Distributional Similarity Models: Clustering vs. Nearest Neighbors

Lillian Lee
Cornell University

Fernando Pereira
AT&T Research

Cooccurrence Modeling

Estimating the probability of cooccurrences is a staple of statistical NLP

▷ language modeling/speech recognition; parsing, WSD, MT, etc.

The sparse data problem: reasonable word cooccurrences are missing from training data (even very large sets)

- (Essen and Steinbiss 92): 12% of test bigrams unseen from 75K training
- (Brown et al 92): 14% of test trigrams unseen from 350M training

How do we estimate the probability of *unseen* events?

Similarity Information

We can take advantage of information provided by distributionally similar words (words occurring in the same contexts):

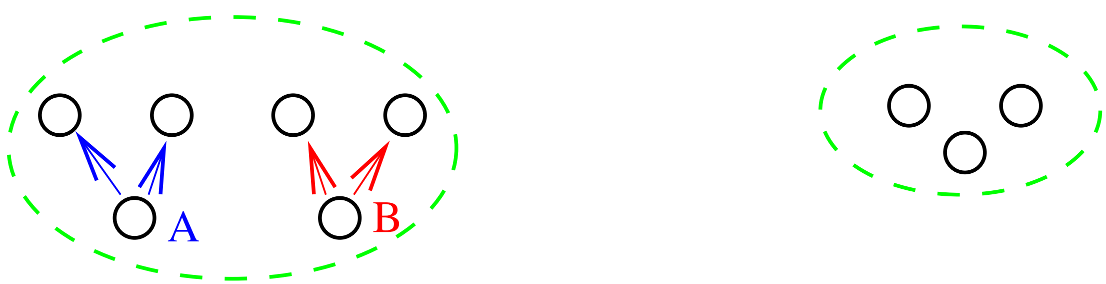
$$\left.\begin{array}{l}\text{“after ACL-95”} \\ \text{“after ACL-97”}\end{array}\right\} \Rightarrow \text{“after ACL-99” is likely}$$

What is the best way to use distributional similarity information?

Distributional Similarity Models

- Clustering [Brown et al. 92; Schütze 92, Pereira-Tishby-Lee 93; Karov-Edelman 96, Li-Abe 97; Rooth et al 99, Lee-Pereira 99]
  - ▷ Group words into global clusters; use clusters as models
  - ▷ Compresses the data
- Nearest neighbors [Dagan-Marcus-Markovitch 93, Dagan-Lee-Pereira 94, Dagan-Lee-Pereira 97, Lee-Pereira 99, Lee 99]
  - ▷ For each word, use words in its specific local neighborhood as model

Example: two clusters vs. two neighbors



Which Model?

"... it is not clear that word co-occurrence patterns can be generalized to class co-occurrence parameters without losing too much information." [DMM95]

Let's find out!

Cluster Model

Goal: $\hat{P}(y|x) > 0$ even when #(x,y) = 0 (assume P(x) > 0)

Method: introduce clusters c as stand-ins for x's – for instance:

$$\hat{P}(y|x) = \sum_c \underbrace{\hat{P}(y|c)}_{\text{class}} \underbrace{\hat{P}(c|x)}_{\text{membership}}$$

A cluster is an average of its members:

$$\hat{P}(y|c) = \sum_x \hat{P}(y|x)\hat{P}(x|c)$$

Probabilistic membership represents *ambiguity* (apple: company, fruit)

Cluster Model (cont.)

We need to find the membership probabilities.

Optimization: maximize mutual info $I(C, Y)$ subject to fixed $I(C, X)$

(maximize cluster informativeness at fixed compression)

$$\hat{P}(c|x) \propto \hat{P}(c) \exp\left(-\beta D(x, c)\right)$$

- This affects cluster positions $\Rightarrow$ iterate
- D: KL-divergence (well-known) – emerges!
- β controls number of clusters k:
  - $\triangleright$ $\beta = 0$: one c suffices
  - $\triangleright$ $\beta = \infty$: must have one c at every x

Increase number of clusters by raising β

Nearest Neighbor Model

Motivation: don't compress data into a few clusters;

for each word, consider its own local neighborhood.

