Text Classification

Topic modelling and dimensionality reduction for documents

M. Vazirgiannis

Jan 2017

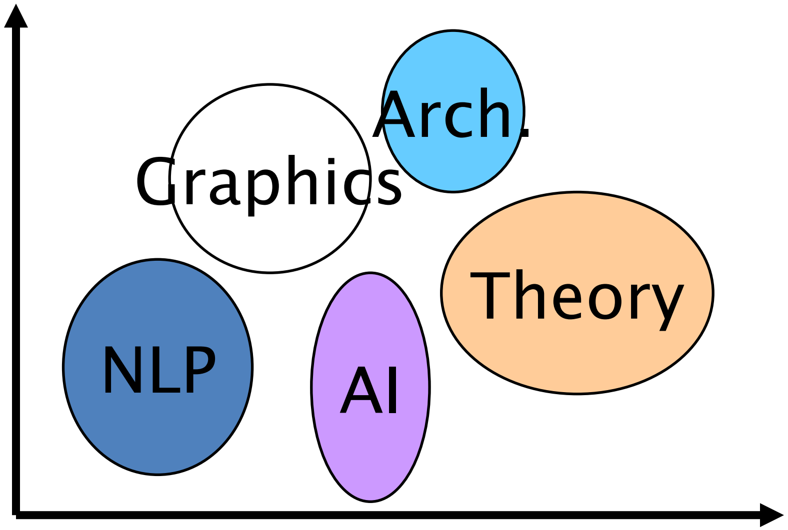
CATEGORIZATION / CLASSIFICATION

Given:

- A description of an instance, $x \in X$, where X is the *instance language* or *instance space*.
  - E.g: how to represent text documents.
- set of categories $C = \{c_1, c_2, \ldots, c_n\}$

Determine:

- The category of x: $c(x) \in C$, where $c(x)$ is a *categorization function* whose domain is X and whose range is C.



EXAMPLES OF TEXT CATEGORIZATION

LABELS=BINARY

“spam” / “not spam”

LABELS=TOPICS

“finance” / “sports” / “asia”

LABELS=OPINION

“like” / “hate” / “neutral”

LABELS=AUTHOR

“Shakespeare” / “Marlowe” / “Ben Jonson”

The Federalist papers

Methods

Supervised learning of document-label assignment function: Autonomy, Kana, MSN, Verity, …

- Naive Bayes (simple, common method)
- k-Nearest Neighbors (simple, powerful)
- Support-vector machines (new, more powerful)

… plus many other methods

- No free lunch: requires hand-classified training data
- But can be built (and refined) by amateurs

Bayesian Methods

- Learning and classification based on probability theory
- Bayes theorem plays a critical role

$$P(C,X) = P(C \mid X)P(X) = P(X \mid C)P(C)$$

- Build a *generative model* that approximates how data is produced
- Uses *prior* probability of each category given no information about an item.
- Categorization produces a *posterior* probability distribution over the possible categories given a description of an item.

$$P(C \mid X) = \frac{P(X \mid C)P(C)}{P(X)}$$

Maximum a posteriori Hypothesis

$$h_{MAP} \equiv \underset{h \in H}{\operatorname{argmax}} P(h \mid D)$$

$$h_{MAP} = \underset{h \in H}{\operatorname{argmax}} \frac{P(D \mid h) P(h)}{P(D)}$$

<u>Max Likelihood</u>

If all hypotheses are a priori equally likely, we only need to consider the $P(D|h)$ term:

$$h_{ML} \equiv \underset{h \in H}{\operatorname{argmax}} P(D \mid h)$$

Naive Bayes Classifiers

Task: Classify a new instance based on a tuple of attribute values

$$\langle x_1, x_2, \ldots, x_n \rangle$$

$$c_{MAP} = \operatorname*{argmax}_{c_j \in C} P(c_j \mid x_1, x_2, \ldots, x_n)$$

$$c_{MAP} = \operatorname*{argmax}_{c_j \in C} \frac{P(x_1, x_2, \ldots, x_n \mid c_j) P(c_j)}{P(c_1, c_2, \ldots, c_n)}$$

$$c_{MAP} = \operatorname*{argmax}_{c_j \in C} P(x_1, x_2, \ldots, x_n \mid c_j) P(c_j)$$

Naïve Bayes Classifier: Assumptions

$$P(c_j)$$

Can be estimated from the frequency of classes in the training examples.

