Semantic Hashing

Ruslan Salakhutdinov and Geoffrey Hinton

University of Toronto, Machine Learning Group

IRGM Workshop

July 2007

Existing Methods

- One of the most popular and widely used in practice algorithms for document retrieval tasks is TF-IDF. However:
  - It computes document similarity directly in the word-count space, which can be slow for large vocabularies.
  - It assumes that the counts of different words provide independent evidence of similarity.
  - It makes no use of semantic similarities between words.

- To overcome these drawbacks, models for capturing low-dimensional, latent representations have been proposed and successfully applied in the domain of information retrieval.

- One such simple and widely-used method is Latent Semantic Analysis (LSA), which extracts low-dimensional semantic structure using SVD to get a low-rank approximation of the word-document co-occurrence matrix.

Drawbacks of Existing Methods

- LSA is a linear method so it can only capture pairwise correlations between words. We need something more powerful.
- Numerous methods, in particular probabilistic versions of LSA were introduced in the machine learning community.
- These models can be viewed as graphical models in which a single layer of hidden topic variables have directed connections to variables that represent word-counts.
- There are limitations on the types of structure that can be represented efficiently by a single layer of hidden variables.
- Recently, Hinton et al. have discovered a way to perform fast, greedy learning of deep, directed belief nets one layer at a time.
- We will use this idea to build a network with multiple hidden layers and with millions of parameters and show that it can discover latent representations that work much better.

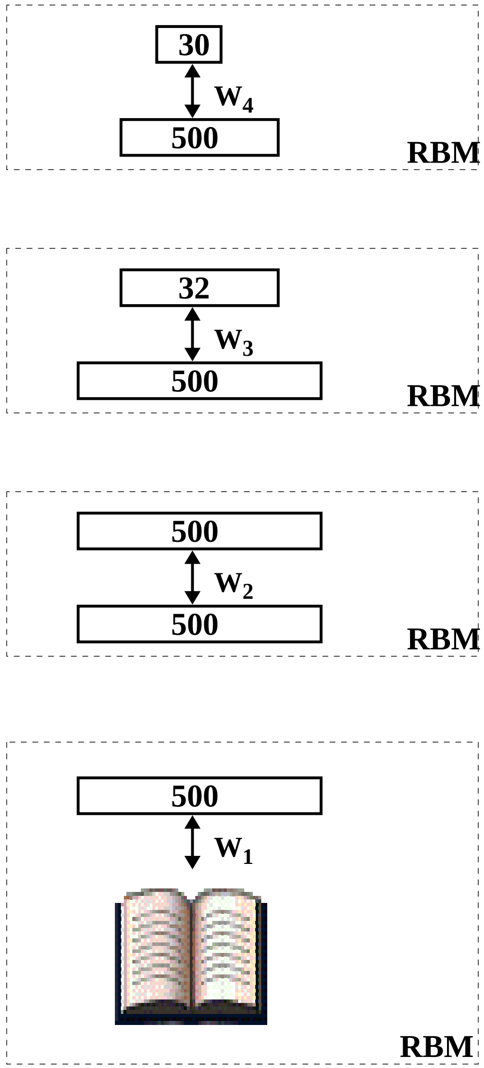
Learning multiple layers

- A single layer of features generally cannot perfectly model the structure in the data.

- We will use a Restricted Boltzmann Machine (RBM), which is a two-layer undirected graphical model, as our building block.

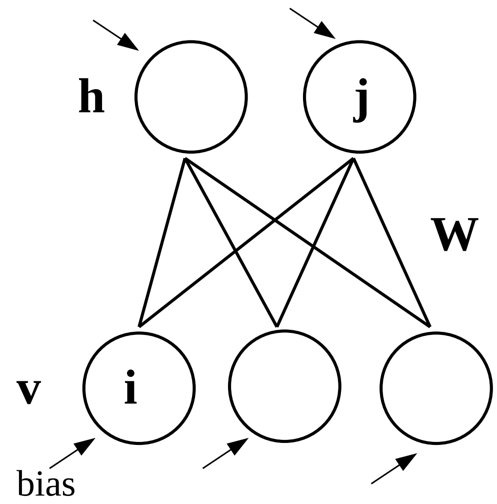
- Perform greedy, layer-by-layer learning:
  - Learn and Freeze W_1
  - Treat the learned RBM features, driven by the training data as if they were data.
  - Learn and Freeze W_2.
  - Greedily learn as many layers of features as desired.