Let $\mathcal{S}(x, k)$ be the k most similar words to x, according to a function of the KL divergence D.

$$\hat{P}(y|x) = \frac{1}{k} \sum_{x' \in \mathcal{S}(n,k)} P(y|x')$$

The trade-off:

- Less generalization compared to using k clusters (more accurate?)
- More storage required

Evaluation Task

Data: 3 sets of $\sim$ 1M verb-object pairs from 1989 and 1990 AP newswire; 10-fold cross-validation (for each set).

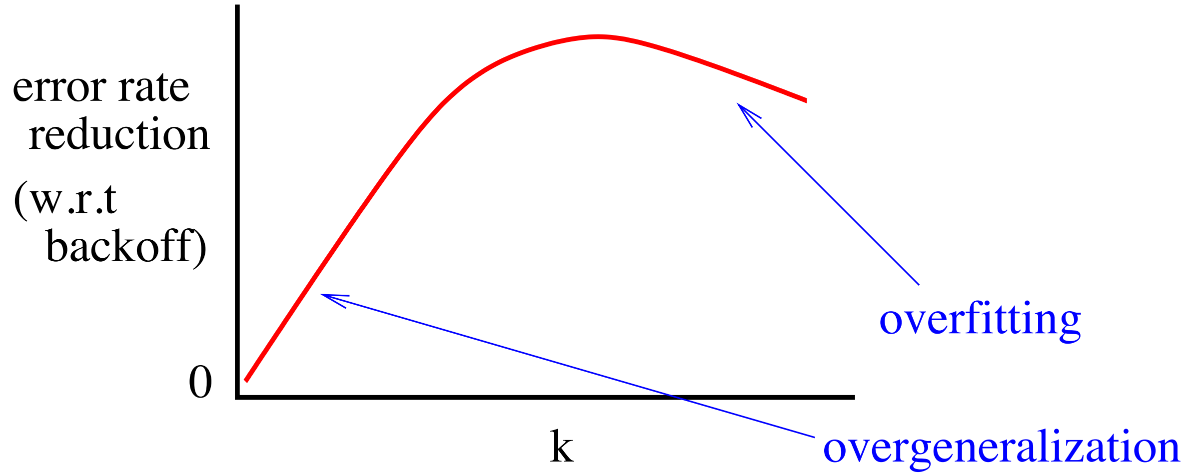
- Test instances: $\{(n, v_1), (n, v_2)\}$, both pairs unseen.
- Task: pick most likely pair

We examine error rate reduction w.r.t. Katz's backoff as a function of number of clusters/neighbors to answer the question:

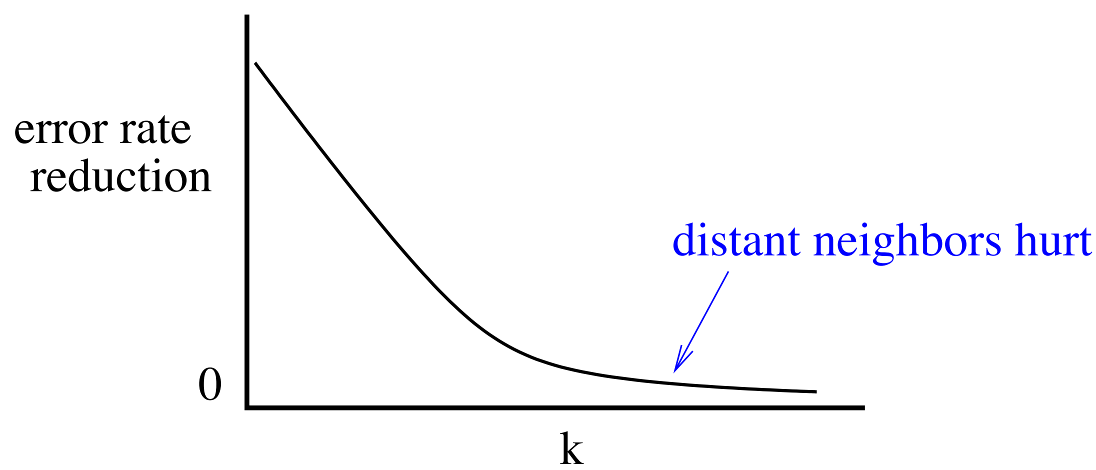
Must clustering overgeneralize?

