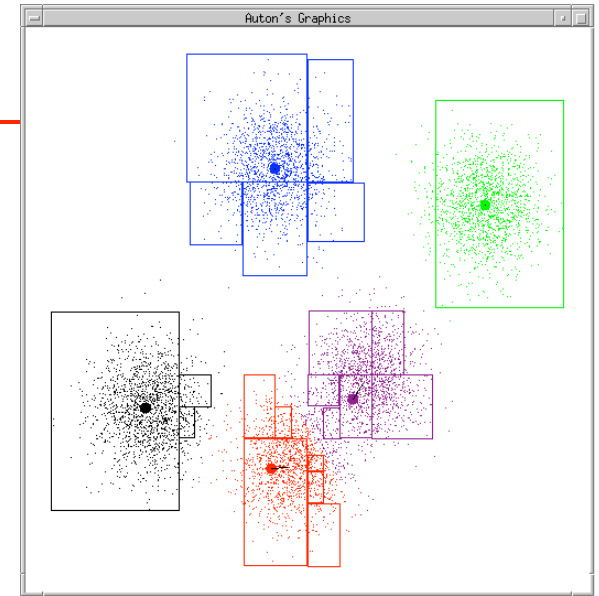
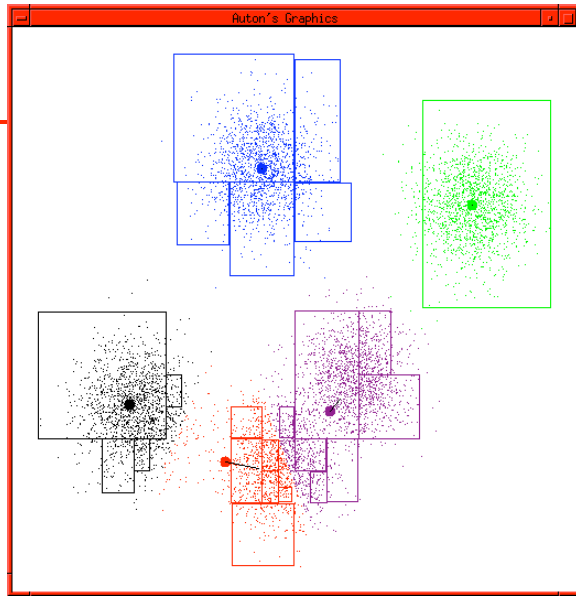
- Ask user how many clusters he'd like. (say, $k=5$)
- Randomly guess k cluster center locations
- Each datapoint finds out which center it's closest to.
- Each center finds the centroid of the points it owns...
- ...and jumps there
- ...Repeat until terminated!

K-Means Illustrated

For fast implementation, see:
Dan Pelleg and Andrew Moore. Accelerating Exact k-means Algorithms with Geometric Reasoning. Proc. Conference on Knowledge Discovery in Databases 1999, (KDD99) (available on www.autonlab.org/pap.html)







K-Means Questions

- What is it trying to optimize?
- Are we sure it will terminate?
- Are we sure it will find an optimal clustering?
- How should we start it?
- How could we automatically choose the number of centers?

....we'll deal with these questions over the next few slides

Distortion

Given..

- an encoder function: ENCODE : $\mathfrak{R}^m \rightarrow [1..k]$
- a decoder function: DECODE : $[1..k] \rightarrow \mathfrak{R}^m$

Define...

$$\text{Distortion} = \sum_{i=1}^R (\mathbf{x}_i - \text{DECODE}[\text{ENCODE}(\mathbf{x}_i)])^2$$

We may as well write

$$\text{DECODE}[j] = \mathbf{c}_j$$

$$\text{so Distortion} = \sum_{i=1}^R (\mathbf{x}_i - \mathbf{c}_{\text{ENCODE}(\mathbf{x}_i)})^2$$

The Minimal Distortion

What properties must centers $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_k$ have when distortion is minimized?

$$\text{Distortion} = \sum_{i=1}^R (\mathbf{x}_i - \mathbf{c}_{\text{ENCODE}(\mathbf{x}_i)})^2$$

(1) \mathbf{x}_i must be encoded by its nearest center

....why?

$$\mathbf{c}_{\text{ENCODE}(\mathbf{x}_i)} = \arg \min_{\mathbf{c}_j \in \{\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_k\}} (\mathbf{x}_i - \mathbf{c}_j)^2$$

...at the minimal distortion

The Minimal Distortion

(2) Partial derivative of Distortion with respect to each center location must be zero. (Center at centroid).

$$\text{Distortion} = \sum_{i=1}^R (\mathbf{x}_i - \mathbf{c}_{\text{ENCODE}(\mathbf{x}_i)})^2$$

$$= \sum_{j=1}^k \sum_{i \in \text{OwnedBy}(\mathbf{c}_j)} (\mathbf{x}_i - \mathbf{c}_j)^2$$

OwnedBy(\mathbf{c}_j) = the set of records owned by center \mathbf{c}_j .

$$\frac{\partial \text{Distortion}}{\partial \mathbf{c}_j} = \frac{\partial}{\partial \mathbf{c}_j} \sum_{i \in \text{OwnedBy}(\mathbf{c}_j)} (\mathbf{x}_i - \mathbf{c}_j)^2$$

$$= -2 \sum_{i \in \text{OwnedBy}(\mathbf{c}_j)} (\mathbf{x}_i - \mathbf{c}_j)$$

$$= 0 \text{ (for a minimum)}$$

Improving the Configuration

What properties can be changed for centers $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_k$ have when distortion is not minimized?