- Each layer of features captures strong high-order correlations between the activities of units in the layer below.



RBM for count data

- Hidden units remain binary and the visible word counts are modeled by the Constrained Poisson Model.



- The energy is defined as:

$$E(\mathbf{v}, \mathbf{h}) = -\sum_i b_i v_i - \sum_j b_j h_j - \sum_{i,j} v_i h_j w_{ij} + \sum_i v_i \log \frac{Z}{N} + \sum_i \log \Gamma(v_i + 1)$$

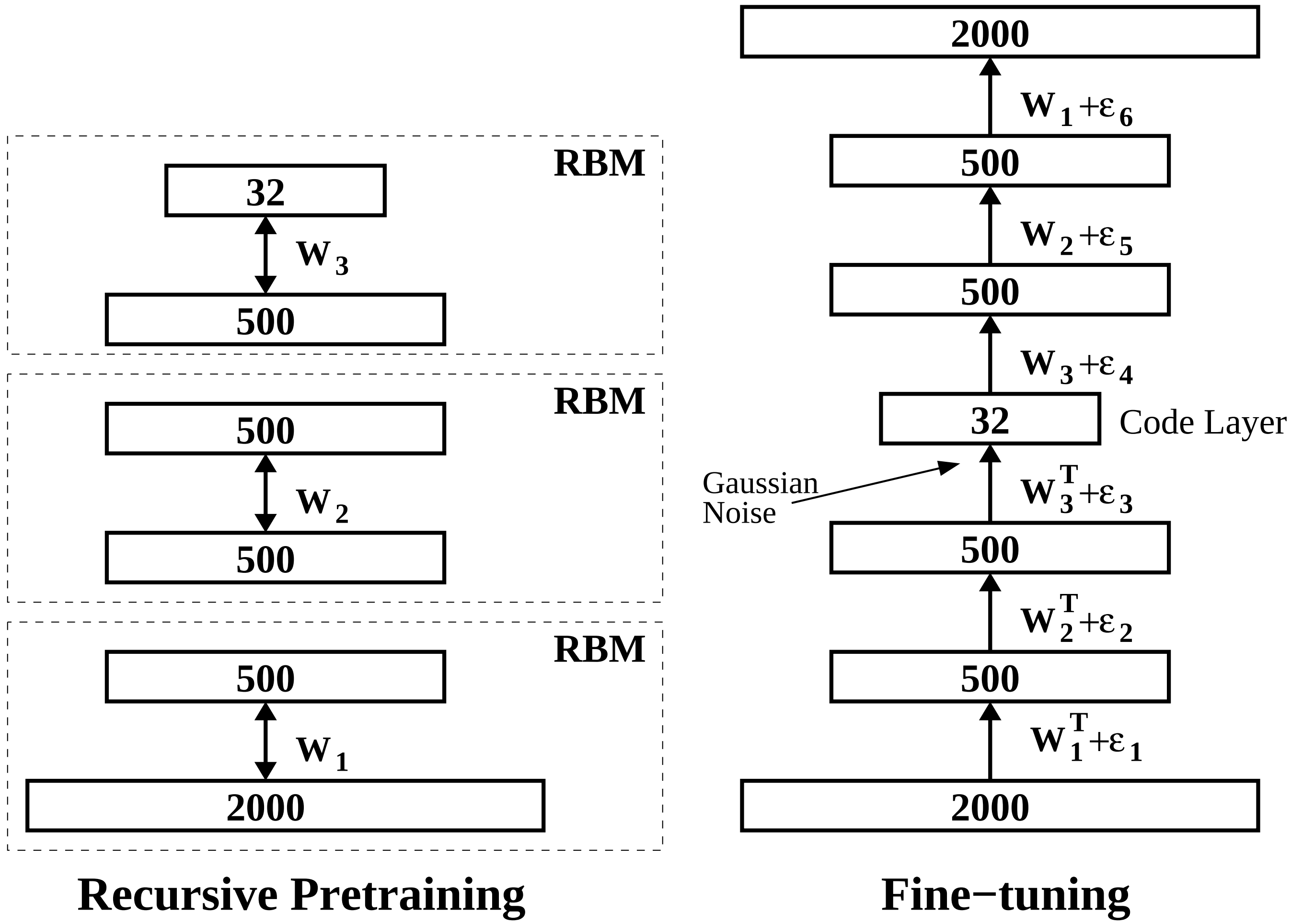
- Conditional distributions over hidden and visible units are:

