Empirical Methods in Natural Language Processing
Lecture 12
Text Classification and Clustering

Philipp Koehn

14 February 2008

Type of learning problems

- *Supervised* learning
  - labeled training data
  - methods: HMM, naive Bayes, maximum entropy, transformation-based learning, decision lists, ...
  - example: language modeling, POS tagging with labeled corpus
- *Unsupervised* learning
  - labels have to be automatically discovered
  - method: **clustering** (this lecture)

Semi-supervised learning

- Some of the training data is labeled, vast majority is not
- *Boostrapping*
  - train initial classifier on labeled data
  - label additional data with initial classifier
  - iterate
- **Active learning**
  - train initial classifier with confidence measure
  - request from human annotator to label most informative examples

Goals of learning

- **Density estimation**: $p(x)$
  - learn the distribution of a random variable
  - example: language modeling
- **Classification**: $p(c|x)$
  - predict correct class (from a finite set)
  - example: part-of-speech tagging, word sense disambiguation
- **Regression**: $p(x,y)$
  - predicting a function $f(x) = y$ with real-numbered input and output
  - rare in natural languages (words are discrete, not continuous)

Text classification

- Classification problem
- First, supervised methods
  - the usual suspects
  - classification by language modeling
- Then, unsupervised methods
  - clustering

The task

- The task
  - given a set of documents
  - sort them into categories
- Example
  - sorting news stories into: *POLITICS*, *SPORTS*, *ARTS*, etc.
  - classifying job adverts into job types: *CLERICAL*, *TEACHING*, ...
  - filtering email into *SPAM* and *NO-SPAM*

The usual approach

- Represent document by *features*
  - words
  - bigrams, etc.
  - word senses
  - syntactic relations
- Learn a model that predicts a category using the features
  - *naive Bayes* $\text{argmax}_c p(c) \; \prod_i p(c|f_i)$
  - *maximum entropy* $\text{argmax}_c \frac{1}{Z} \prod_i \lambda_i^{f_i}$
  - *decision/transformation rules* $\{f_0 \rightarrow c_j, ..., f_n \rightarrow c_k\}$
- Set-up very similar to *word sense disambiguation*

Language modeling approach

- Collect documents for each class
- Train a *language model* p^c_{LM} for each class c separately
- Classify a new document d by

$$\text{argmax}_c p^c_{LM}(d)$$

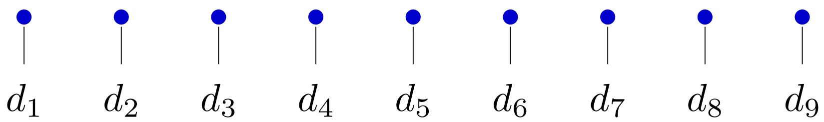
- Intuition: which language model most likely produces the document?
- Effectively uses words and n-gram features

Clustering

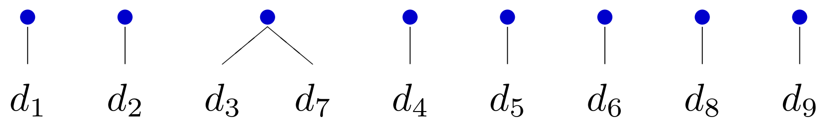
- Unsupervised learning
  - *given*: a set of documents
  - *wanted*: grouping into appropriate classes
- **Agglomerative clustering**
  - group the two most similar documents together
  - repeat

Agglomerative clustering

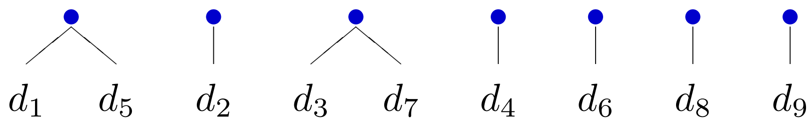
- Start: 9 documents, 9 classes



- Documents d_3 and d_7 are most similar

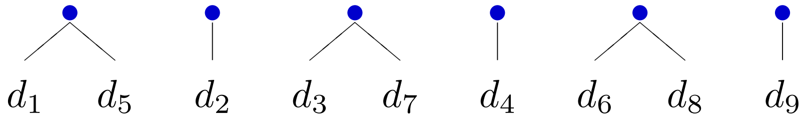


- Documents d_1 and d_5 are most similar

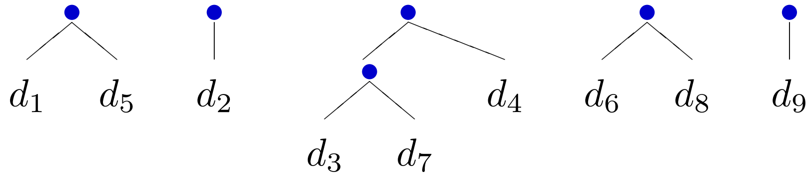


Agglomerative clustering (2)