(1) Change encoding so that \mathbf{x}_i is encoded by its nearest center

(2) Set each center to the centroid of points it owns.

There's no point applying either operation twice in succession.

But it can be profitable to alternate.

...And that's K-means!

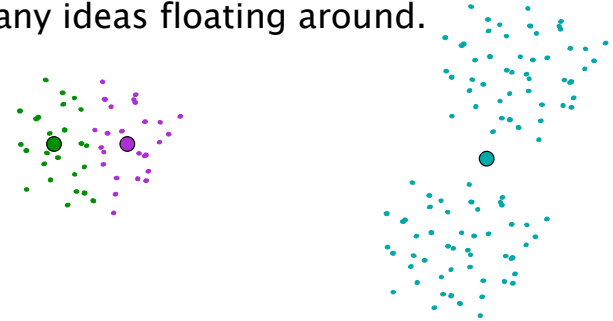
Easy to prove this procedure will terminate in a state at which neither (1) or (2) change the configuration. Why?

Is It Optimal?

- Not necessarily.
- Can you invent a configuration that has converged, but does not have the minimum distortion?

Seeking Good Optima

- Idea 1: Be careful about where you start. (Use datapoints, far apart if possible.)
- Idea 2: Do many runs of K-Means, each from a different random starting point.
- Many ideas floating around.

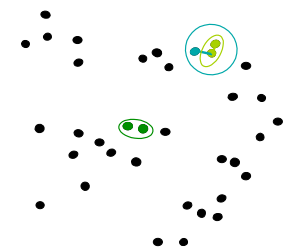


Common Uses of K-Means

- Often used as an exploratory data analysis tool
- In one-dimension, a good way to quantize real-valued variables into k non-uniform buckets
- Used on acoustic data in speech understanding to convert waveforms into one of k categories (known as Vector Quantization)
- Also used for choosing color palettes on old fashioned graphical display devices!

(Note: How choose numbers of centers?)

Single Linkage Hierarchical Clustering



1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster
4. Repeat



Some Details

How do we define similarity between clusters?

- Minimum distance between points in clusters (simply Euclidian Minimum Spanning Trees)
- Maximum distance between points in clusters
- Average distance between points in clusters

You're left with a nice dendrogram (taxonomy, hierarchy, tree) of datapoints

Stuff to Know

- All the details of K-means
- The theory behind K-means as an optimization algorithm
- How K-means can get stuck
- The outline of hierarchical clustering

Single Linkage Comments

- It's nice that you get a hierarchy instead of an amorphous collection of groups
- If you want k groups, just cut the $(k-1)$ longest links
- There's no real statistical or information-theoretic foundation to this.
- Also known in the trade as Hierarchical Agglomerative Clustering (note the acronym)

Clustering Desiderata

Abstractly, clustering algorithm f maps distance matrix D between objects to partition P of objects.

What is an ideal clustering algorithm?

1. Richness: for all P , exists D s.t. $f(D) = P$.
2. Scale invariance: for all $c > 0$ $f(D) = f(cD)$.
3. Consistency: for any D , if $f(D) = P$, then for any D' with smaller within-cluster distances and larger between-cluster distances, $f(D') = P$.

Any Pair of These Possible

Using single-link clustering, we can vary the stopping condition and get:

- *Richness and consistency*:
Stop when distances are more than r .
- *Scale invariance and consistency*:
Stop when number of clusters is k .
- *Richness and scale invariance*:
Stop when distances exceed $r/\max_{ij} D_{ij}$.

Why Clustering?

Which brings up the question, why did we want to do clustering anyway?

What is the task?

Can we accomplish the same things by assigning *distances* or *similarities* to pairs of objects?

All Three Impossible!

It is impossible to specify a clustering algorithm that satisfies all three conditions simultaneously (Kleinberg 02).

Essentially, the proof shows that consistency and scale invariance severely constrain what partitions can be created.

Alternatives to Clustering

An embedding algorithm e maps distances D to locations L in some vector space so that $\text{dist}(L_i, L_j)$ approximates D_{ij} .

- multidimensional scaling (MDS)
- singular value decomposition (SVD)
- non-negative matrix factorization (NMF)
- non-linear embedding (LLE, Isomap)
- and more

No hard decision on clusters.

Latent Semantic Analysis

Start with a set of text documents and form a term by document matrix M_{ij} .

Say the similarity between two words is defined by the dot product of the vectors of documents in which the words appear.

Near synonyms often not assigned high similarity score if they tend not to co-occur. Clustering could help.

Different idea: embed the words in a lower dimensional space, capturing similarity.

Will tend to drop “insignificant” differences.

Example LSA Applications

Train on 30k+ documents, 16k+ words. Scores at human level in vocabulary, multiple-choice psych exams.

Can grade essay tests.

Recognizes synonyms and antonyms as being closely related (but separable).

Similarity patterns more like kids ($r=.5$) than adults ($r=.3$).

Exhibits priming-like effects.

Useful for IR, and cross-language IR.

Can judge coherence, predict what to read.

Open Question

What is something we can do with clustering that we don't know how to do with embedding?

- high-level representations?
- composability?
- collecting cluster-based statistics?

Lots to think about.