Lecture 4: EITN01 Web Intelligence and Information Retrieval

Anders Ardö

EIT – Electrical and Information Technology, Lund University

February 6, 2013

Outline

PageRank

$$PR(u)_t = d \sum_{v \in B_u} \frac{PR(v)_{t-1}}{N_v} + (1-d)E(u)$$

- Random surfer model
  - Click on a random link in the page
  - Eventually gets bored and jumps to a random page
- Converges to a stable solution
- Problems
  - size of the Web
  - pages without links - 'dangling pages' (rank sinks)
  - converging
  - link-spamming

PageRank + TF*IDF

Relevance ranking

Combine PageRank with vector space model

$$PR(D) * sim(Q, D)$$

or

$$f(PR(D)) * sim(Q, D)$$

In practice

- proximity
- structure: title, link-anchor text
- metadata: keywords, description
- and $\cdots$

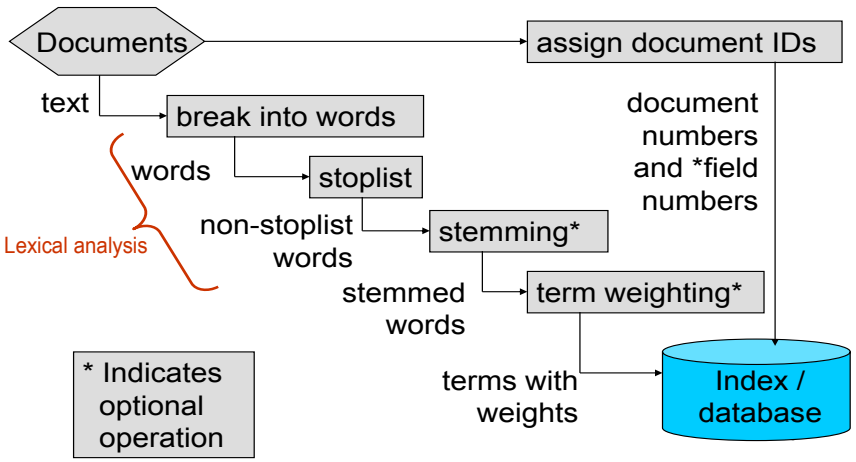
Previous lecture

- Text
  - Unicode character set (UTF-8) $>$ 100000 characters
  - Zipf's law $\Rightarrow$ Skewed distribution - stopwords
  - Heaps' law: *Vocabulary* $\sim n^{\beta}$; $\beta < 1$
  - Metadata
    - Author, source, length
    - Dublin Core Metadata Element Set
  - Word similarity models: Hamming Distance, Edit (Levenshtein) Distance
- Markup languages
- Text operations

Markup Languages

- SGML - Standard Generalized Markup Language
  - HTML - HyperText Markup Language
  - XML - eXtensible Markup Language
- TEX/ LATEX

Representation/Indexing (fig 1.2)



Lecture 4 agenda

Literature:

- LSI: http://en.wikipedia.org/wiki/Latent_semantic_analysis
- SVM: http://en.wikipedia.org/wiki/Support_vector_machine Overviews - not detailed math
- "Text Categorization with Support Vector Machines: Learning with Many Relevant Features." http://www.joachims.org/publications/joachims_98a.pdf

1. Reiteration
2. LSI (Latent Semantic Indexing)- concepts
3. SVM (Support Vector Machines) - classification
4. String matching - classification
5. Evaluation
6. Advice

Outline

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

Map terms $\Rightarrow$ topics
Map BOW $\Rightarrow$ Concepts

How LSI works

- Decompose term-document matrix into three other special matrices by Singular Value Decomposition (SVD)
- SVD (linear algebra)
  - similar to Principal Component Analysis (PCA) and Factor analysis
  - focuses on eigenanalyses
- The matrices show a breakdown of the original relationships into linearly independent components
- Many of these components are very small and can be ignored - leading to an approximate model that contains fewer dimensions.

