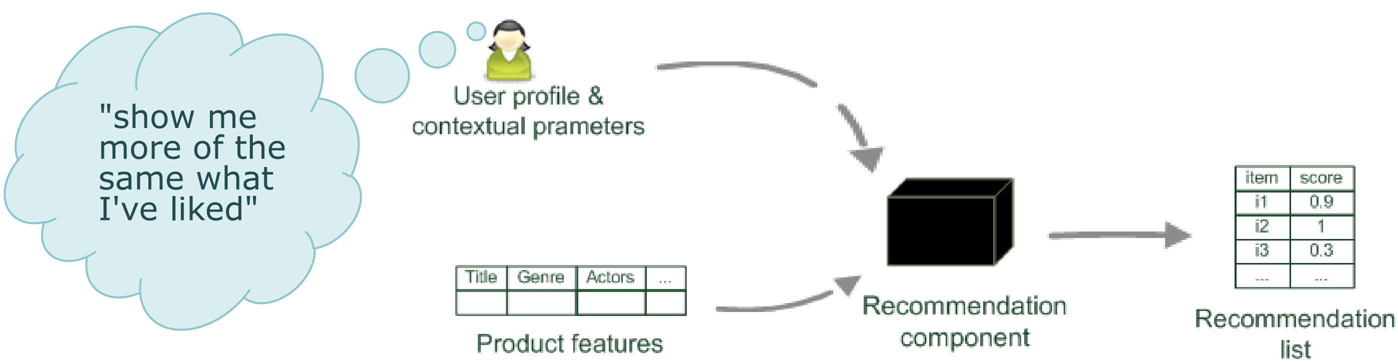
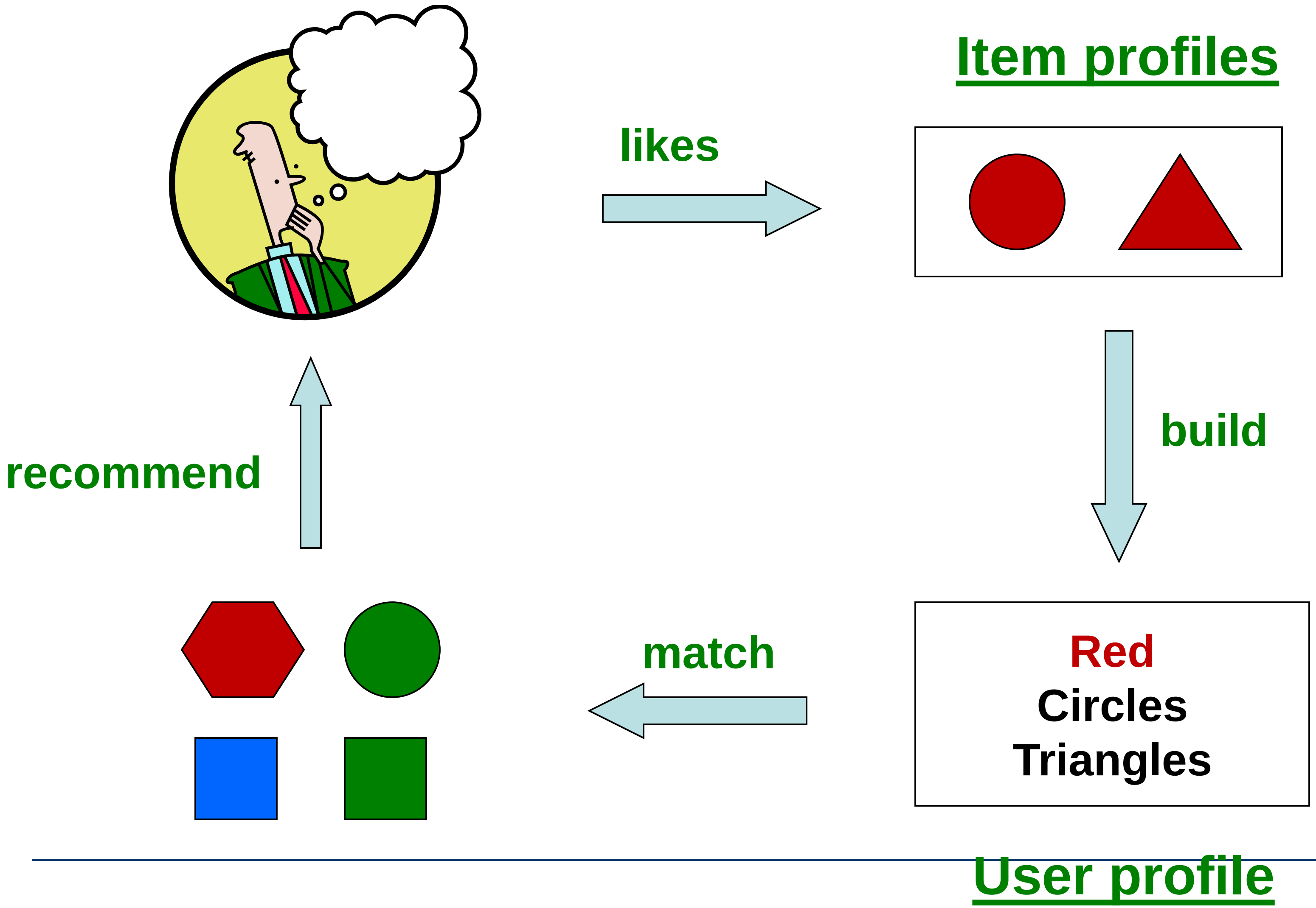
Content-based recommendation