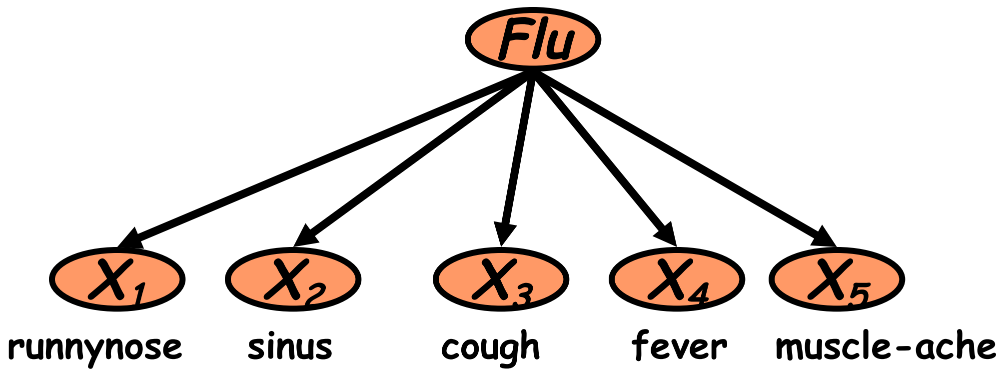
$$P(x_1, x_2, \ldots, x_n | c_j)$$

Need very, very large number of training examples

$\Rightarrow$ Conditional Independence Assumption:

Assume that the probability of observing the conjunction of attributes is equal to the product of the individual probabilities.

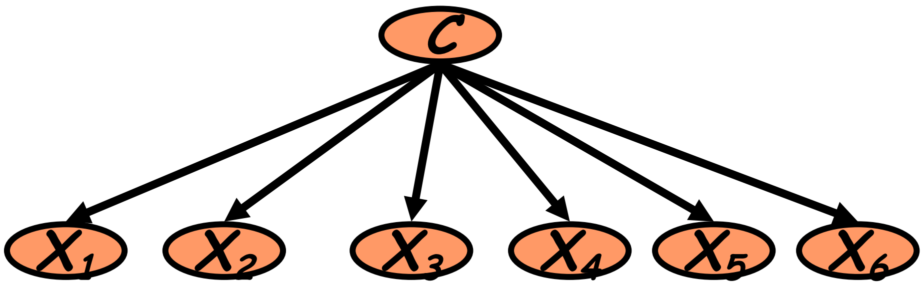
The Naïve Bayes Classifier



Conditional Independence Assumption:
features are independent of each other given the class:

$$P(X_1,\ldots,X_5 \mid C) = P(X_1 \mid C) \bullet P(X_2 \mid C) \bullet \cdots \bullet P(X_5 \mid C)$$

Learning the Model



Common practice:maximum likelihood

simply use the frequencies in the data

$$\hat{P}(c_j) = \frac{N(C = c_j)}{N}$$

$$\hat{P}(x_i \mid c_j) = \frac{N(X_i = x_i, C = c_j)}{N(C = c_j)}$$

Using Naive Bayes Classifiers to Classify Text: Basic method

Attributes are *text positions, values are words*.

$$c_{NB} = \operatorname*{argmax}_{c_j \in C} P(c_j) \prod_i P(x_i \mid c_j)$$
$$= \operatorname*{argmax}_{c_j \in C} P(c_j) P(x_1 = \text{"our"} \mid c_j) \cdots P(x_n = \text{"text"} \mid c_j)$$

- Still too many possibilities
- Assume that classification is *independent* of the positions of the words
  - Use same parameters for each position

Naïve Bayes Posterior Probabilities

- Classification results of naïve Bayes (the class with maximum posterior probability) are usually fairly accurate.
- However, due to the inadequacy of the conditional independence assumption, the actual posterior-probability numerical estimates are not.

  Output probabilities are generally very close to 0 or 1.

Feature selection via Mutual Information

- We might not want to use all words, but just reliable, good discriminators
- In training set, choose *k* words which best discriminate the categories.
- One way is in terms of Mutual Information:

$$I(w,c) = \sum_{e_w \in \{0,1\}} \sum_{e_c \in \{0,1\}} p(e_w, e_c) \log \frac{p(e_w, e_c)}{p(e_w)p(e_c)}$$

For each word *w* and each category *c*

OTHER APPROACHES TO FEATURE SELECTION

- T-TEST
- CHI SQUARE
- TF/IDF

NAÏVE BAYES NOT SO NAIVE

- Naïve Bayes: First and Second place in KDD-CUP 97 competition, among 16 (then) state of the art algorithms
  - Robust to Irrelevant Features
  - Irrelevant Features cancel each other without affecting results
  - Instead Decision Trees & Nearest-Neighbor methods can heavily suffer from this.
- Very good in Domains with many equally important features
  - Decision Trees suffer from *fragmentation* in such cases – especially if little data
- A good baseline for text classification
- Optimal if the Independence Assumptions hold:
  - If assumed independence is correct, then it is the Bayes Optimal Classifier for problem
- Very Fast:
  - Learning with one pass over the data; testing linear in the number of attributes, and document collection size
- Low Storage requirements
- Handles Missing Values

OTHER CLASSIFICATION METHODS

K-NN

DECISION TREES

LOGISTIC REGRESSION

SUPPORT VECTOR MACHINES

REFERENCES

Mosteller, F., & Wallace, D. L. (1984). *Applied Bayesian and Classical Inference: the Case of the Federalist Papers* (2nd ed.). New York: Springer-Verlag.