What LSI achieves

- We accomplish more than dimensionality reduction here:
  - Docs with lots of overlapping terms stay together
  - Terms from these docs also get pulled together.
- Example: The synonymous terms car and automobile get pulled together because both co-occur in docs with tires, radiator, and cylinder
- Car and automobile are associated indirectly through 2nd order co-occurrence
- Fewer dimensions, more 'collapsing of axes', better recall, worse precision
- More dimensions, less collapsing, worse recall, better precision

LSI - SVD

M x N matrix A (term/document) $\rightarrow$ factorization (SVD):

$$A = U\Sigma V^T$$

- The columns of U are orthogonal eigenvectors of AA^T
- Σ is a diagonal matrix with singular values
- The columns of V are orthogonal eigenvectors of A^TA

LSI Interlude

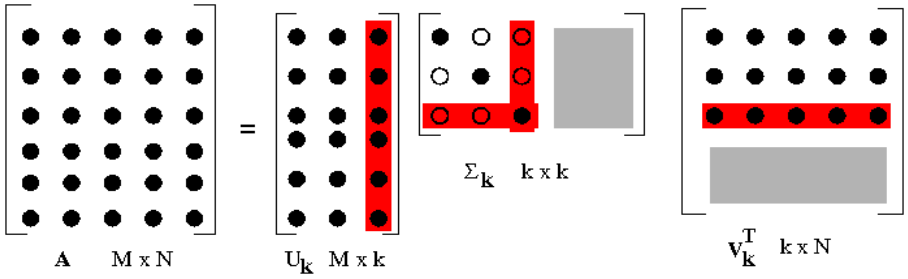
$AA^T = U\Sigma V^T V\Sigma U^T = U\Sigma^2 U^T$

entry (i, j) measures overlap ith and jth terms in documents

- Synonymy
- (Polysemy)

LSI - reduced SVD

- Reduce dimensionality => retain only k largest singular values
- Saved space



- M x N matrix A (term/document) $\rightsquigarrow$ reduced SVD:

$$A \approx A_k = U_k \Sigma_k V_k^T$$

LSI

- Performs a reduced dimension (k) approximation of term/document matrix
- Low-dimensional space reflects semantic associations
- Dimension reduction
  - Documents represented by vector of dimension k
    $$V_k^T = \Sigma_k^{-1} U_k^T doc$$
  - Map also queries into the low-dimensional space
    $$q_k = \Sigma_k^{-1} U_k^T q$$
  - Queries become k-dimensional and not sparse
  - Compute similarity in this low-dimensional (k) space

Example - SVDLIBC

- `svd -d 100 -o SVDres termDocMatrix.txt`
- perform an reduced SVD on a matrix `termDocMatrix.txt`
- $k = 100$ dimensions
- results
  - U_k^T in `SVDres-Ut`
  - Σ_k in `SVDres-S` (just the singular values)
  - V_k^T in `SVDres-Vt`
- uses file formats *Sparse text* or *Dense text*

SVDLIBC - Sparse text format

numRows numCols totalNonZeroValues
for each column:
  numNonZeroValues
  for each non-zero value in the column:
    rowIndex value

```
4 3 6
2
0 2.3
2 3.8
1
1 1.3
3
0 4.2
1 2.2
2 0.5
```


SVDLIBC - Dense text format

numRows numCols
for each row:
  values

```
4 3
2.3   0   4.2
 0   1.3  2.2
3.8   0   0.5
 0    0    0
```


LSI - SVDLIBC - example

1. Doc weight vectors $\rightarrow$ LSI concept space ($V_k^T = \Sigma_k^{-1} U_k^T doc$)
2. Query weight vectors $\rightarrow$ LSI concept space ($q_k = \Sigma_k^{-1} U_k^T q$)
3. Calculate similarity in concept space
$$simCos((\Sigma_k^{-1} U_k^T doc), (\Sigma_k^{-1} U_k^T q))$$

Help functions

- functions `saveWeightMatrixCompressed()`, `readUt()`, `readS()` writes and reads the matrix formats
  - need terms and documents to be mapped to numbers
  - handled by function `mapDocsTerms2numbers()`
- function `runSVD($dimensions)` executes SVDLIBC

LSI - Concept extraction

use rows of $\Sigma_k^{-1} U_k^T$ as concepts

Concept 1		Concept 2	
carlstrom	0.0354	regia	0.0523
rick	0.0354	oct	0.0521
amelnx	0.0354	chrisp	0.0314
advmar	0.0354	problems	0.0273
cuttings	0.0329	pm	0.0265
september	0.0322	ip-forum	0.0264
miller	0.0265	stratification	0.0261
re	-0.0287	uk	-0.0267
wants	-0.0303	bladderwort	-0.0361
aquatic	-0.0363	cuttings	-0.0317
rotundifolia	-0.0371		
bladderwort	-0.0605		

HARD to interpret

Outline

Why automated classification?

- Information explosion
  - Documents increasingly available electronically
  - Lots of unstructured full-text documents on the Web
- High cost of manual classification (1-2 / hour)
- Challenging research issue
- Fun!