Expected Results

CLUSTERING



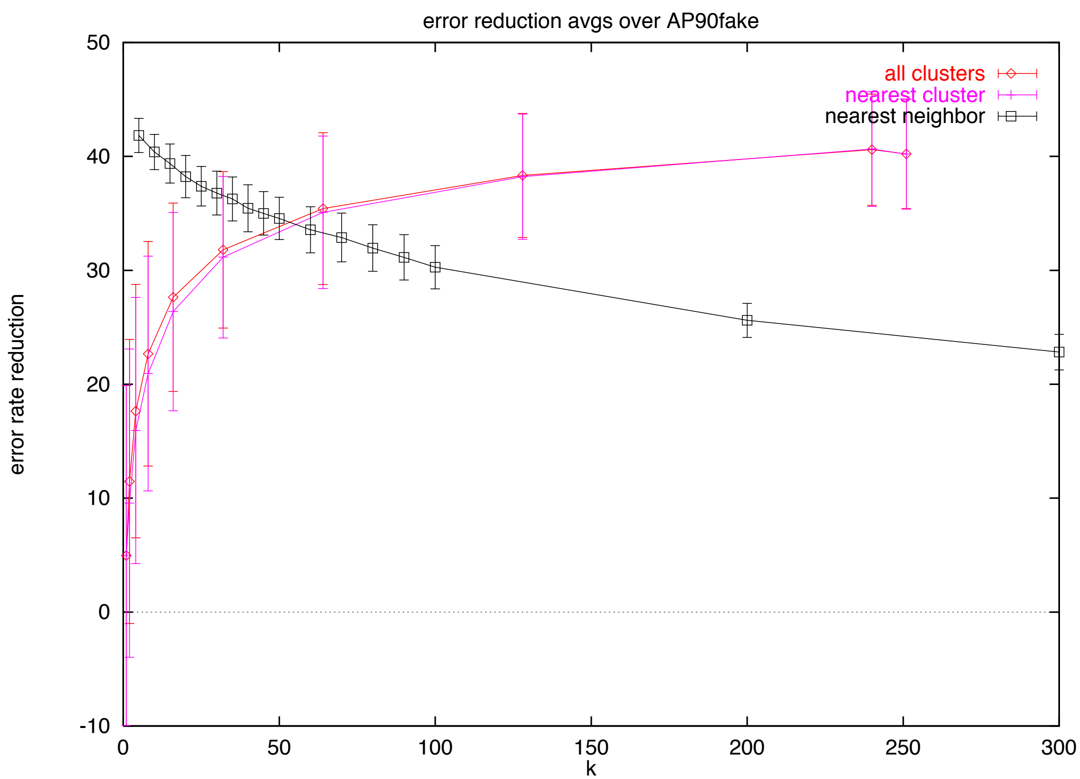
NEAREST NEIGHBOR



Implausible Alternative Test

Recall: test instances: $\{(n, v_1), (n, v_2)\}$, both unseen in training.