P. Pantel and D. Lin, 1998. “SPAMCOP: A Spam classification and organization program”, In Proc. Of the 1998 workshop on learning for text categorization, AAAI

Sebastiani, F., 2002, “Machine Learning in Automated Text Categorization”, ACM Computing Surveys, 34(1), 1-47

Dim. Reduction–Eigenvectors

A: nxn matrix

- eigenvalues λ: $|A-\lambda I|=0$
- Eigenvectors x : $Ax=\lambda x$
- Matrix rank: # linearly independent rows or columns
- A real symmetric table A nxn can be expressed as: $A=U\Lambda U^T$
- U's columns are A's eigenvectors
- Λ' s diagonal contains A's eigenvalues
- $\mathbf{A=U\Lambda U^T=\lambda_1 x_1 x^T{}_1+\lambda_2 x_2 x^T{}_2+\ldots+\lambda_n x_n x^T{}_n}$
- $x_1 x^T{}_1$ represents projection via x_1 (λ_i eigenvalue, x_i eigenvector)
- Interpretations: xx^T vs. x^Tx

Singular Value Decomposition (SVD)

Eigen values and eigenvectors decomposition is applied to square matrices. For non square matrices we apply ***Singular Value Decomposition.***

Let **X** a **mxn table**, $X = U\Sigma V^T$

U : orthogonal mxm, its columns are the eigenvectors of XX^T.

U,V define orthogonal basis: $U^T U = VV^T = 1$

Σ: mxn contains A's singular values (square roots of XX^T eigenvalues)

V : nxn, its columns are the eigenvectors of $X^T X$

Singular Value Decomposition (SVD) - I

Proof:

$X = U\Sigma V^T$, $X^T = V\Sigma^T U^T$ =>

$XX^T = U\Sigma(V^T V)\Sigma U^T = U\Sigma\Sigma^T U^T$

Similarly: $X^T X = V\Sigma^T \Sigma V^T$

Therefore: U: eigenvectors of XX^{T} (V: eigenvectors of $X^T X$)

Σ: sqrt of the eigenvalues of XX^T

X k-dimensional representation: $X_k = U_k \Sigma_k V_k{}^T$

Singular Value Decomposition (SVD) - II

Matrix approximation $$X_k = U_k \Sigma_k {V_k}^T$$

The best rank k approximation Y' of a matrix X. (minimizing the Frobenius norm)

$$\|A\|_F^2 = \sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2 = \mathrm{trace}(AA^H) = \sum_{i=1}^{\min\{m,n\}} \sigma_i^2$$

where A^H transpose of A, σ_i are the singular values of A, and the trace function is used.

SVD application - Latent Structure in documents

- Documents are represented based on the Vector Space Model
- Vector space model consists of the keywords contained in a document.
- In many cases baseline keyword based performs poorly – not able to detect synonyms.
- Therefore document clustering is problematic
- Example where of keyword matching with the query: "IDF in computer-based information look-up"

	access	document	retrieval	information	theory	database	indexing	computer
Doc1	x	x	x			x	x	
Doc2				x	x			x
Doc3			x	x				x

Indexing by Latent Semantic Analysis (1990) Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, Richard Harshman, Journal of the American Society of Information Science

Latent Semantic Indexing (LSI) -I

- Finding similarity with exact keyword matching is problematic.
- Using SVD we process the initial document-term document.
- Then we choose the k larger singular values. The resulting matrix is of order k and is the most similar to the original one based on the Frobenius norm than any other k-order matrix.

Latent Semantic Indexing (LSI) - II

- The initial matrix is SVD decomposed as: $A=ULV^T$
- Choosing the top-k singular values from L we have:

$$A_k=U_kL_kV_k^T \ ,$$

- L_k is square kxk containing the top-k singular values of the diagonal in matrix L,
- U_k, the mxk matrix containing the first k columns in U (left singular vectors)
- V_k^T, the kxn matrix containing the first k lines of V^T (right singular vectors)

Typical values for $\varkappa$~200-300 (empirically chosen based on experiments appearing in the bibliography)

LSI capabilities

- Term to term similarity: $A_k A_k^T = U_k L_k^2 U_k^T$

  $A_k = U_k L_k V_t$

- Document-document similarity: $A_k^T A_k = V_k L_k^2 V_k^T$

- Term document similarity (as an element of the transformed – document matrix)

- Extended query capabilities transforming initial query q to q_n : $q_n = q^T U_k L_k^{-1}$

- Thus q_n can be regarded a line in matrix V_k

LSI – an example

LSI application on a term – document matrix

C1: Human machine Interface for Lab ABC computer application

C2: A survey of user opinion of computer system response time

C3: The EPS user interface management system

C4: System and human system engineering testing of EPS

C5: Relation of user-perceived response time to error measurements

M1: The generation of random, binary unordered trees

M2: The intersection graph of path in trees

M3: Graph minors IV: Widths of trees and well-quasi-ordering

M4: Graph minors: A survey

- The dataset consists of 2 classes, 1st: “human – computer interaction” (c1-c5) 2nd: related to graph (m1-m4). After feature extraction the titles are represented as follows.

