Automated classification

insights into benefits, costs and lessons learned

Anders Ardö
Anders.Ardo@eit.lth.se
http://www.eit.lth.se/staff/anders.ardo

EIT – Electrical and Information Technology,
Lund University, Sweden

International UDC Seminar 2009:
Classification at a Crossroads
The Hague 29-30 October 2009

Outline

1. Background
2. SVM (Support Vector Machines)
3. String matching
4. Evaluation
5. Lessons learned
6. References

Why automated classification?

- Information explosion

Why automated classification?

- Information explosion
  - Documents increasingly available electronically

Why automated classification?

- Information explosion
  - Documents increasingly available electronically
  - Lots of unstructured full-text documents on the Web

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)

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

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!

History ...

(Personal - NetLab)

1992 Automated classification of WAIS databases using 2 top levels of UDC

1993 Demonstrated at SIGWAIS/SIGNIDR III conference

1997 Automated classification of Engineering Web resources using Ei

2000 EU project DESIRE: toolkit (Matcher)

2003 EU project ALVIS: Matcher + Crawler => Focused Web crawler (Combine)

2007 PhD thesis: "*Automated Subject Classification of Textual Documents in the Context of Web-based Hierarchical Browsing*", Koraljka Golub

2009 Vertical Search Engines Demo

Automated Classification technologies

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

Outline

1. Background
2. SVM (Support Vector Machines)
3. String matching
4. Evaluation
5. Lessons learned
6. References

SVM

- Developed by Vapnik 1992

SVM

- Developed by Vapnik 1992
- Classification for linear (and non-linear) problems

SVM

- Developed by Vapnik 1992
- Classification for linear (and non-linear) problems
  - “Kernels” handle non-linear problems (by mapping to linear case)

SVM

- Developed by Vapnik 1992
- Classification for linear (and non-linear) problems
  - “Kernels” handle non-linear problems (by mapping to linear case)
- Machine learning

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)

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

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

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

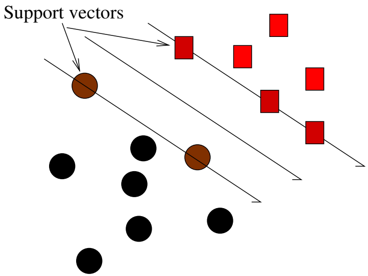
SVM

- 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

SVM

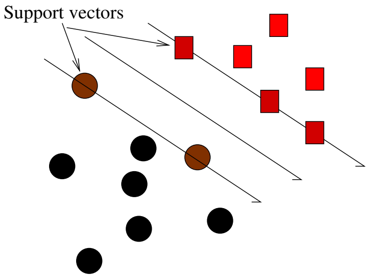
- Developed by Vapnik 1992
- Classification for linear (and non-linear) problems
    - "Kernels" handle non-linear problems (by mapping to linear case)
- Machine learning
- Data represented as n-dimensional vectors (vector space model)
- Need a training set with positive and negative documents
- General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

SVM



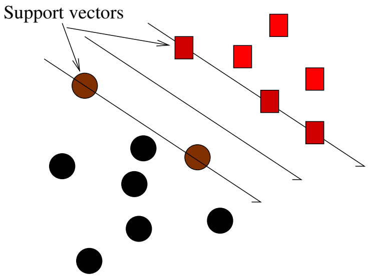
- Efficiently handles $\sim$ 10 000 dimensions given that input vectors are sparse

SVM



- Efficiently handles $\sim$ 10 000 dimensions given that input vectors are sparse
- Decision function specified by support vectors (from training examples)

SVM



- Efficiently handles $\sim$ 10 000 dimensions given that input vectors are sparse
- Decision function specified by support vectors (from training examples)
- SVM maximize the margin around the separating hyper-plane

Why SVM for text categorization?

Advantages

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

Why SVM for text categorization?

