Language model: Bag of Words

- Vocab = set of all the words in corpus
- Document = Words in document w.r.t vocab with multiplicity
  - Sentence 1: "The cat sat on the hat"
  - Sentence 2: "The dog ate the cat and the hat”
  - Vocab = { the, cat, sat, on, hat, dog, ate, and }
  - Sentence 1: { 2, 1, 1, 1, 1, 0, 0, 0 }
  - Sentence 2 : { 3, 1, 0, 0, 1, 1, 1, 1}

Language model: One Hot Encoding

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

Each document is represented by a *binary vector* $\in \{0,1\}^{|V|}$!

Language Model: N Gramm(bigramm)

- Vocab = set of all n-grams in corpus
- Document = n-grams in document w.r.t vocab with multiplicity
  - For bigram:
  - Sentence 1: "The cat sat on the hat"
  - Sentence 2: "The dog ate the cat and the hat”
  - Vocab = { the cat, cat sat, sat on, on the, the hat, the dog, dog ate, ate the, cat and, and the}
  - Sentence 1: { 1, 1, 1, 1, 1, 0, 0, 0, 0, 0}
  - Sentence 2 : { 1, 0, 0, 0, 0, 1, 1, 1, 1, 1}

Language Model: TF-IDF

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	5.25	3.18	0	0	0	0.35
Brutus	1.21	6.1	0	1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95

TF-IDF (t) = TF(t) * IDF(t) → **TF-IDF (t) = N(TF(t)) * IDF(t)**

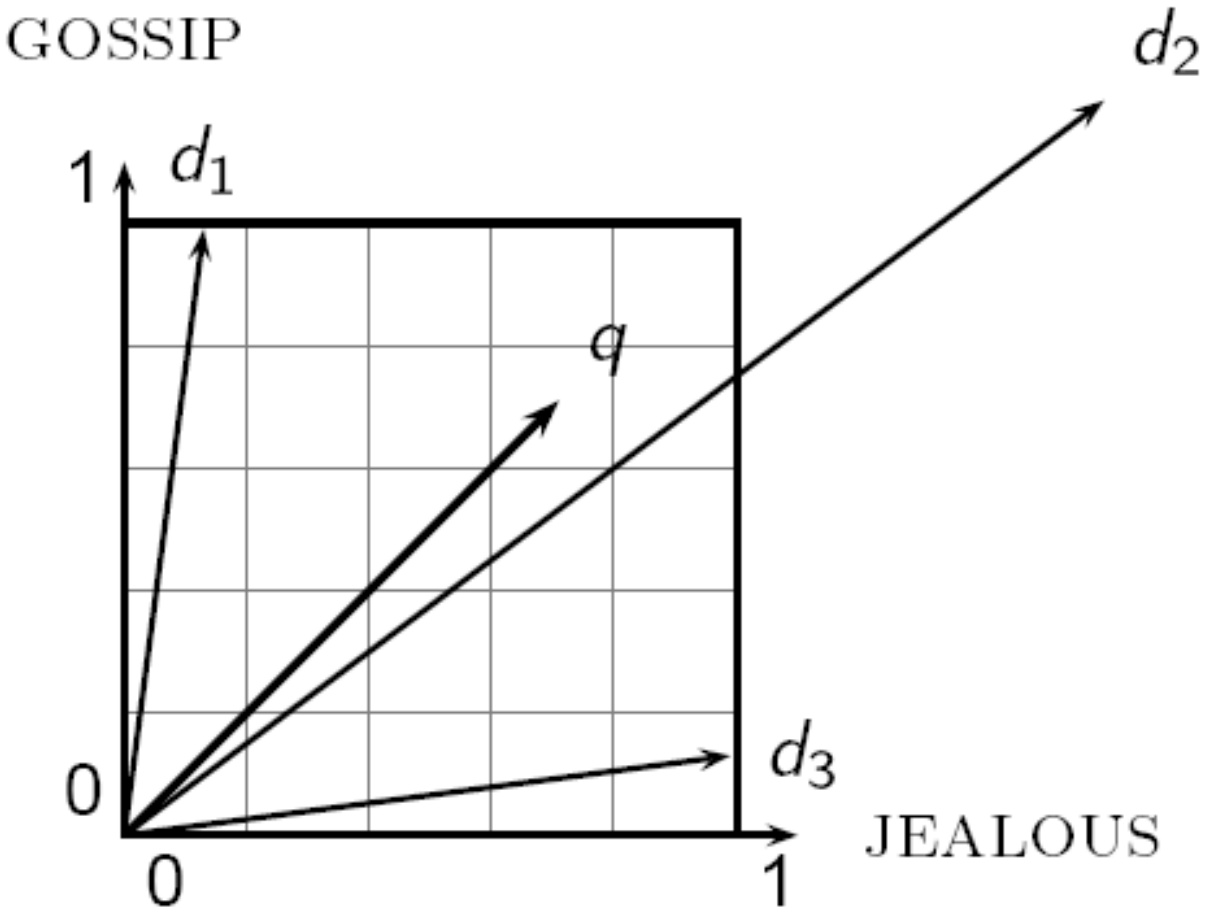
Each document is now represented by a *real-valued vector* of tf-idf weights $\in \mathbb{R}^{|V|}$