Indexing by Latent Semantic Analysis (1990) Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, Richard Harshman, Journal of the American Society of Information Science

LSI – an example

	C1	C2	C3	C4	C5	M1	M2	M3	M4
human	1	0	0	1	0	0	0	0	0
Interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
User	0	1	1	0	1	0	0	0	0
System	0	1	1	2	0	0	0	0	0
Response	0	1	0	0	1	0	0	0	0
Time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
Survey	0	1	0	0	0	0	0	0	1
Trees	0	0	0	0	0	1	1	1	0
Graph	0	0	0	0	0	0	1	1	1
Minors	0	0	0	0	0	0	0	1	1

LSI – an example

$A=ULV^T$

A=

1	0	0	1	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
0	1	1	0	1	0	0	0	0
0	1	1	2	0	0	0	0	0
0	1	0	0	1	0	0	0	0
0	1	0	0	1	0	0	0	0
0	0	1	1	0	0	0	0	0
0	1	0	0	0	0	0	0	1
0	0	0	0	0	1	1	1	0
0	0	0	0	0	0	1	1	1
0	0	0	0	0	0	0	1	1

LSI – an example

$A=ULV^T$

$U=$

0.22	-0.11	0.29	-0.41	-0.11	-0.34	0.52	-0.06	-0.41	0	0	0
0.20	-0.07	0.14	-0.55	0.28	0.50	-0.07	-0.01	-0.11	0	0	0
0.24	0.04	-0.16	-0.59	-0.11	-0.25	-0.30	0.06	0.49	0	0	0
0.40	0.06	-0.34	0.10	0.33	0.38	0.00	0.00	0.01	0	0	0
0.64	-0.17	0.36	0.33	-0.16	-0.21	-0.17	0.03	0.27	0	0	0
0.27	0.11	-0.43	0.07	0.08	-0.17	0.28	-0.02	-0.05	0	0	0
0.27	0.11	-0.43	0.07	0.08	-0.17	0.28	-0.02	-0.05	0	0	0
0.30	-0.14	0.33	0.19	0.11	0.27	0.03	-0.02	-0.17	0	0	0
0.21	0.27	-0.18	-0.03	-0.54	0.08	-0.47	-0.04	-0.58	0	0	0
0.01	0.49	0.23	0.03	0.59	-0.39	-0.29	0.25	-0.23	0	0	0
0.04	0.62	0.22	0.00	-0.07	0.11	0.16	-0.68	0.23	0	0	0
0.03	0.45	0.14	-0.01	-0.30	0.28	0.34	0.68	0.18	0	0	0

LSI – an example

$A=ULV^T$

L=

3.3 4	0	0	0	0	0	0	0	0
0	2.54	0	0	0	0	0	0	0
0	0	2.35	0	0	0	0	0	0
0	0	0	1.64	0	0	0	0	0
0	0	0	0	1.50	0	0	0	0
0	0	0	0	0	1.31	0	0	0
0	0	0	0	0	0	0.85	0	0
0	0	0	0	0	0	0	0.56	0
0	0	0	0	0	0	0	0	0.36
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

LSI – an example

$A=ULV^T$

$V=$

0.20	-0.06	0.11	-0.95	0.05	-0.08	0.18	-0.01	-0.06
0.61	0.17	-0.50	-0.03	-0.21	-0.26	-0.43	0.05	0.24
0.46	-0.13	0.21	0.04	0.38	0.72	-0.24	0.01	0.02
0.54	-0.23	0.57	0.27	-0.21	-0.37	0.26	-0.02	-0.08
0.28	0.11	-0.51	0.15	0.33	0.03	0.67	-0.06	-0.26
0.00	0.19	0.10	0.02	0.39	-0.30	-0.34	0.45	-0.62
0.01	0.44	0.19	0.02	0.35	-0.21	-0.15	-0.76	0.02
0.02	0.62	0.25	0.01	0.15	0.00	0.25	0.45	0.52
0.08	0.53	0.08	-0.03	-0.60	0.36	0.04	-0.07	-0.45

LSI – an example

Choosing the 2 largest singular values we have

U_k=

0.22	-0.11
0.20	-0.07
0.24	0.04
0.40	0.06
0.64	-0.17
0.27	0.11
0.27	0.11
0.30	-0.14
0.21	0.27
0.01	0.49
0.04	0.62
0.03	0.45

L_k=

3.34	0
0	2.54

V_k^T=

0.20	0.6 1	0.46	0.54	0.28	0.00	0.02	0.02	0.08
- 0.06	0.1 7	-0.13	-0.23	0.11	0.19	0.44	0.62	0.53

LSI (2 singular values)