$$p(h_j = 1|\mathbf{v}) = \frac{1}{1 + \exp(-b_j - \sum_i w_{ij} v_i)}$$

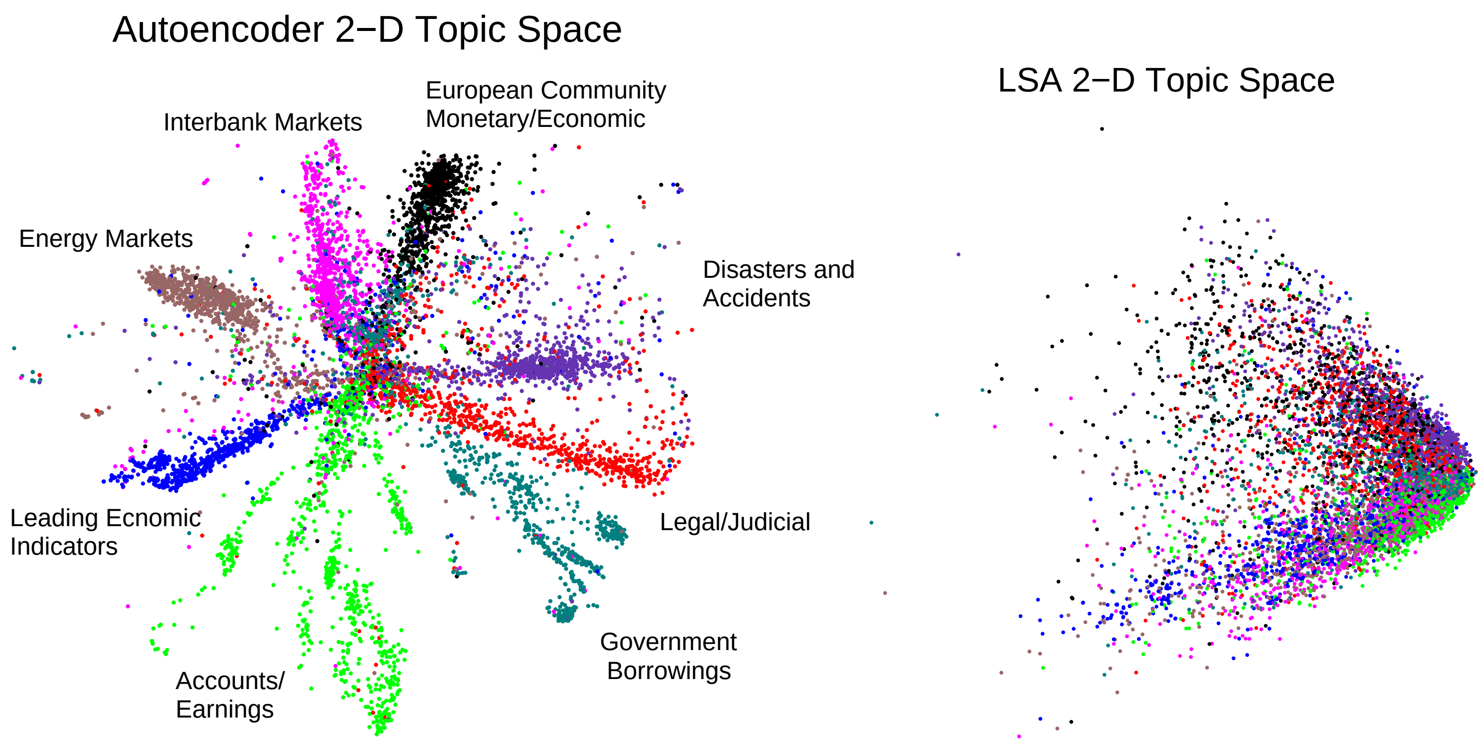
$$p(v_i = n|\mathbf{h}) = \text{Poisson}\left(\frac{\exp\left(\lambda_i + \sum_j h_j w_{ij}\right)}{\sum_k \exp\left(\lambda_k + \sum_j h_j w_{kj}\right)} N\right)$$

- where N is the total length of the document.