TF-IDF Model: Pros

- **Automatic** selection of index terms
- **Partial matching** of queries and documents (dealing with the case where no document contains all search terms)
- **Ranking** according to **similarity score** (dealing with large result sets)
- **Term weighting** schemes (improves retrieval performance)
- Various extensions
  - Document clustering
  - Relevance feedback (modifying query vector)
- Geometric foundation

Why distance is a bad idea

The Euclidean distance between *q* and d_2 is large even though the distribution of terms in the query *q* and the distribution of terms in the document d_2 are very similar.



Problems with Lexical Semantics

- Ambiguity and association in natural language
  - **Polysemy**: Words often have a **multitude of meanings** and different types of usage *(more severe in very heterogeneous collections).*
  - The vector space model is unable to discriminate between different meanings of the same word.

$$\mathrm{sim}_{\mathrm{true}}(d, q) < \cos(\angle(\vec{d}, \vec{q}))$$

Example of Polysemy

He left the bank five minutes ago.
He left the bank five years ago
He caught a fish at the bank.

A world record.
A record of the conversation.
Record it!

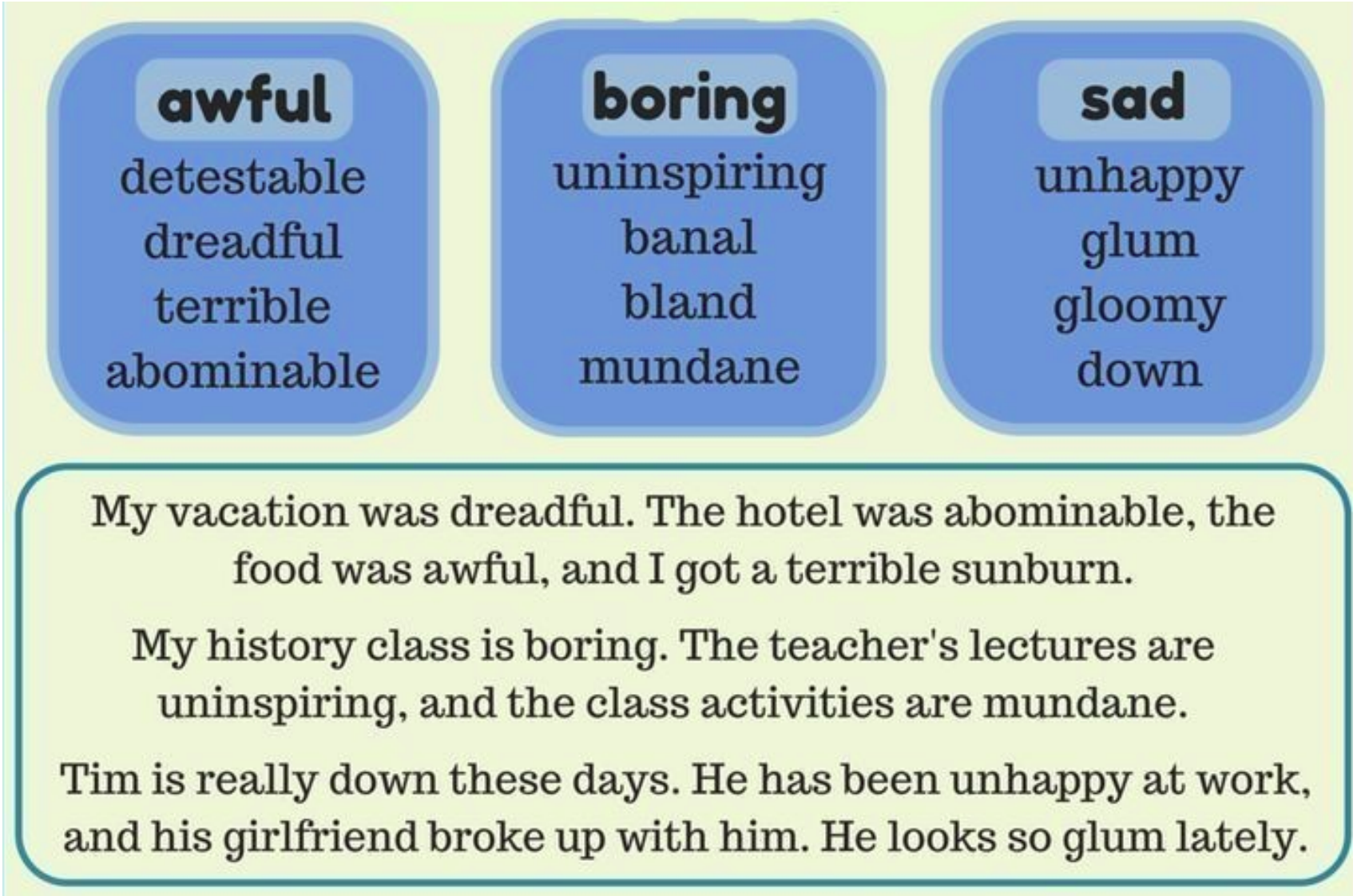
I need some paper.
I wrote a paper.
I read the paper.