Text classification

- Goal: classify documents into predefined categories
- Examples
  - Subject classification: 'business', 'sports', 'engineering', ...
  - Review classification: 'positive' or 'negative'
  - Web page classification: 'Personal homepage' or others
- One approach: supervised machine learning ($\Rightarrow$ SVM)
  - Each predefined category needs a set of training documents
  - From training sets train a classifier
  - Use classifier to classify new documents
- Many other approaches
  (Naive Bayes, Neural Networks, Nearest neighbor, String matching, ...)

Automated Classification technologies

- Machine learning methods
  - Statistical models (Bayes, **SVM**, ...)
  - ANN
- Information Retrieval methods
  - Clustering (no predefined categories)
- Library Science methods
  - **String matching** + Thesaurus

Overview - Machine learning methods

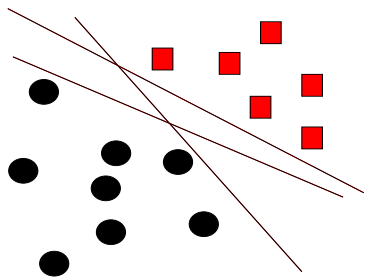
Work-flow:

1. Text preprocessing and indexing
   - As before
2. Training
   - Need a training set with positive and negative example documents
   - Train a classifier
3. Classification
   - Apply the trained classifier to new documents

SVM

- Developed by Vapnik 1992
- Classification for linear (and non-linear) problems
  - "Kernels" handle non-linear problems (by mapping to linear case)
- Machine learning
- Data represented as n-dimensional vectors (vector space model)
- Need a training set with positive and negative documents
- General classifier
- Decision: yes/no
- Finds the optimal hyper-plane for linearly separable patterns
- Can be extended to multiclass/hierarchical classification
- Applications
  - pattern recognition, object recognition, speaker identification, text classification, ...

SVM

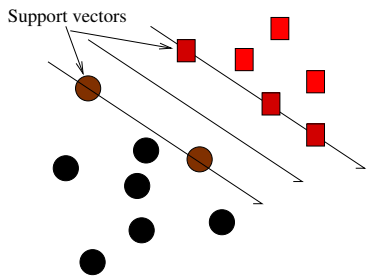


- Which hyper-plane?
- lots of possible solutions
- SVM finds an optimal solution

SVM

- Pick hyper-plane with max distance to nearest data point
- Support vectors = data points closest to decision hyper-plane
- They are most difficult to classify
- They have a direct influence on the optimum decision hyper-plane
- Independent of dimensionality
- Non-linear separability?
  - Use 'kernel-trick'
    map vectors into a new (linear) space - Kernel functions

SVM



- SVM maximize the margin around the separating hyper-plane
- Decision function specified by support vectors (from training examples)
- Quadratic programming problem

Hot text classification method

Why SVM for text categorization?

Advantages

- "Most popular and effective method"
- High dimensionality input
- Uses all features - no feature selection
- Bag-Of-Words model - document vectors
- Sound mathematical theory for optimal decision function
- Performs well when collection characteristics does not change
- Fast once trained

Problems

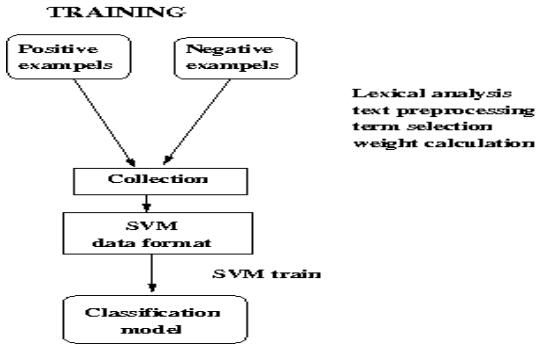
- Requires training examples
- Language
- Depends on a relatively homogeneous collection
- Sensitive for selection of negative examples
- Error propagation for deep classification hierarchies
- One classifier per class

Tools - libsvm

- Libsvm implementation of SVMs in C
- Homepage http://www.csie.ntu.edu.tw/∼cjlin/libsvm
- Documentation in `S:\Documentation\svmlib\`
- We will use
  - learning module: `svm-train`
  - classification module: `svm-predict`