Content-based recommendation

- **While CF – methods do not require any information about the items,**
  - it might be reasonable to exploit such information; and
  - recommend fantasy novels to people who liked fantasy novels in the past
- **What do we need:**
  - some information about the available items such as the genre ("content")
  - some sort of *user profile* describing what the user likes (the preferences)
- **The task:**
  - learn user preferences
  - locate/recommend items that are "similar" to the user preferences



Logic Behind Content-based Recommendation



What is the "content"?

- **Many CB-recommendation techniques were applied to recommending text documents.**
  - Like web pages or newsgroup messages for example. E.g. sports news
- **Content of items can also be represented as text documents.**
  - With textual descriptions of their basic characteristics.
  - Structured: Each item is described by the same set of attributes

Title	Genre	Author	Type	Price	Keywords
The Night of the Gun	Memoir	David Carr	Paperback	29.90	Press and journalism, drug addiction, personal memoirs, New York
The Lace Reader	Fiction, Mystery	Brunonia Barry	Hardcover	49.90	American contemporary fiction, detective, historical
Into the Fire	Romance, Suspense	Suzanne Brockmann	Hardcover	45.90	American fiction, murder, neo-Nazism

  - Unstructured: free-text description.

Content representation and item similarities (by content)

- **User Profile**

Title	Genre	Author	Type	Price	Keywords
The Night of the Gun	Memoir	David Carr	Paperback	29.90	Press and journalism, drug addiction, personal memoirs, New York
The Lace Reader	Fiction, Mystery	Brunonia Barry	Hardcover	49.90	American contemporary fiction, detective, historical
Into the Fire	Romance, Suspense	Suzanne Brockmann	Hardcover	45.90	American fiction, murder, neo-Nazism

- **Item representation (a new item)**

Title	Genre	Author	Type	Price	Keywords
…	Fiction	Brunonia, Barry, Ken Follett	Paperback	25.65	Detective, murder, New York

$keywords(b_j)$ describes Book b_j with a set of keywords

- **Simple approach**
  - Compute the similarity of an unseen item with the user profile based on the keyword overlap (e.g. using the Dice coefficient, i.e., Jaccard similarity)
  - Or use and combine multiple metrics

$$\frac{2 \times |\boldsymbol{keywords}(\boldsymbol{b_i}) \cap \boldsymbol{keywords}(\boldsymbol{b_j})|}{|\boldsymbol{keywords}(\boldsymbol{b_i})| + |\boldsymbol{keywords}(\boldsymbol{b_j})|}$$

Text Processing

- **Tokenization**
  - Cut character sequence into word tokens
    - Deal with ***"John's"***, ***a state-of-the-art solution***
- **Normalization**
  - Map text and query term to same form
    - You want ***U.S.A.*** and ***USA*** to match
- **Stemming**
  - We may wish different forms of a root to match
    - ***authorize***, ***authorization***
- **Stop words**
  - We may omit very common words (or not)
    - ***the, a, to, of***