Advantages

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

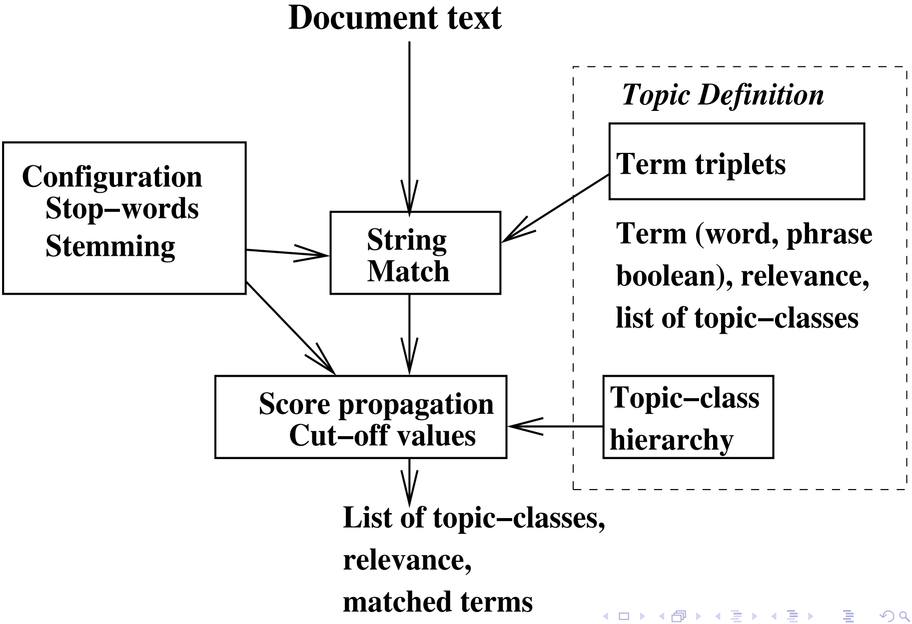
Problems

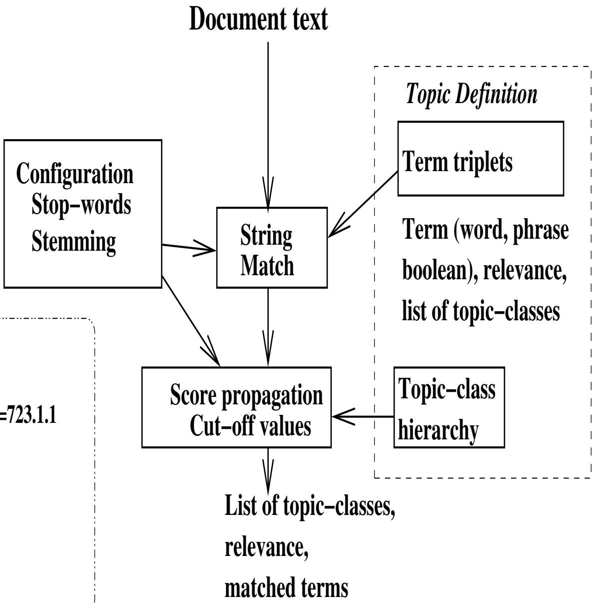
- Requires training examples
- Language
- Depends on a relatively homogeneous collection
- Sensitive for selection of negative examples
- Error propagation for deep classification hierarchies
- One classifier per class

Outline

1. Background
2. SVM (Support Vector Machines)
3. String matching
4. Evaluation
5. Lessons learned
6. References

Classification process





Example term triplets

40: ALGOL @and programming language=723.1.1

15: CCTV=716.4

40: CAT scans=723.5

20: CAT scans=531, 801, 461.1

−10000: hotel=7

Thesauri based

- Reuse intellectual effort

Thesauri based

- Reuse intellectual effort
- Topic terms (features) from thesaurus

Thesauri based

- Reuse intellectual effort
- Topic terms (features) from thesaurus
    - ... are they present in the text?