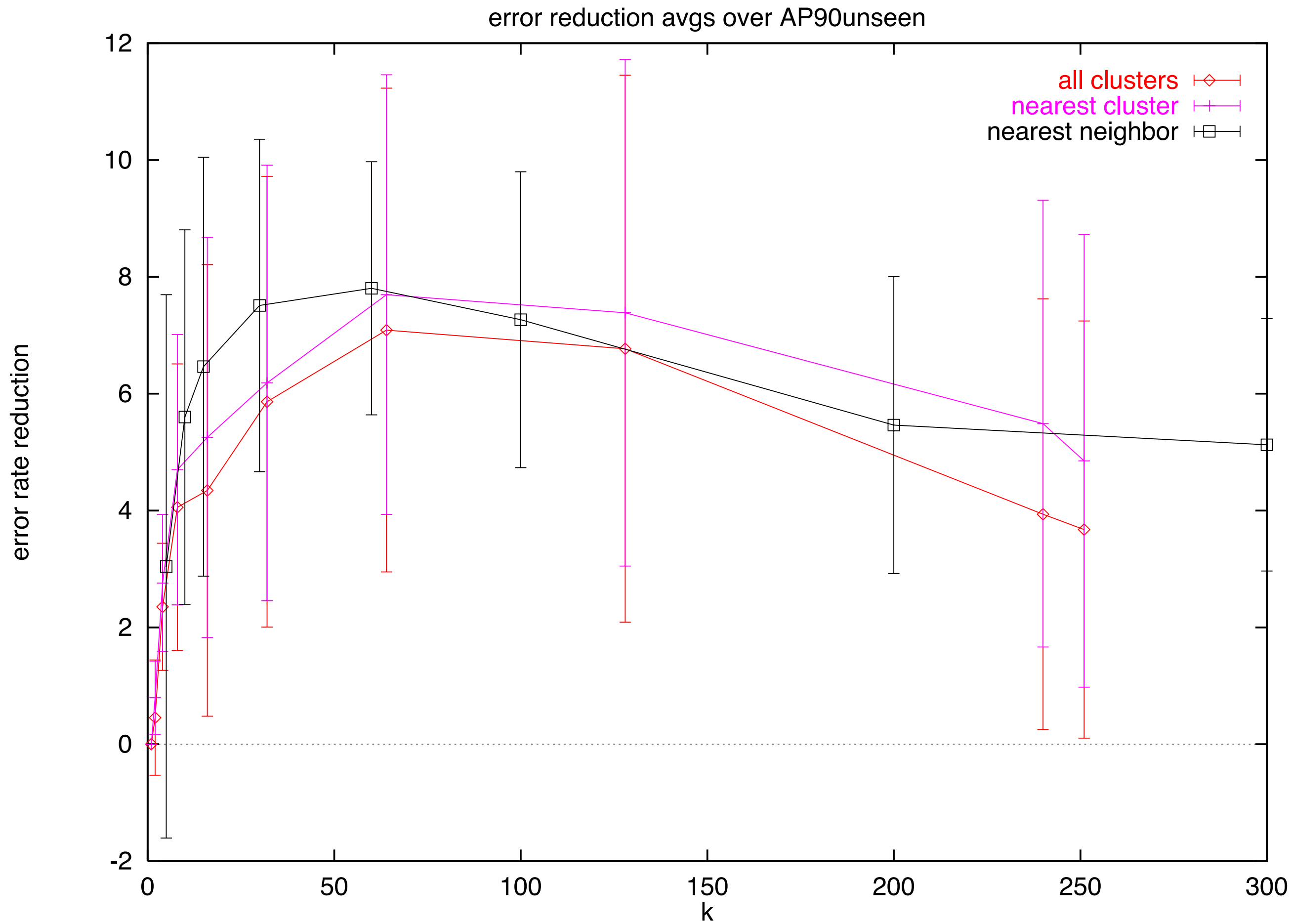
$\#(n, v1) \geq 2, \quad \#(n, v2) = 0, \quad \#(v2)$ large.