Problems with Lexical Semantics

- **Synonym**: Different terms may have **identical or similar meanings** (weaker: words indicating the same topic).
- No associations between words are made in the vector space representation.

$$\mathrm{sim}_{\mathrm{true}}(d, q) > \cos(\angle(\vec{d}, \vec{q}))$$

Example of Synonym



Motivation

- Term-document matrices are very large
- But the number of topics that people talk about is small (in some sense)
  - Clothes, movies, politics, …
- Can we represent the term-document space by a lower dimensional latent space?

Topic Modeling

- Topic Modeling is a set of techniques that aim to **discover** and annotate large archives of documents with **thematic information**.
- TM is a set of methods that analyze the words (or other fine-grained features) of the original documents to **discover** the themes that run through them, how those themes are **connected** to each other, and how they **change** over time.
- Often, the **number of topics** to be discovered is predefined.
- Topic modeling can be seen as a **dimensionality reduction** technique
- Topic modeling, like clustering, **do not require any prior annotations or labeling**, but in contrast to clustering, can assign document to multiple topics.
- Topic Model types:
  - Linear algebra based (e.g. **LSA**)
  - Probabilistic modeling based (e.g. pLSA, LDA, Random Projections)

General Idea of Latent Semantic Indexing (LSI)

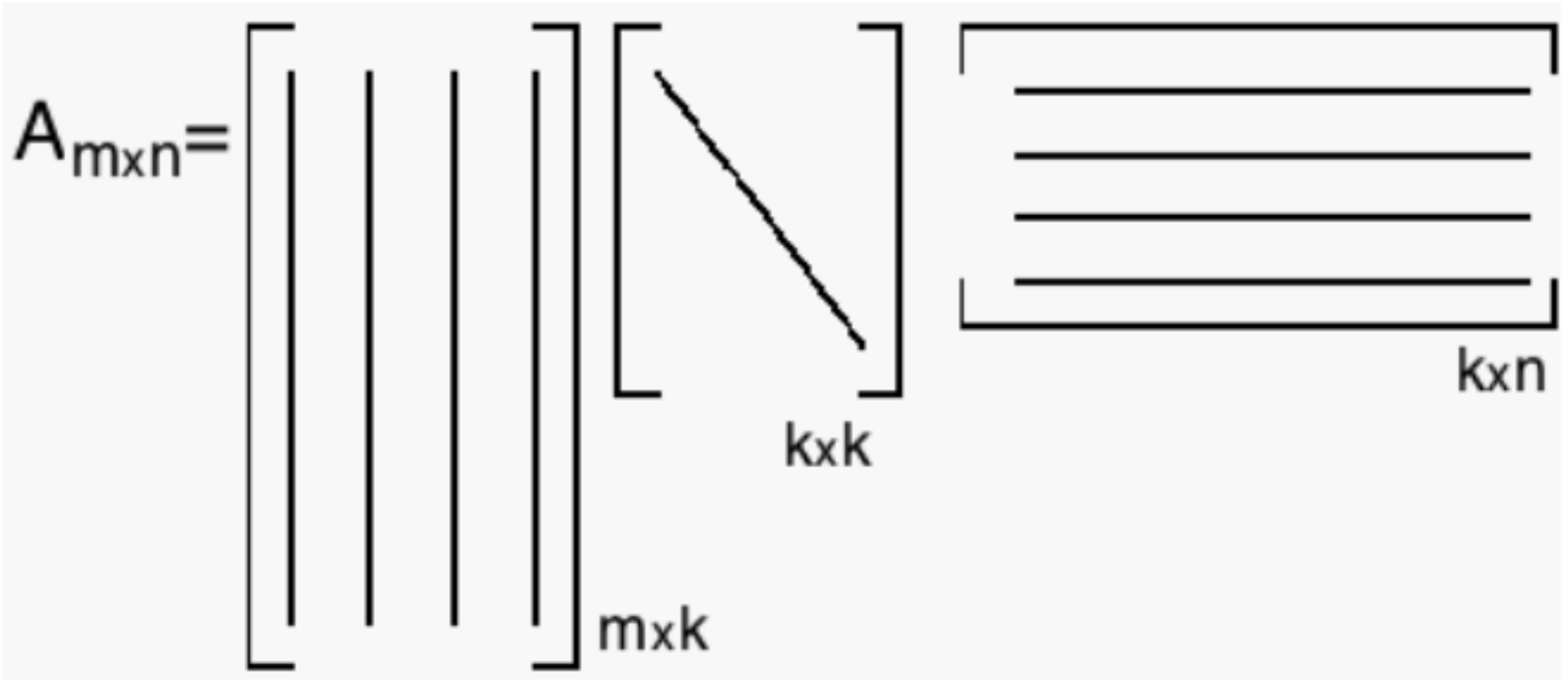
- Map documents (and terms) to a **low-dimensional** representation.
- Design a mapping such that the low-dimensional space reflects **semantic associations** (latent semantic space).
- Compute document similarity based on the **inner product** in this **latent semantic space**