The Big Picture



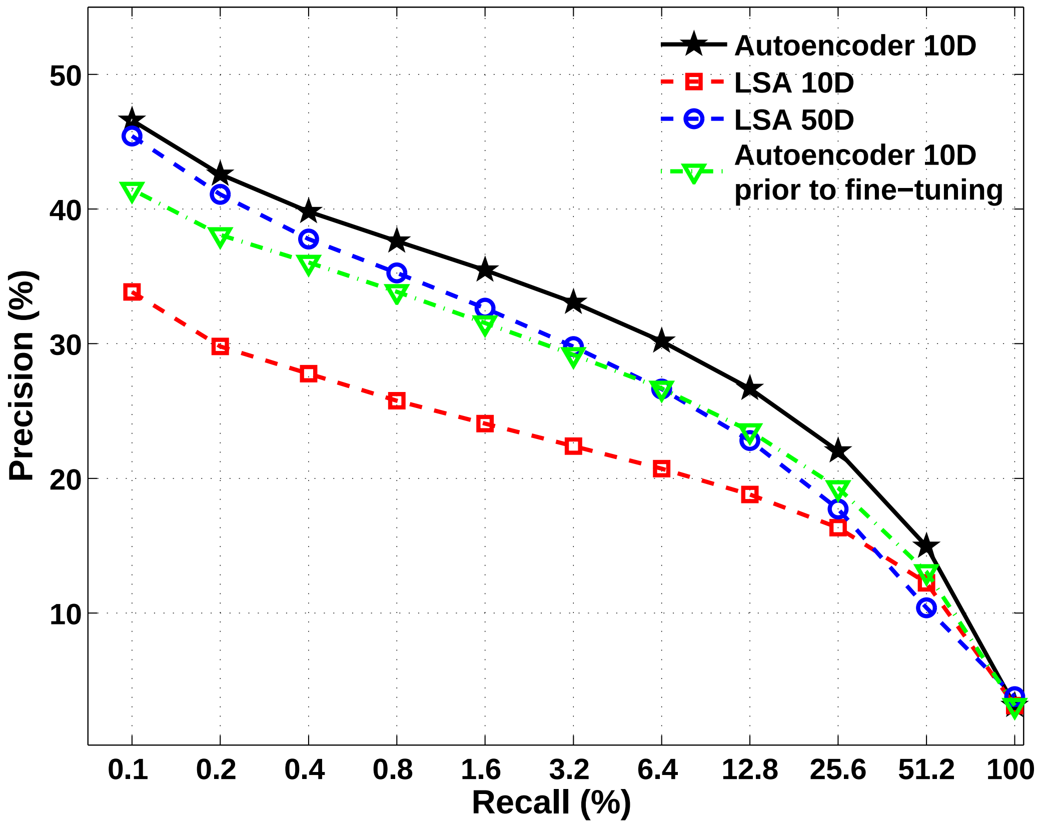
Reuters Corpus: Learning 2-D code space



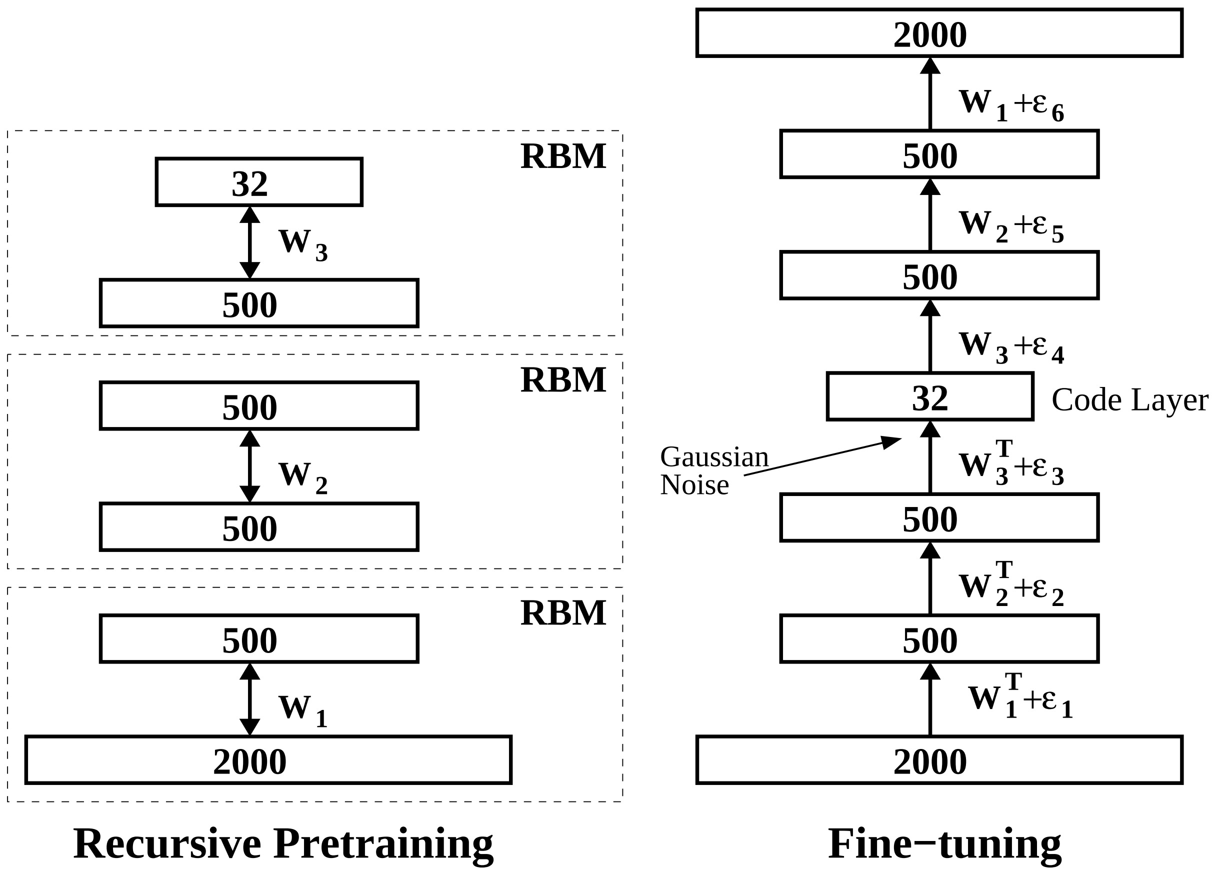
- We use a 2000-500-250-125-2 autoencoder to convert test documents into a two-dimensional code.
- The Reuters Corpus Volume II contains 804,414 newswire stories (randomly split into **402,207** training and **402,207** test).
- We used a simple "bag-of-words" representation. Each article is represented as a vector containing the counts of the most frequently used 2000 words in the training dataset.