$A_k =$

	C1	C2	C3	C4	C5	M1	M2	M3	M4
human	0.16	0.40	0.38	0.47	0.18	-0.05	-0.12	-0.16	-0.09
Interface	0.14	0.37	0.33	0.40	0.16	-0.03	-0.07	-0.10	-0.04
Computer	0.15	0.51	0.36	0.41	0.24	0.02	0.06	0.09	0.12
User	0.26	0.84	0.61	0.70	0.39	0.03	0.08	0.12	0.19
System	0.45	1.23	1.05	1.27	0.56	-0.07	-0.15	-0.21	-0.05
Response	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
Time	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
EPS	0.22	0.55	0.51	0.63	0.24	-0.07	-0.14	-0.20	-0.11
Survey	0.10	0.53	0.23	0.21	0.27	0.14	0.31	0.44	0.42
Trees	-0.06	0.23	-0.14	-0.27	0.14	0.24	0.55	0.77	0.66
Graph	-0.06	0.34	-0.15	-0.30	0.20	0.31	0.69	0.98	0.85
Minors	-0.04	0.25	-0.10	-0.21	0.15	0.22	0.50	0.71	0.62

LSI Example

- Query: "human computer interaction" retrieves documents: c_1,c_2, c_4 but *not* c_3 and c_5.
- If we submit the same query (based on the transformation shown before) to the transformed matrix we retrieve (using cosine similarity) all c_1-c_5 even if c_3 and c_5 have no common keyword to the query.
- According to the transformation for the queries we have:

Query transformation

	query
human	1
Interface	0
computer	1
User	0
System	0
Response	0
Time	0
EPS	0
Survey	0
Trees	0
Graph	0
Minors	0

q=

1
0
1
0
0
0
0
0
0
0
0
0

Query transformation

$q^T=$

1	0	1	0	0	0	0	0	0	0	0	0

$U_k=$

0.22	-0.11
0.20	-0.07
0.24	0.04
0.40	0.06
0.64	-0.17
0.27	0.11
0.27	0.11
0.30	-0.14
0.21	0.27
0.01	0.49
0.04	0.62
0.03	0.45

$L_k=$

0.3	0
0	0.39

$q_n=q^TU_kL_k=$

0.138	-0.0273

Query transformation

Map docs to the 2 dim space $V_k L_k$=

0.20	-0.06
0.61	0.17
0.46	-0.13
0.54	-0.23
0.28	0.11
0.00	0.19
0.01	0.44
0.02	0.62
0.08	0.53

3.34	0
0	2.54

=

0.67	-0.15
2.04	0.43
1.54	-0.33
1.80	-0.58
0.94	0.28
0.00	0.48
0.03	1.12
0.07	1.57
0.27	1.35