Baseline: 79.9%

Plausible Alternative Test

$$\#(n, v1) \geq 2\#(n, v2) > 0$$

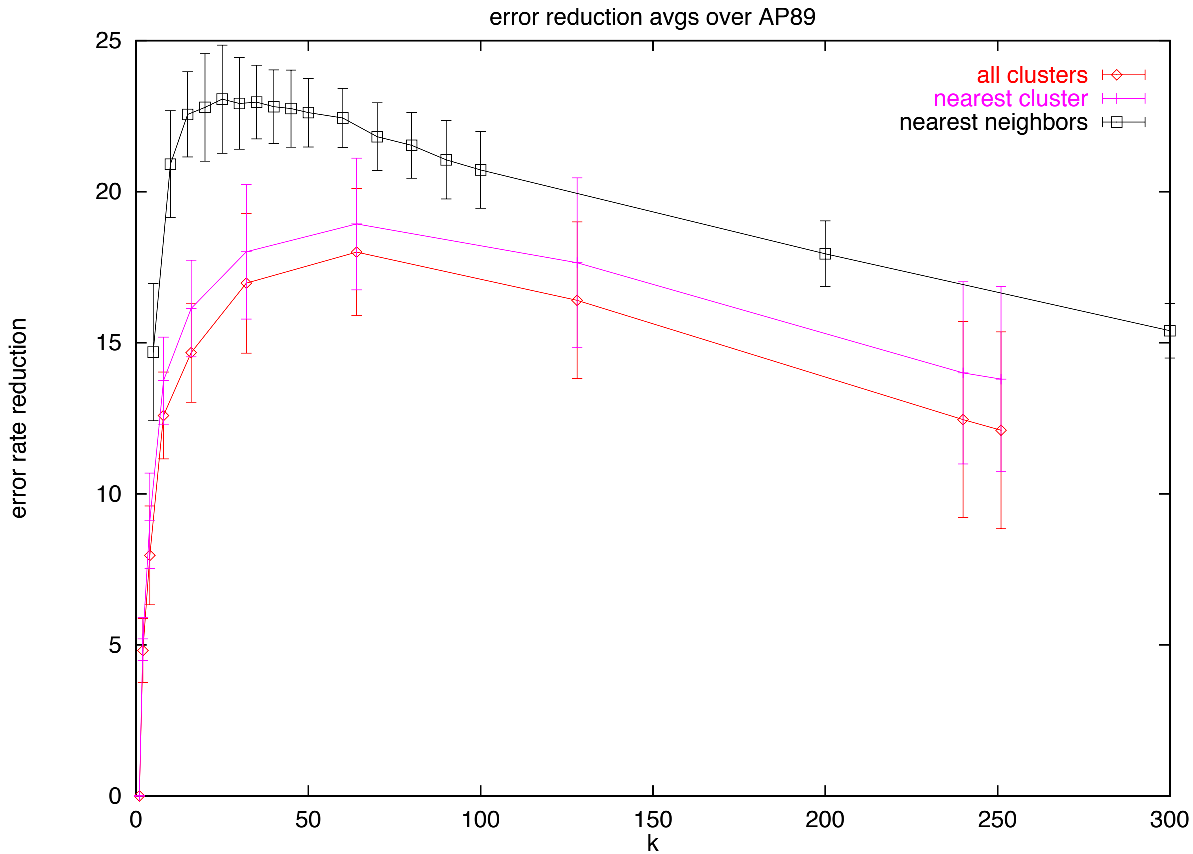


Baseline: 39.9%

Plausible Alternative, Sparse Test

As before, $\#(n, v1) \geq 2\#(n, v2) > 0$.