Term-Frequency - Inverse Document Frequency ($TF - IDF$)

- **Simple keyword representation has its problems**
  - in particular when automatically extracted as
    - not every word has similar importance, e.g., “fiction” vs “the”
    - longer documents have a higher chance to have an overlap with the user profile
- **Standard measure: TF-IDF**
  - Encodes text documents in multi-dimensional Euclidian space
    - weighted term vector
  - TF: Measures, how often a term appears (density in a document)
    - assuming that important terms appear more often
    - normalization has to be done in order to take document length into account
  - IDF: Aims to reduce the weight of terms that appear in all documents
    - E.g. “the”

TF-IDF II

- **Given a keyword i and a document j**
- $TF(i, j)$
  - Term frequency of keyword i in document j
- $IDF(i)$
  - Inverse document frequency calculated as $\boldsymbol{IDF(i) = \frac{1}{n(i)}}$ **or** $\boldsymbol{\log \frac{N}{n(i)}}$
    - N: the total number of recommendable documents
    - $n(i)$: number of documents from N in which keyword i appears
- $TF - IDF$
  - Calculated as: $\boldsymbol{TF\text{-}IDF(i, j) = TF(i, j) * IDF(i)}$
  - Trade of between frequency and uniqueness

Example TF-IDF representation

- **Combined TF-IDF weights**
  - Each document (book) is represented by a real-valued vector of $TF\text{-}IDF$ weights $\in \mathbb{R}^{|v|}$

	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

Example taken from http://informationretrieval.org

Bag of words model

- **Vector representation doesn't consider the ordering of words in a document**
- ***John is quicker than Mary*** **and** ***Mary is quicker than John*** **have the same vectors**
- **This is called the bag of words model.**
- **In a sense, this is a step back: The positional index was able to distinguish these two documents.**

Documents as vectors

- So we have a |V|-dimensional vector space
- Terms are axes of the space
- Documents are points or vectors in this space
- Very high-dimensional: tens of millions of dimensions when you apply this to a web search engine
- These are very sparse vectors - most entries are zero.

Improving the vector space model

- **Vectors are usually long and sparse (due to large vocabulary size)**
- **Remove stop words**
  - They will appear in nearly all documents.
  - E.g. "a", "the", "on", …
- **Use stemming**
  - Aims to replace variants of words by their common stem
  - E.g. "went" ⟹ "go", "stemming" ⟹ "stem", …
  - Past and progressive tense to normal present tense
- **Size cut-offs**
  - Only use top n most representative words to remove "noise" from data
  - E.g. use top 100 words