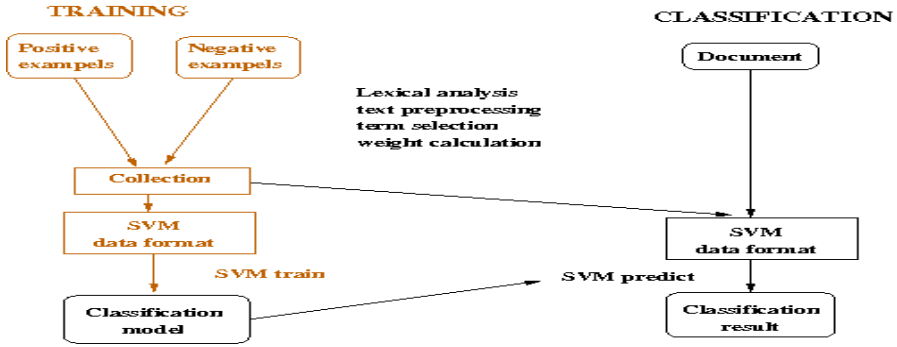
Tools - libsvm - Example - Train

- To train a classifier from `train.data`
  - `svm-train -t 2 -g 0.6 train.data model.data`
  - Use kernel: radial basis function ($t = 2$) with $\gamma = 0.6$
  - SVM classifier model in `model.data`



Tools - libsvm - Example - Classify

- To classify new documents in `test.data`
  - `svm-predict test.data model.data test.result`
  - use model from training: `model.data`
  - Output (in `test.result`)
    - -1 → negative class
    - 1 → positive class



Tools - libsvm - file formats

Training data format - `train.data`

```
One line per training example
<target> <featureNo>:<value> <featureNo>:<value> ... <featureNo
```

```
<target> is 1 for positive; -1 for negative examples
<featureNo> is feature number
<value> is a real value
```

Must be in featureNo order

Example file:

`-1 1:0.43 3:0.12 9284:0.2`

Negative example (target -1) with weights
0.43 for feature 1
0.12 for feature 3
0.2 for feature 9284

Tools - libsvm - file formats

Classification (test) data format - `test.data`

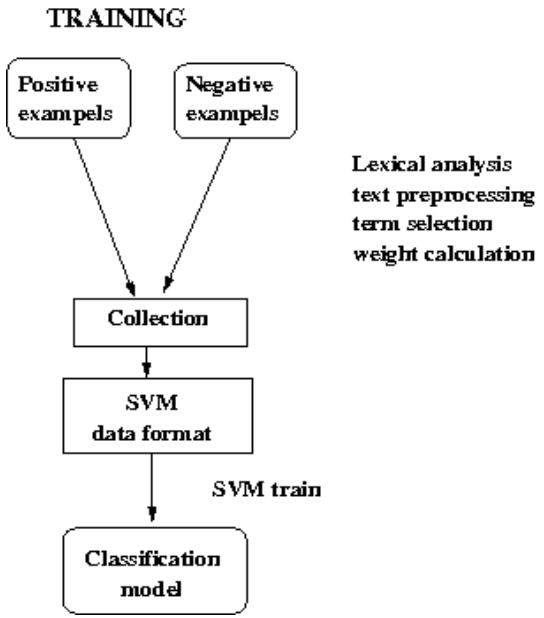
```
same format as the training examples with 0 as target label
```

```
0 10:0.3 37:0.12 42:0.23 284:0.21
0 101:0.02 157:0.912 478:0.99 2864:0.51
```

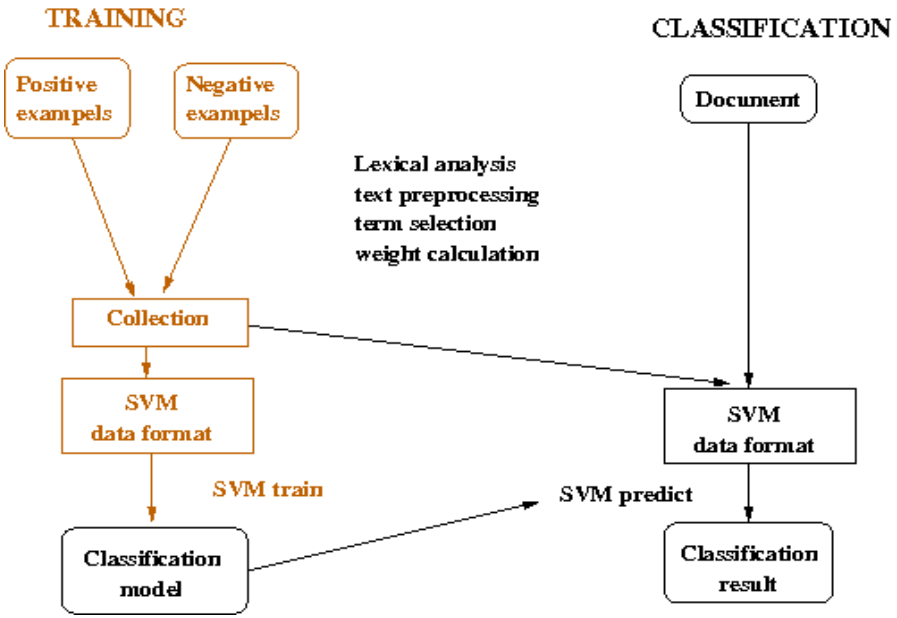
Classification (test) result format - `test.result`

```
one line per test example
containing the label for that test example
```