$q_n L_k$ =

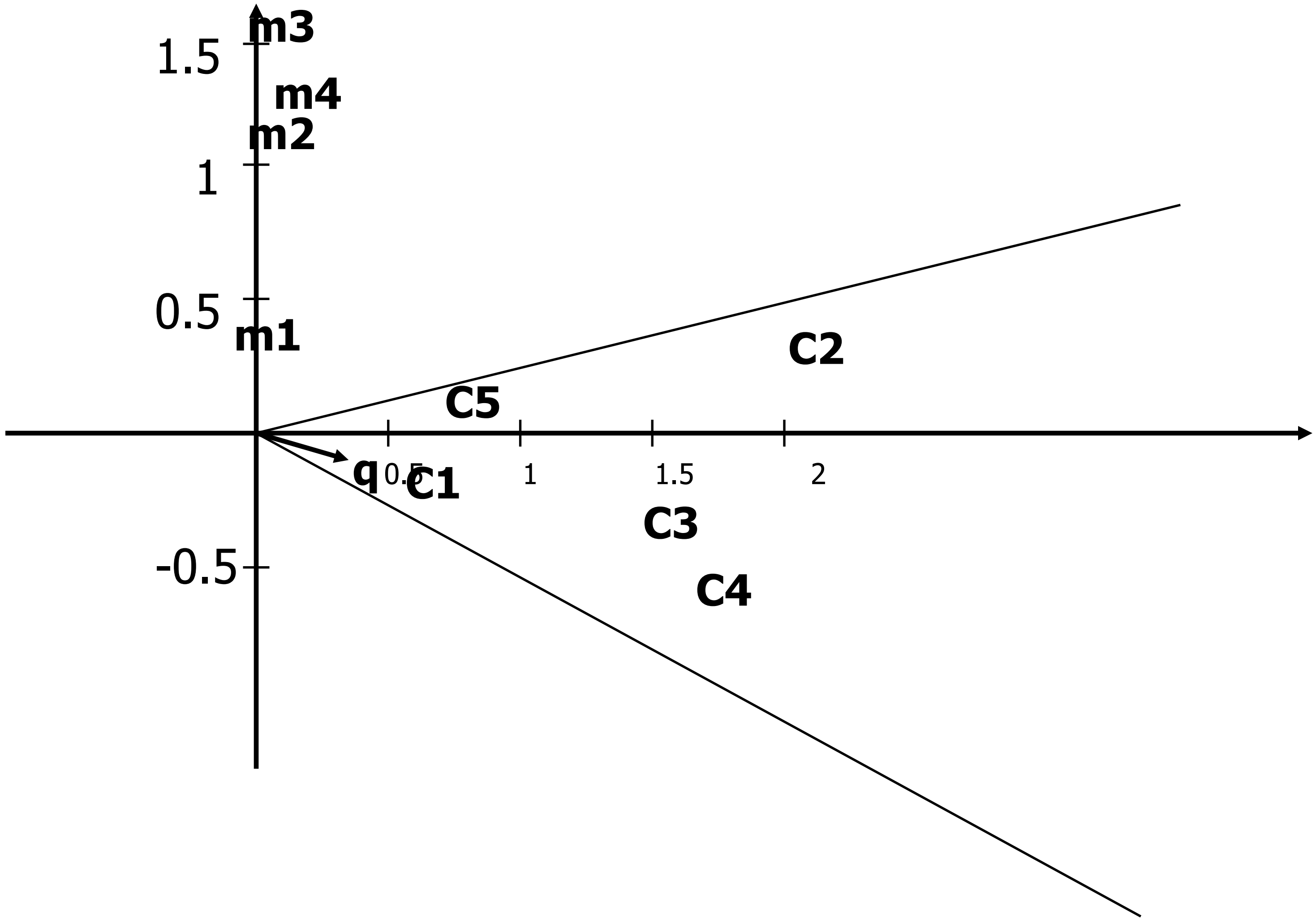
0.138	-0.0273

3.34	0
0	2.54

=

0.46	-0.069

Query transformation



Query transformation

- Comparison of the transformed query to the new document vectors based on cosine similarity, where the similarity is computed as:

$$Cos(x,y)=<x,y>/||x||.||y||$$

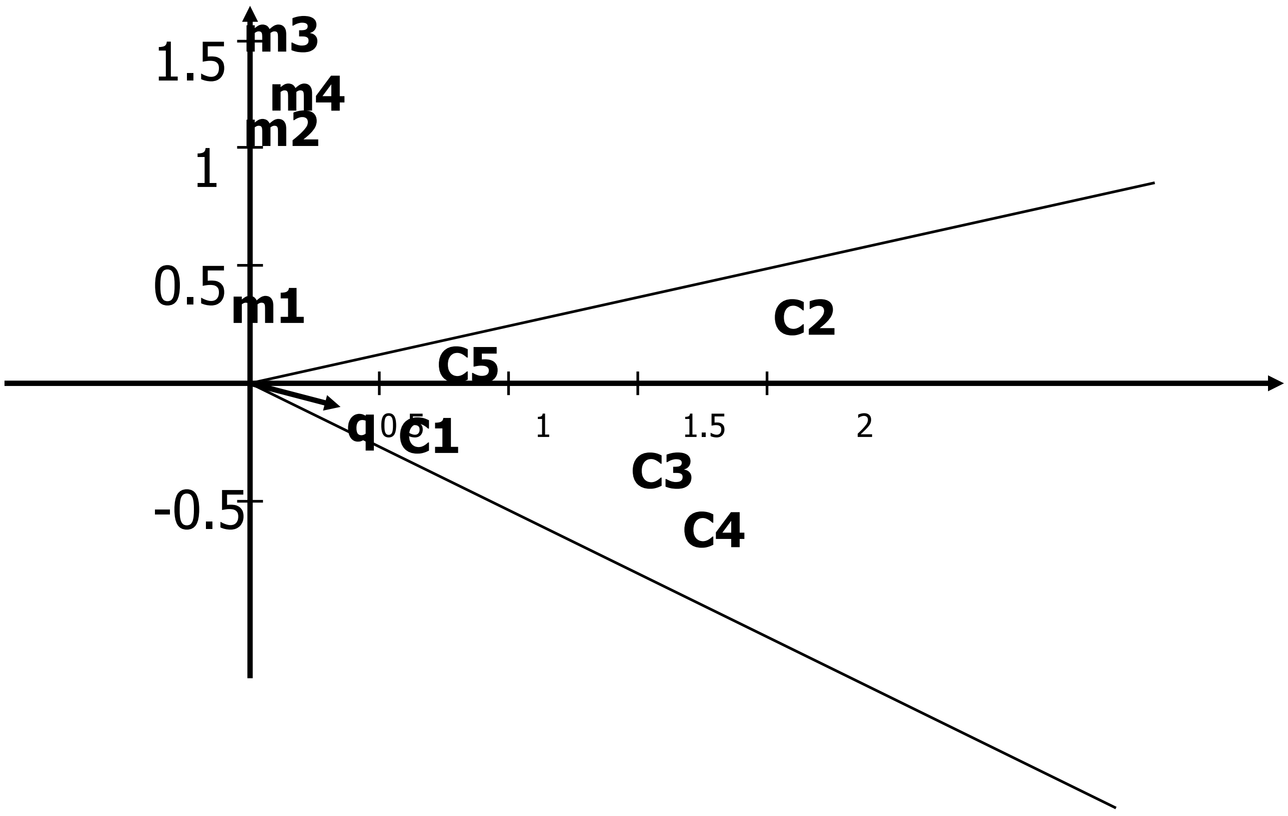
Where $x=(x_1,\ldots,x_n)$, $y=(y_1,\ldots,y_n)$