Goals of LSI

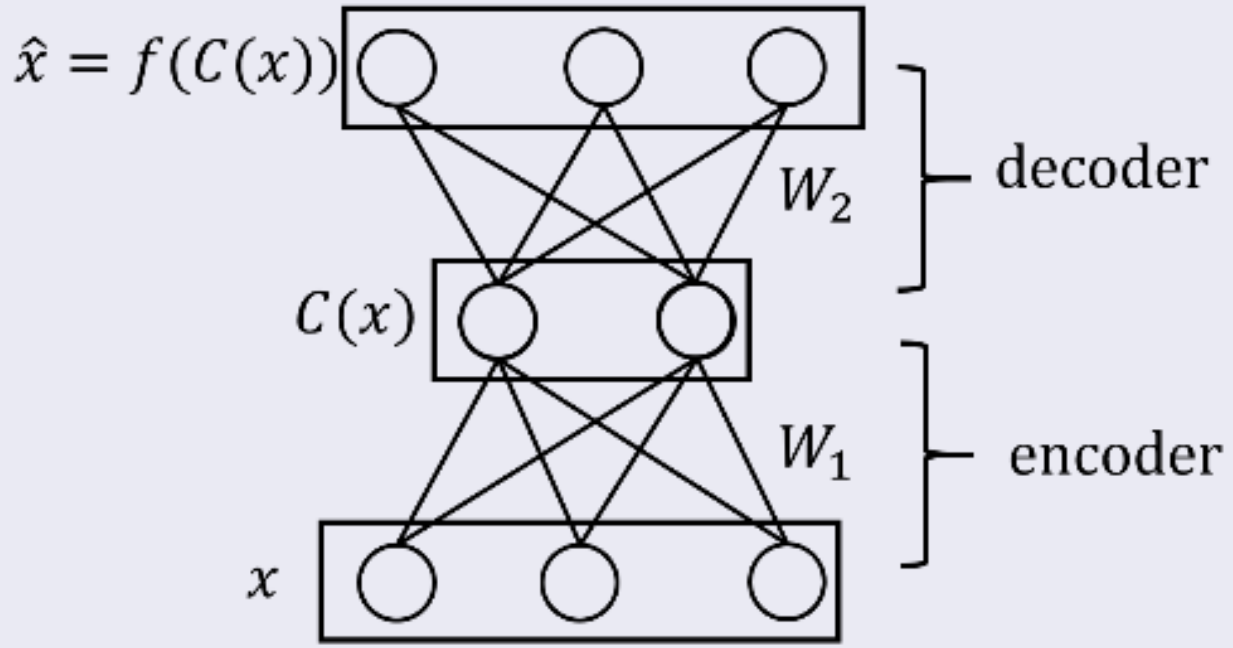
- LSI takes documents that are semantically similar (= talk about the same topics), but are not similar in the vector space (because they use different words) and re-represents them in a reduced vector space in which they have higher similarity.