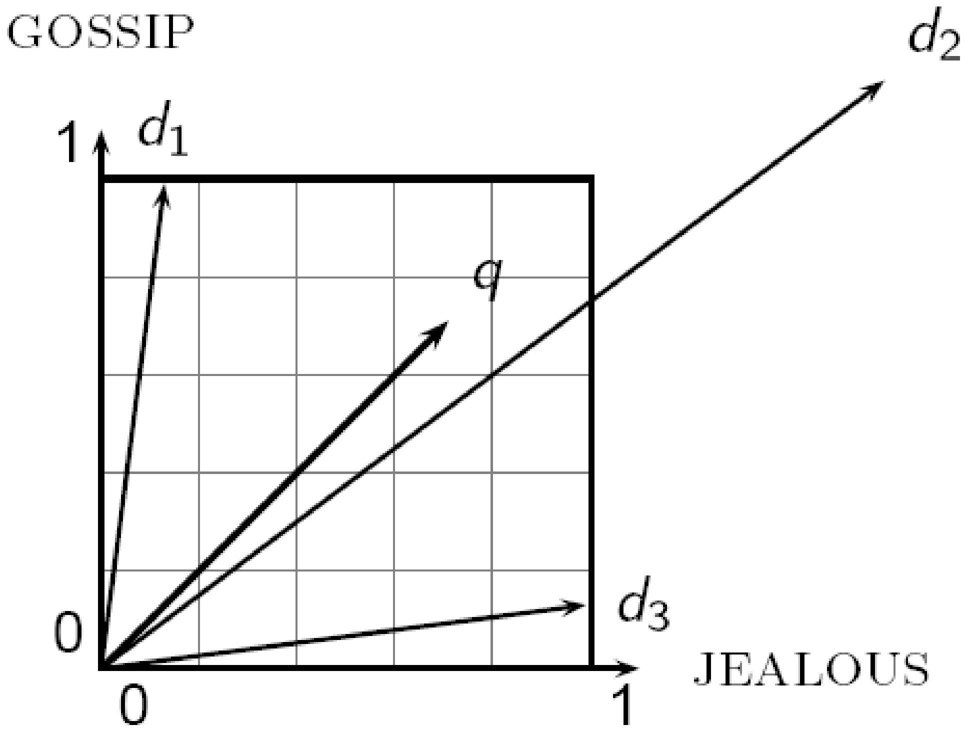
Improving the vector space model II

- **Use lexical knowledge, use more elaborate methods for feature selection**
  - Remove words that are not relevant in the domain
- **Detection of phrases as terms**
  - More descriptive for a text than single words
  - E.g. "United Nations"
- **Still has some limitations**
  - Semantic meaning remains unknown
  - Example: usage of a word in a negative context
    - "there is nothing on the menu that a vegetarian would like.."
    - The word "vegetarian" will receive a higher weight than desired

  ⇨ An unintended match with a user interested in vegetarian restaurants
    - Recent research, NLP for recommendation, deep neural networks (embedding) on text analysis for recommendation

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.



Use angle instead of distance

- **Thought experiment: take a document *d* and append it to itself. Call this document *d′*.**
- **“Semantically” d and d′ have the same content**
- **The Euclidean distance between the two documents can be quite large**
- **The angle between the two documents is 0, corresponding to maximal similarity.**
- **Key idea: Rank documents according to angle with query.**

Length Normalization

- **A vector can be (length-) normalized by dividing each of its components by its length – for this we use the L_2 norm:**

$$\left\|\vec{x}\right\|_2 = \sqrt{\sum_i x_i^2}$$

- **Dividing a vector by its L_2 norm makes it a unit (length) vector (on surface of unit hypersphere)**

- **Effect on the two documents d and d′ (d appended to itself) from earlier slide: they have identical vectors after length-normalization.**
  - Long and short documents now have comparable weights

Cosine Computation

Dot product

Unit vectors

$$\cos(\vec{q},\vec{d}) = \frac{\vec{q} \bullet \vec{d}}{|\vec{q}||\vec{d}|} = \frac{\vec{q}}{|\vec{q}|} \bullet \frac{\vec{d}}{|\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2}\sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

***q_i* is the weight of term *i* in the user text**
***d_i* is the weight of term *i* in the document**

$\cos(\vec{q},\vec{d})$ is the cosine similarity of $\vec{q}$ and $\vec{d}$ … or, equivalently, the cosine of the angle between $\vec{q}$ and $\vec{d}$.

Recommending items

- **Simple method: nearest neighbors**
  - Given a set of documents D already rated by the user (like/dislike)
    - Either explicitly via user interface
    - Or implicitly by monitoring user's behavior
  - Find the n nearest neighbors in D for an not-yet-seen item i
    - Use similarity measures (like cosine similarity) to capture similarity of two documents
  - Take these neighbors to predict a rating for i
    - e.g. $n = 5$ most similar items to i.
      4 of n items were liked by current user $\Rightarrow$ item i will also be liked by this user
  - Variations:
    - Varying neighborhood size n
    - Set up an upper similarity threshold to prevent system from recommending items the user already has seen

Okapi BM25[1]

- **Better content similarity measure**
- **BM25 “Best Match 25” (they had a bunch of tries!)**
  - Developed in the context of the Okapi system
  - Started to be increasingly adopted by other teams during the TREC competitions
  - It works well
- **Goal: be sensitive to term frequency and document length while not adding too many parameters**
  - (Robertson and Zaragoza 2009; Spärck Jones et al. 2000)

[1] http://www.staff.city.ac.uk/~sb317/papers/foundations_bm25_review.pdf

"Early" versions of BM25

- **BIM simplification to IDF**

$$c_i^{BM25v2}(tf_i) = \log \frac{N}{df_i} \times \frac{(k_1+1)tf_i}{k_1+tf_i}$$

  - (k_1+1) factor doesn't change ranking, but makes term score 1 when $tf_i = 1$
  - Similar to *tf*-*idf*, but term scores are bounded