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)

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)

$$\text{Relevance\_score} =$$

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

Normalize with respect to document size

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)

$$\text{Relevance\_score} =$$

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

or

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

Normalize with respect to document size

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)

$$\text{Relevance\_score} =$$

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

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

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

1 Background

2 SVM (Support Vector Machines)

3 String matching

4 Evaluation

5 Lessons learned

6 References

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

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

Evaluation

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

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:

1: Precision for classification of Compendex bibliographic records:

SVM	0.74 - 0.91
String match	0.26 - 0.97

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:

1: Precision for classification of Compendex bibliographic records:

SVM	0.74 - 0.91
String match	0.26 - 0.97

2: Depends on the hierarchical depth of the classification

Correct to	**String match**	**SVM**
3 levels	0.71p	0.61p
2 levels	0.87p	0.81p
top level	p	p

Outline

1. Background
2. SVM (Support Vector Machines)
3. String matching
4. Evaluation
5. Lessons learned
6. References

Lessons learned I

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

use SVM

Lessons learned 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

Lessons learned 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

Lessons learned 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, ...

Lessons learned 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

Lessons learned II

- Careful with text preprocessing (stopwords and stemming)

Lessons learned II

- Careful with text preprocessing (stopwords and stemming)
- Hard to do a good evaluation

Lessons learned II

- Careful with text preprocessing (stopwords and stemming)
- Hard to do a good evaluation
- Learn strengths and weaknesses

Lessons learned II

- Careful with text preprocessing (stopwords and stemming)
- Hard to do a good evaluation
- Learn strengths and weaknesses
- Experiment!

Lessons learned 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

Lessons learned 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

Lessons learned 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

Idea - compromise and use all

1. Start with a reasonable classification system and thesaurus

Idea - compromise and use all

1. Start with a reasonable classification system and thesaurus
2. Collect documents and classify by String Matching

Idea - compromise and use all

1. Start with a reasonable classification system and thesaurus
2. Collect documents and classify by String Matching
3. Use result to generate SVM training set

Idea - compromise and use all

1. Start with a reasonable classification system and thesaurus
2. Collect documents and classify by String Matching
3. Use result to generate SVM training set
4. Train SVM classifier

Idea - compromise and use all

1. Start with a reasonable classification system and thesaurus
2. Collect documents and classify by String Matching
3. Use result to generate SVM training set
4. Train SVM classifier
5. Reclassify all documents

Idea - compromise and use all

1. Start with a reasonable classification system and thesaurus
2. Collect documents and classify by String Matching
3. Use result to generate SVM training set
4. Train SVM classifier
5. Reclassify all documents
6. Manually inspect results and update SVM training set

Idea - compromise and use all

1. Start with a reasonable classification system and thesaurus
2. Collect documents and classify by String Matching
3. Use result to generate SVM training set
4. Train SVM classifier
5. Reclassify all documents
6. Manually inspect results and update SVM training set
7. Go-to 3 until result good enough

Idea - compromise and use all

1. Start with a reasonable classification system and thesaurus
2. Collect documents and classify by String Matching
3. Use result to generate SVM training set
4. Train SVM classifier
5. Reclassify all documents
6. Manually inspect results and update SVM training set
7. Go-to 3 until result good enough
8. Production level service

Idea - compromise and use all

1. Start with a reasonable classification system and thesaurus
2. Collect documents and classify by String Matching
3. Use result to generate SVM training set
4. Train SVM classifier
5. Reclassify all documents
6. Manually inspect results and update SVM training set
7. Go-to 3 until result good enough
8. Production level service

Outline

1 Background

2 SVM (Support Vector Machines)

3 String matching

4 Evaluation

5 Lessons learned

6 References

References/More info

- This presentation:
  http://combine.it.lth.se/UDCseminar2009/

- Koraljka Golub PhD thesis: "*Automated Subject Classification of Textual Documents in the Context of Web-based Hierarchical Browsing*"
- Combine focused crawler
  tools download: *http://combine.it.lth.se/#downloads*
  documentation on automated classification:
  http://combine.it.lth.se/documentation/DocMain/node6.html
- Demonstrators (incl UDC classifiers):
  http://dbkit05.eit.lth.se/exp/Demos/