- Similar terms map to similar location in low dimensional space
- Noise reduction by dimension reduction

Singular Value Decomposition(SVD)



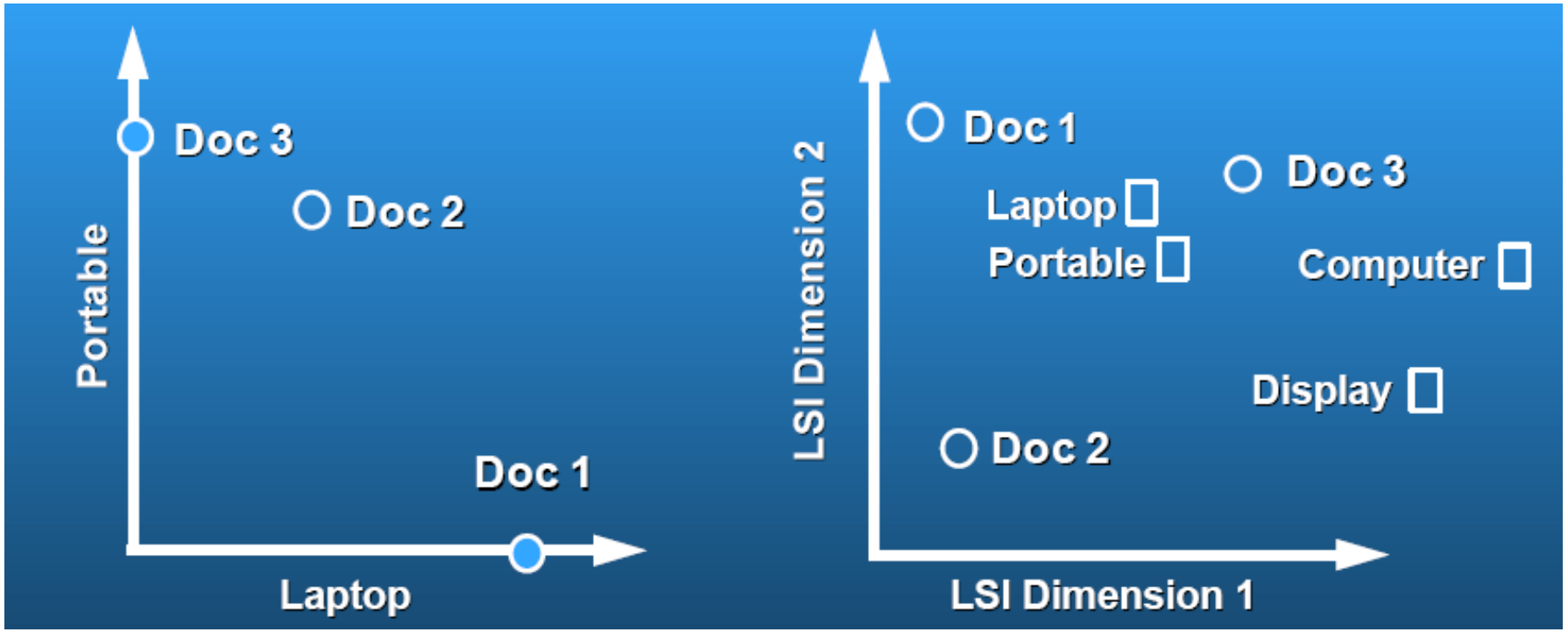
Relation to Autoencoder/PCA



- The bottom layer and the top layer has the same number of neurons

Latent Semantic Analysis

- **Latent semantic space**: illustrating



Latent Semantic Indexing LSI

	Doc1	Doc2	Doc3
LSI Dim1	0.1	0.2	2.2
LSI Dim2	2.5	0.2	1.9

Tooling

gensim: topic modeling for humans
- Free python library
- Memory independent
- Distributed computing

http://radimrehurek.com/gensim

Stanford Topic Modeling Toolbox

http://nlp.stanford.edu/software/tmt

MAchine **L**earning for **L**anguag**E** **T**oolkit (MALLET) is a Java-based package for:
- statistical natural language processing
- document classification
- Clustering
- topic modeling
- information extraction
- and other machine learning applications to text.

http://mallet.cs.umass.edu

Implementing LSI with gensim