Training process - SVM

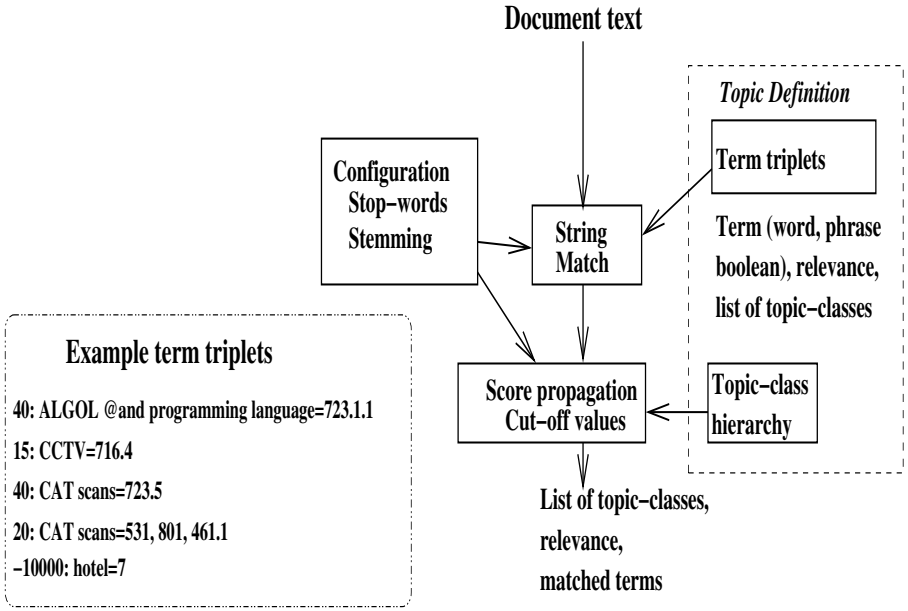


Classification process - SVM



Outline

Classification process



String matching

Thesauri based

- Reuse intellectual effort
- Topic terms (features) from thesaurus
  - ... are they present in the text?
    - ... relevance: how many; where in the text (document structure)

$$\text{Concept\_score} =$$

$$\sum_{all\ locations} \left(\sum_{all\ terms} (hits[location_j][term_i] * weight[term_i] * weight[location_j]) \right)$$

or

$$\sum_{all\ terms} \left(\sum_{all\ matches} \frac{weight[term_i]}{\log(k * position[term_i][match_j]) * proximity[term_i][match_j]} \right)$$

Normalize with respect to document size

Why String matching for text categorization?

Advantages

- Reuse intellectual effort
- Can take advantage of document structure
- Feature selection by thesaurus
- Language
- No training
- Deep hierarchies
- Multiclass classification

Problems

- No context for topic terms
- Stopwords can cause trouble
- Relies on a good thesaurus
- No generalization

Outline

Evaluation challenge

Comparing human assigned classes to automated classification

- Collection policies
- Users vs indexers
- Inter- and intra-indexers consistency
- Availability of representative pre-classified collections

Hard to do good evaluations

Evaluation

- SVM
  - Most evaluations done in “lab-like environments”
  - Very good - 70 - 90 % correctness
  - Popular
- String matching
  - Few evaluations done
  - Good - 60 - 90 % correctness

Examples:

Precision for classification of Compendex bibliographic records:

SVM	0.74 - 0.91
String match	0.26 - 0.97

Outline

Advice I

- Homogeneous collection
- Good training examples (both positive and negative)
- Shallow hierarchy

use SVM

- Mixed collection
- Good thesaurus with subject terms
- Multiple classes in a hierarchy

use String match

Parameters

- text preprocessing
- document vector values
- kernel
- gamma, coef0, cost, degree, nu, epsilon, shrinking, degree, ...

- text preprocessing
- add synonyms
- word sense disambiguation
- word weights
- cut-off value

Advice II

- Careful with text preprocessing (stopwords and stemming)
- Hard to do a good evaluation
- Learn strengths and weaknesses
- Experiment!
- There is no “fit all cases best” solution
- Not perfect
  - ... but useful