Results for 10-D codes

- We use the cosine of the angle between two codes as a measure of similarity.
- Precision-recall curves when a 10-D query document from the test set is used to retrieve other test set documents, averaged over 402,207 possible queries.

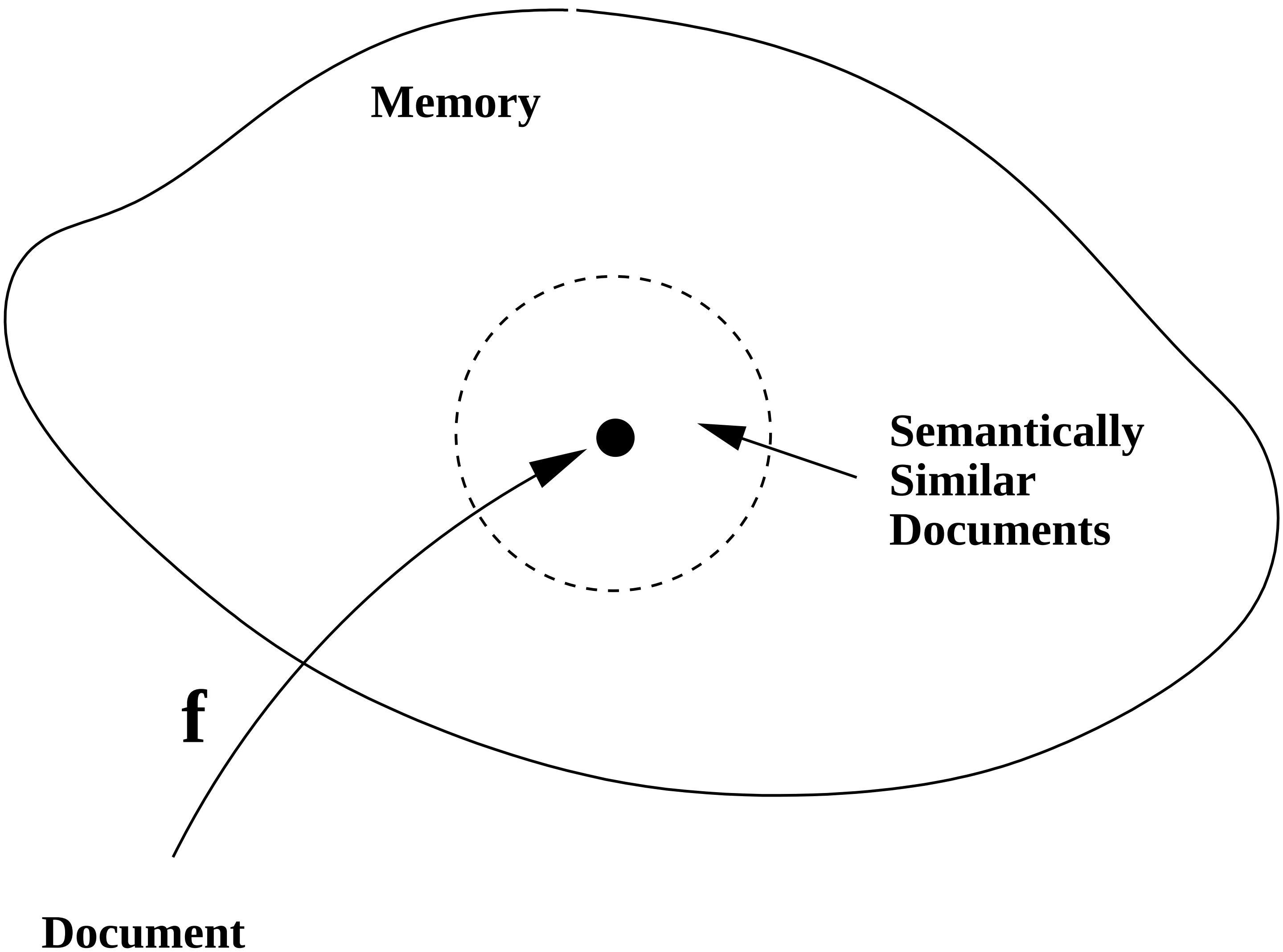


Semantic Hashing

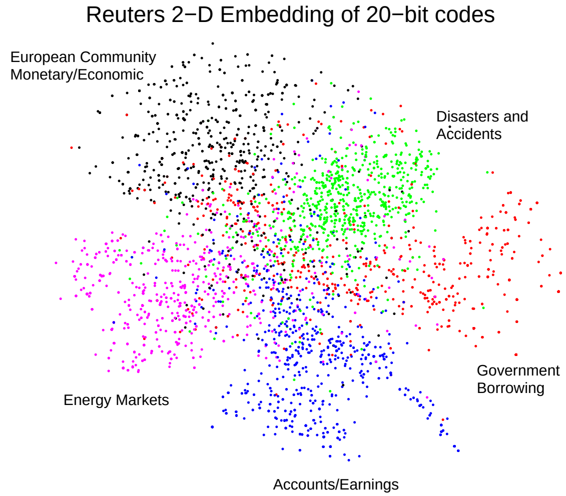


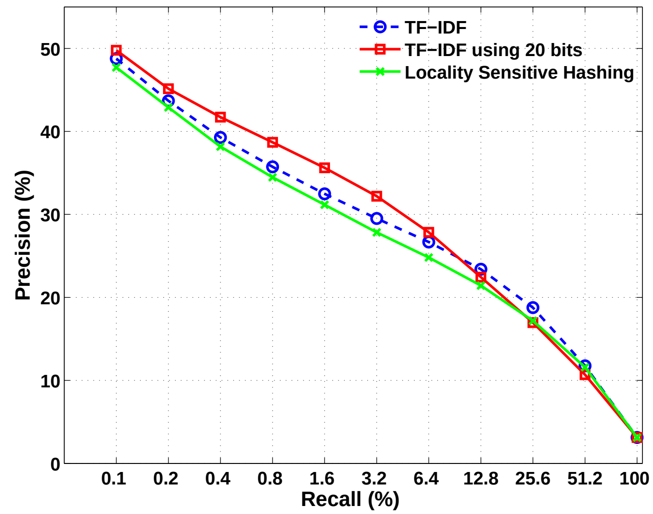
- Learn to map documents into *semantic* 32-D binary code and use these codes as memory addresses.
- We have the ultimate retrieval tool: Given a query document, compute its 32-bit address and retrieve all of the documents stored at the similar addresses **with no search at all**.

The Main Idea of Semantic Hashing



Semantic Hashing





- We used a simple C implementation on Reuters dataset (402,212 training and 402,212 test documents).
- For a given query, it takes about 0.5 milliseconds to create a short-list of about 3,000 semantically similar documents.
- It then takes 10 milliseconds to retrieve the top few matches from that short-list using TF-IDF, and it is more accurate than full TF-IDF.
- Locality-Sensitive Hashing takes about 500 milliseconds, and is less accurate. Our method is 50 times faster than the fastest existing method and is more accurate.