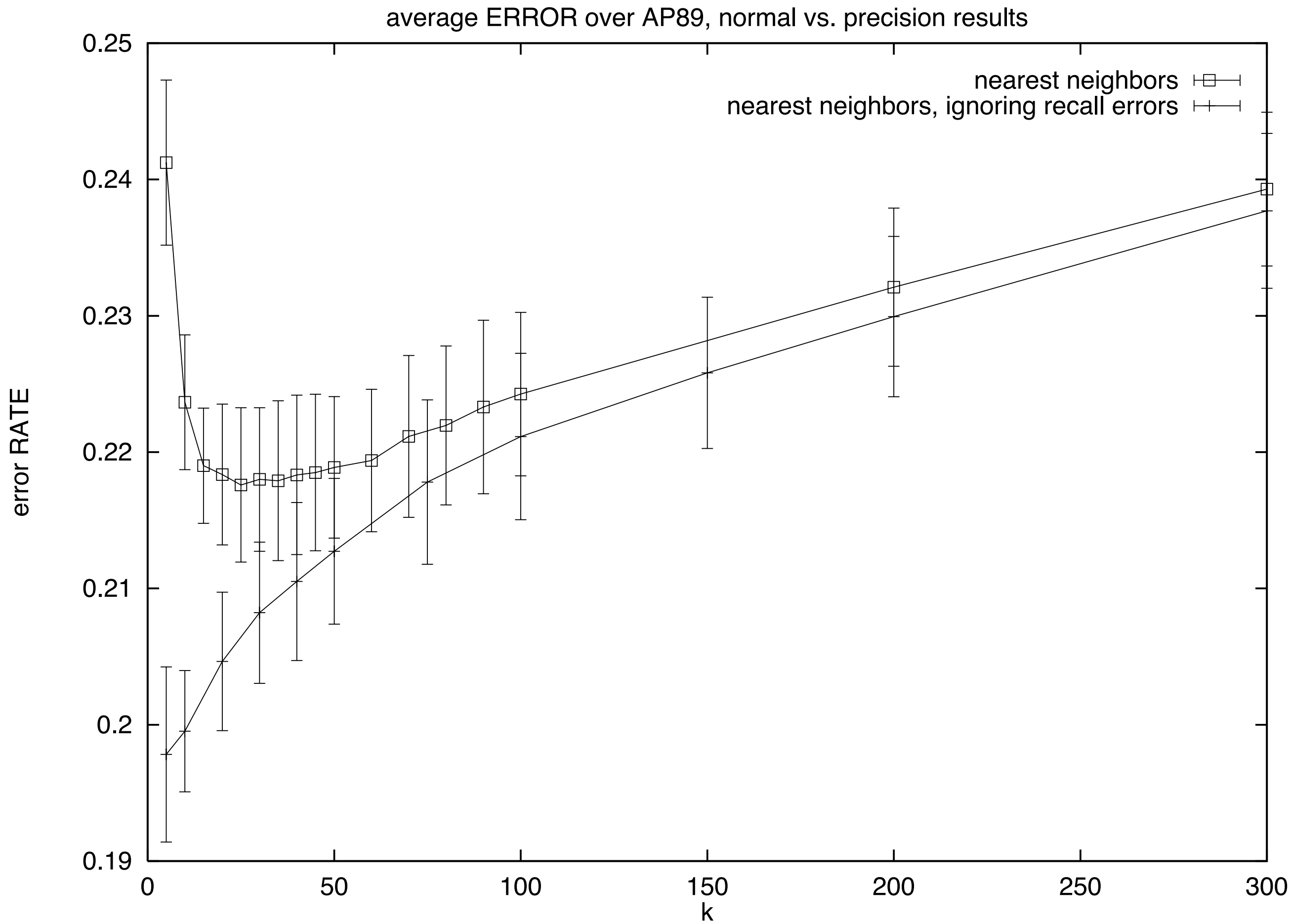
But, corpus split by type (artificial?); pairs occurring just once deleted.



Baseline: 28.3%

Sparseness Affects Nearest Neighbors

Ignoring (n, v) where no n' occurs with v gives expected error rate behavior.



Clustering is immune to this problem.

Conclusion

Clustering and nearest-neighbors generally obtain surprisingly similar optimal performance rates.

$\triangleright$ Small optimal k values: computational/memory efficiency

Questions:

- Why does nearest-neighbors do better in the sparse test?
- Why are the two models so close in the other tests?
- Why does clustering have higher variance?