Document length normalization

- **Longer documents are likely to have larger tf_i values**

- **Why might documents be longer?**
  - Verbosity: suggests observed tf_i too high
  - Larger scope: suggests observed tf_i may be right

- **A real document collection probably has both effects**
- **… so should apply some kind of partial normalization**

Document length normalization

- **Document length:**

$$dl = \sum_{i \in V} tf_i$$

- ***avdl*: Average document length over collection**

- **Length normalization component**

$$B = \left((1 - b) + b \frac{dl}{avdl} \right), \qquad 0 \le b \le 1$$

  - $b = 1$ full document length normalization
  - $b = 0$ no document length normalization

Okapi BM25

- **Normalize *tf* using document length**

$$tf_i' = \frac{tf_i}{B}$$

$$c_i^{BM25}(tf_i) = \log \frac{N}{df_i} \times \frac{(k_1+1)tf_i'}{k_1 + tf_i'}$$

$$= \log \frac{N}{df_i} \times \frac{(k_1+1)tf_i}{k_1((1-b)+b\frac{dl}{avdl}) + tf_i}$$

- **BM25 ranking function**

$$RSV^{BM25} = \sum_{i \in q} c_i^{BM25}(tf_i);$$

Okapi BM25

$$RSV^{BM25} = \sum_{i \in q} \log \frac{N}{df_i} \cdot \frac{(k_1 + 1)tf_i}{k_1((1 - b) + b \frac{dl}{avdl}) + tf_i}$$

- **k_1 controls term frequency scaling**
  - $k_1 = 0$ is binary model; k_1 large is raw term frequency
- **b controls document length normalization**
  - $b = 0$ is no length normalization; $b = 1$ is relative frequency (fully scale by document length)
- **Typically, k_1 is set around 1.2–2 and b around 0.75**

Why is BM25 better than VSM tf-idf?

- **Suppose your query is [machine learning]**
- **Suppose you have 2 documents with term counts:**
  - doc1: learning 1024; machine 1
  - doc2: learning 16; machine 8

- **tf-idf: $\log_2$ tf * $\log_2$ (N/df)**
  - doc1: 11 * 7 + 1 * 10 = **87**
  - doc2: 5 * 7 + 4 * 10 = **75**
- **BM25: k_1 = 2**
  - doc1: 7 * 3 + 10 * 1 = **31**
  - doc2: 7 * 2.67 + 10 * 2.4 = **42.7**

Some other content-based methods

Probabilistic methods -- Basics

- **For events *A* and *B*:**

$$p(A,B) = p(A \cap B) = p(A \mid B)p(B) = p(B \mid A)p(A)$$

- **Bayes' Rule:**

Prior

$$p(A \mid B) = \frac{p(B \mid A)p(A)}{p(B)} = \frac{p(B \mid A)\boxed{p(A)}}{\sum_{X=A,\bar{A}} p(B \mid X)p(X)}$$

Posterior

Probabilistic methods

- **Recommendation as classical text classification problem**
  - Long history of using probabilistic methods
- **Simple approach:**
  - 2 classes: hot/cold
  - Simple Boolean document representation
  - Calculate probability that document is hot/cold based on Bayes theorem

Doc-ID	recommender	intelligent	learning	school	Label
1	1	1	1	0	1
2	0	0	1	1	0
3	1	1	0	0	1
4	1	0	1	1	1
5	0	0	0	1	0
6	1	1	0	0	?

$$
\begin{aligned}
&P(X|Label = 1)\\
&= P(recommender = 1|Label = 1)\\
&\times P(intelligent = 1|Label = 1)\\
&\times P(learning = 0|Label = 1)\\
&\times P(school = 0|Label = 1)\\
&= {}^{3}/_{3} \times {}^{2}/_{3} \times {}^{1}/_{3} \times {}^{2}/_{3} \approx 0.149
\end{aligned}
$$

Probabilistic methods

Doc-ID	recommender	intelligent	learning	school	Label
1	1	1	1	0	1
2	0	0	1	1	0
3	1	1	0	0	1
4	1	0	1	1	1
5	0	0	0	1	0
6	1	1	0	0	?