$$<x,y>=x_1*y_1+\ldots+x_n*y_n$$

Query transformation

- The cosine similarity matrix of query vector to the documents is:

	query
C1	0.99
C2	0.94
C3	0.99
C4	0.99
C5	0.90
M1	-0.14
M2	-0.13
M3	-0.11
M4	0.05



Topic Modeling

- Flux of information: Wikipedia articles, blogs, Flickr images, astronomical survey data, social networking activity
- Need algorithms to organize, search, and understand this information.

Topic modeling

- aims at discovering the theme(s) of documents
- is a method for analyzing large quantities of unlabeled data.

Topic is a probability distribution over a collection of words

Topic model

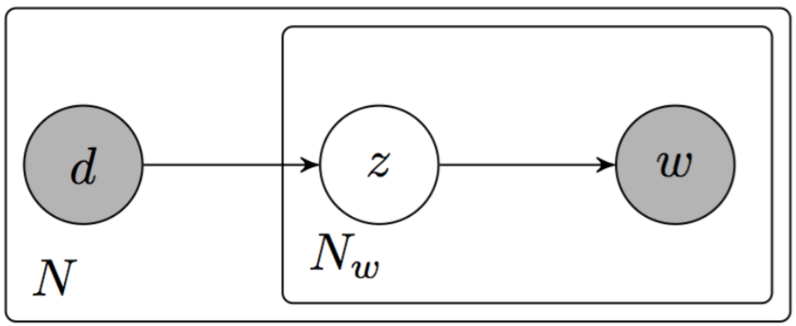
- statistical relationship between a group of observed and latent (unknown) random variables
- specifies a probabilistic procedure to generate the topics—a generative model.
- provides a “thematic summary” of a collection of documents.
- answers the question themes documents discuss – i.e. collection of news articles could discuss e.g. political, sports, and business related themes.

Probabilistic LSA

- Probabilistic Latent Semantic Analysis (pLSA) is topic model method
- main goal: model co- occurrence information under a probabilistic framework to discover the underlying semantic structure of the data.
- Developed Th. Hofmann,1999 = initially used for text-based applications (indexing, retrieval, clustering);
- spread in other fields: such as computer vision or audio processing.
- Goal of pLSA: use co-occurrence matrix to extract the “topics” and explain the documents as a mixture of them.

PLSA

Documents: $d \in D = \{d_1, \cdots, d_N\}$— observed variables, $|D|= N$
Words: $w \in W = \{w_1, \cdots, w_M\}$— observed variables, $|W|=M$
Topics: $z \in Z = \{z_1, \cdots, z_K\}$—latent (or hidden) variables.
$|Z|=K$, has to be specified a priori.



- graphical model representation.
- generative process for each of the N documents.
- Nw: number of words in document d.
- Each word w has associated a latent topic z from which is generated.
- Shaded circles: observed variables,

PLSA – Generative process

- select a document d with probability $P(d)$.
- for each word $w_i, i \in \{1, \cdots, N\}$ in document d_n:
  Select a topic z_i from a multinomial conditioned as $P(z/d_n)$.
  Select a word w_i from a multinomial conditioned as $P(w/z_i)$.

Assuming
- bag-of-words model the joint distribution of the observed data factorize as a product:

$$P(\mathcal{D}, \mathcal{W}) = \prod_{(d,w)} P(d, w).$$

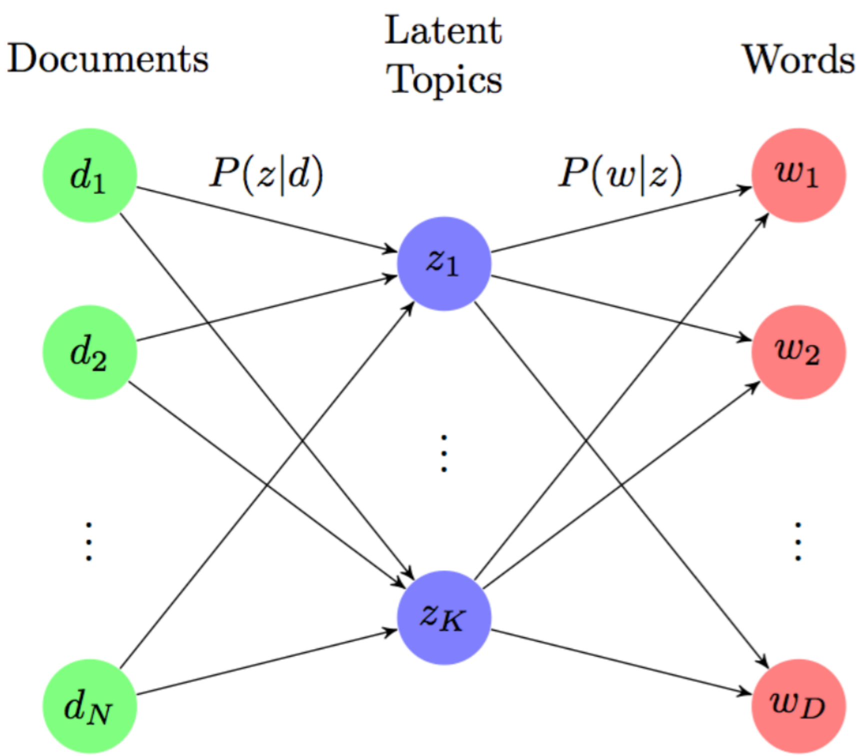
- Conditional independence

$$P(w|d) = \sum_{z \in \mathcal{Z}} P(w|z) P(z|d)$$

$$P(w, d) = \sum_{z \in \mathcal{Z}} P(z) P(d|z) P(w|z).$$

PLSA – mixture model

$$P(w|d) = \sum_{z \in \mathcal{Z}} P(w|z)P(z|d)$$



The general structure of pLSA model.

- intermediate layer of latent topics links documents to words
- each document is a mixture of topics weighted by the probability P(z|d)
- each word expresses a topic with probability P(w|z).

$$L = \prod_{(d,w)} P(w|d) = \prod_{d \in \mathcal{D}} \prod_{w \in \mathcal{W}} P(w|d)^{n(d,w)}$$

n(d, w) frequency of word w in d

PLSA – Log Likelihood Maximization

- Parameters can be estimated with Likelihood Maximization
- Find values maximizing predictive probability for observed word occurrences
- predictive probability of pLSA mixture: *P(w|d)*, so the objective function is:

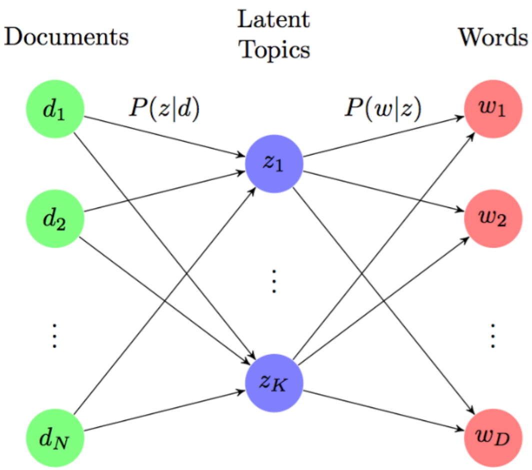
$$L = \prod_{(d,w)} P(w|d) = \prod_{d \in \mathcal{D}} \prod_{w \in \mathcal{W}} P(w|d)^{n(d,w)}$$

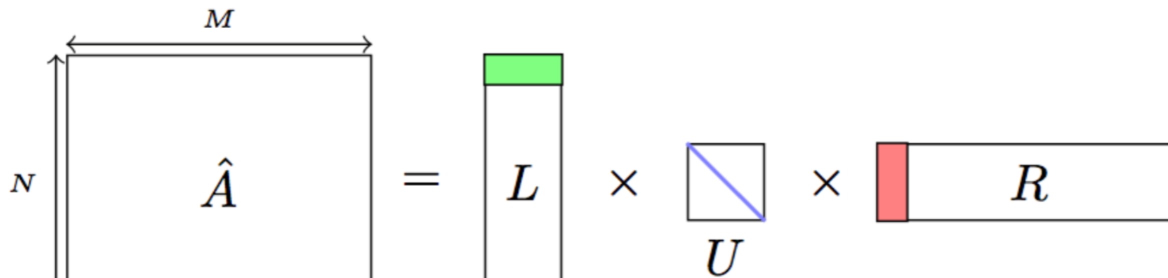
$n(d, w)$ frequency of word w in d

Can be solved with Expectation-Maximization (EM) algorithm for the log-likelihood:

$$\mathcal{L} = \log L = \sum_{d \in \mathcal{D}} \sum_{w \in \mathcal{W}} n(d,w) \cdot \log \sum_{z \in \mathcal{Z}} P(w|z) P(z|d).$$

PLSA – as Matrix Decomposition





A: document-term matrix.
L: document probabilities P (d|z).
U: diagonal matrix - prior probabilities of the topics P (z).
R: word probability P (w|z).

References

- “Latent Dirichlet Allocation: Towards a Deeper Understanding Colorado Reed January 2012
- D. Blei. Introduction to probabilistic topic models. Communications of the ACM, 2011.
- Probabilistic Topic Models, David M. Blei, Department of Computer Science Princeton University, September 2, 2012
- Probabilistic Latent Semantic Analysis, Dan Oneata˘
- Thomas Hofmann. Probabilistic latent se- mantic indexing. In Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR ’99, pages 50–57, New York, NY, USA, 1999. ACM