- Documents d_6 and d_8 are most similar

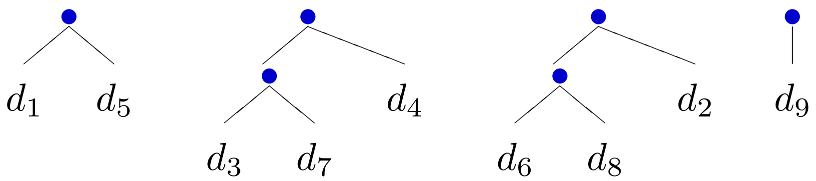


- Document d_4 and class $\{d_8, d_6\}$ are most similar

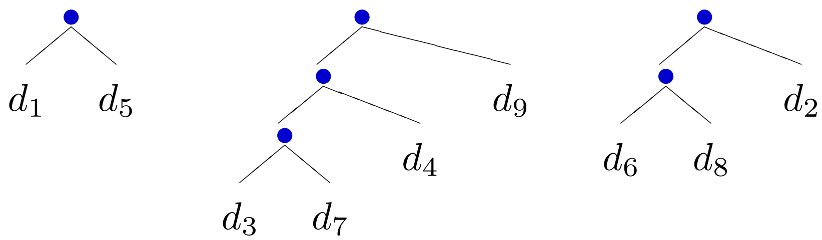


Agglomerative clustering (3)

- Document d_2 and class $\{d_6, d_8\}$ are most similar

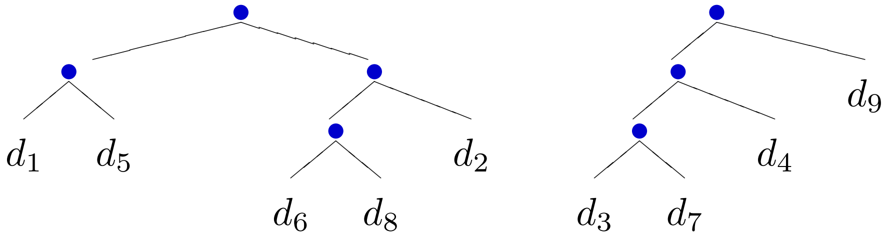


- Document d_9 and class $\{d_3, d_4, d_7\}$ are most similar



Agglomerative clustering (4)

- Class $\{d_1, d_5\}$ and class $\{d_2, d_6, d_8\}$ are most similar



- If we stop now, we have two classes

Similarity

- We loosely used the concept **similarity**
- How do we know how similar two documents are?
- How do we **represent** documents in the first place?

Vector representation of documents

Documents are represented by a **vector of word counts**.

Example document
Manchester United won 2 – 1 against Chelsea , Barcelona tied Madrid 1 – 1 , and Bayern München won 4 – 2 against Nürnberg

The word counts may be **normalized**, so all the vector components add up to one.

Manchester	1	0.04
United	1	0.04
won	2	0.08
2	2	0.08
–	3	0.12
1	3	0.12
against	2	0.08
Chelsea	1	0.04
,	2	0.08
Barcelona	1	0.04
tied	1	0.04
Madrid	1	0.04
and	1	0.04
Bayern	1	0.04
München	1	0.04
4	1	0.04
Nürnberg	1	0.04

Similarity

- A popular similarity metric for vectors is the **cosine**

$$\text{sim}(\vec{x}, \vec{y}) = \frac{\sum_{i=1}^{m} x_i \times y_i}{\sqrt{\sum_{i=1}^{m} x_i \times \sum_{i=1}^{m} y_i}} = \vec{x} \cdot \vec{y}$$

- We also need to define the similarity between
  - a document and a class
  - two classes

Similarity with classes

- **Single link**
  – merge two classes based on similarity of their *most* similar members

- **Compete link**
  – merge two classes based on similarity of their *least* similar members

- **Group average**
  – define class vector, or **center of class**, as

$$\vec{c} = \frac{1}{M} \sum_{\vec{x} \in c} \vec{x}$$

  – compare with other vectors using similarity metric

Additional Considerations

- **Stop words**
  – words such as *and* and *the* are very frequent and not very informative
  – we may want to ignore them

- **Complexity**
  – at any point in the clustering algorithm, we have to compare every document with every other document
  → complexity **quadratic** with the number of documents $O(n^2)$

- When do we stop?
  – when we have a pre-defined number of classes
  – when the lowest similarity is higher than a pre-defined threshold

Other clustering methods

- **Top-down hierarchical** clustering, or **divisive** clustering
  - start with one class
  - divide up classes that are least **coherent**

- **K-means** clustering
  - create initial clusters with arbitrary *center of cluster*
  - assign documents to the cluster with the closests center
  - compute *center of cluster*
  - iterate until convergence