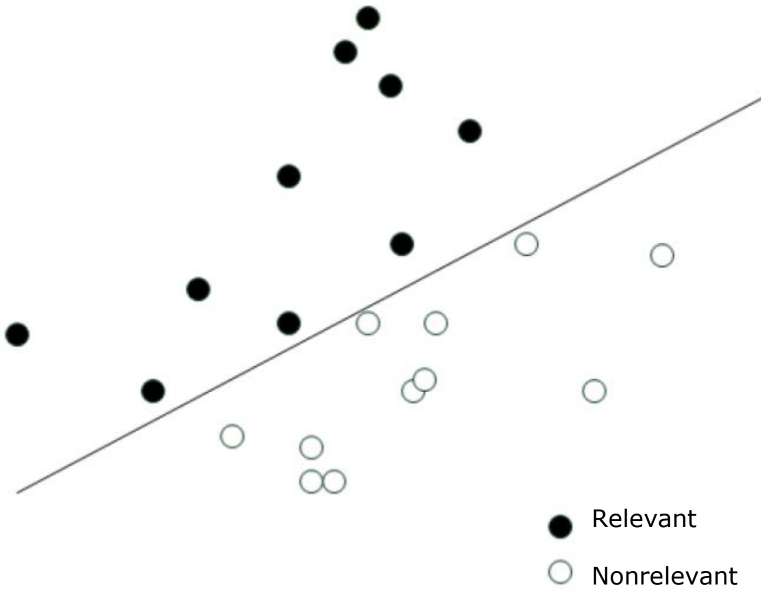
$$\begin{aligned} & P(X \mid Label = 1) \\ = \; & P(recommender = 1 \mid Label = 1) \\ \times \; & P(intelligent = 1 \mid Label = 1) \\ \times \; & P(learning = 0 \mid Label = 1) \\ \times \; & P(school = 0 \mid Label = 1) \\ = \; & {}^{3}/_{3} \times {}^{2}/_{3} \times {}^{1}/_{3} \times {}^{2}/_{3} \approx 0.149 \end{aligned}$$

- **How to compute P(Label|Doc$_6$)?**

$$P(Label = 1 \mid Doc_6) = P(Doc_6 \mid Label = 1) * P(Label = 1)/P(Doc_6)$$

Linear classifiers

- **Most learning methods aim to find coefficients of a linear model**
- **A simplified classifier with only two dimensions can be represented by a line**
- **The line has the form $\boldsymbol{w_1 x_1 + w_2 x_2 = b}$**
  - x_1 and x_2 correspond to the vector representation of a document (using e.g. TF-IDF weights)
  - w_1, w_2 and b are parameters to be learned
  - Classification of a document based on checking
    $$w_1 x_1 + w_2 x_2 > b$$
- **In n-dimensional space the classification function is $\boldsymbol{\vec{w}^T \vec{x} = b}$**
- **Other linear classifiers:**
  - Naive Bayes classifier, Support vector machines



Advantages of content-based recommendation methods

- **No need for data on other users**
  - No cold-start or sparsity problems
- **Able to recommend to users with unique tastes**
- **Able to recommend new & unpopular items**
- **Able to provide explanations**
  - Can provide explanations of recommended items by listing content-features that caused an item to be recommended

Limitations of content-based recommendation methods

- **Keywords alone may not be sufficient to judge quality/relevance of a document or web page**
  - content may also be limited / too short
  - content may not be automatically extractable (multimedia)
  - -> Extracting more informative contents for specific application scenarios
- **Overspecialization**
  - Algorithms tend to propose "more of the same"
  - Or: too similar news items
  - -> Occasionally attempt to recommend “unsimilar” items to users to test if the user’s preference has shifted -> Increase novelty and serendipity, help to track user preferences shift.

Discussion & summary

- In contrast to collaborative approaches, content-based techniques do not require user community in order to work, and can provide very specific recommendations that meet the given requirements
  - Even if there is no similar user to the target user, content-based method can still work

- Presented approaches can learn a model of user's interest preferences based on explicit or implicit feedbacks
  - But accurately interpreting the implicit feedback from user behaviors can be difficult, needs deep understanding of the specific application

- Danger exists that recommendation lists contain too many similar items
  - All learning techniques require a certain amount of training data
  - Some learning methods tend to overfit the training data
- Pure content-based systems are rarely found in commercial environments