Lecture 8: Feedback and Classification

1. Relevance Feedback

2. Document Classification

Relevance Feedback

- Remember the query process from the first lecture:

Result Improvement

- There are three main approaches to result improvement:
  - Manual modification of query (query refinement)
  - Browsing / “Find similar pages”
  - Faceted Search
  - Relevance feedback (RF)

- Manual modification requires active user engagement
- Browsing requires a “good” clustering, which is hard
- Relevance feedback is much easier to use
- Today, we consider two examples of relevance feedback:
  - RF in probabilistic retrieval (BIR)
  - RF in vector space retrieval

Demos

- Faceted search:
  - http://dblp.l3s.de

- Relevance feedback in IR:
  - http://demo.zites.net/search

- Relevance feedback in image search:
  - http://amazon.ece.utexas.edu/~qasim/cires.htm

Implicit Relevance Feedback

- Surf Canyon:
  - http://www.surfcanyon.com

- Other ways to get implicit relevance feedback:
  - Eye tracking
  - Mouse movements
  - Clicks in result list
    - Click on third result but no click on first or second result implies that the first and second result are not relevant.
In the vector space model, relevance feedback is classically done using Rocchio's algorithm (Rocchio, 1971).

**Idea:**
Move the query point…
– …into the direction of relevant documents, and
– …away from nonrelevant documents.

Relevance feedback without asking the user? YES!

**The "manual" part of relevance feedback can be automated.**

**Pseudo Relevance Feedback:**
– Generate a result list for the user’s query
– Assumption: “The top k documents are relevant!”
  + Usually true if k is small
  + Use this assumption for relevance feedback
  + Repeat this several times…

**RF in Probabilistic Retrieval (2)**

- Example:
  Query = “jaguar”
  
<table>
<thead>
<tr>
<th>Relevant Terms</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jaguar car</td>
<td>1/2</td>
</tr>
<tr>
<td>Jaguar system</td>
<td></td>
</tr>
<tr>
<td>Jaguar animal</td>
<td></td>
</tr>
<tr>
<td>Jaguar fast</td>
<td></td>
</tr>
</tbody>
</table>

**RF in Probabilistic Retrieval**

- Remember the BIR retrieval model
  – We had to estimate $Pr(D_i = 1 | D \in R_q)$:
    - How many relevant documents contain term i?
  – We estimated it using heuristics: Choose 0.9!
- Better estimation: Exploit user feedback!
  – Show the user the current retrieval result (with 0.9 estimation)
  – Let him/her label the relevant ones
  – Determine the proportion of relevant documents containing term i by counting
- Use the new estimation to return a better result set
  – This process can be repeated…

**Pros:**
– Works well on average
- **Cons:**
  – Can go horribly wrong for some queries: Topic drift!

**Example of topic drift in pseudo RF:**
Query = “apple”

**Rocchio’s Algorithm**

- **Theory:**
  – The new query should...
    + …maximize cosine similarity to all relevant documents
    + …minimize cosine similarity to all nonrelevant documents
  – Let $C$ be the set of documents returned to the user
  – Let $C_\text{rel}$ be the set of documents rated as relevant
  – Let $C_\text{nonrel}$ be the set of documents rated as nonrelevant
  – Note: $C \cup C_\text{rel} \cup C_\text{nonrel}$ could be true
  – Task: Find the query point $q$ that maximizes
    $$
    \frac{1}{|C_\text{rel}|} \sum_{d \in C_\text{rel}} q \cdot d
    - \frac{1}{|C_\text{nonrel}|} \sum_{d \in C_\text{nonrel}} q \cdot d
    $$

To keep things simple, assume that both the query and all documents are unit vectors
– Vector length does not really matter with cosine similarity

Then the problem becomes:
Maximize \( \langle q, d \rangle \)
subject to \( |q| = 1 \)

This optimization problem can be solved using the method of Lagrange multipliers

Rocchio’s Algorithm (2)

Rocchio’s Algorithm (3)

Maximize (in \( q \))

subject to \( |q| = 1 \)

Observation underlying Lagrange multipliers:
Any maximum of the following expression (in \( q, A \)) yields a maximum of the original expression:

\[ \frac{1}{|C_{+}|} \sum_{d \in C_{+}} d - \frac{1}{|C_{-}|} \sum_{d \in C_{-}} d - \lambda \left( \sum_{i=1}^{m} q_{i}^2 - 1 \right) \]

\([|q| = 1\) is enforced, since otherwise no maximum exists

Rocchio’s Algorithm (4)

How to find the maximum of this expression?
Equate all partial derivatives (wrt. \( q_1, \ldots, q_m, \lambda \)) to zero!
– Partial derivative with respect to \( q \):

\[ \frac{1}{|C_{+}|} \sum_{d \in C_{+}} d - \frac{1}{|C_{-}|} \sum_{d \in C_{-}} d - 2\lambda q_i = 0 \]
– Partial derivative with respect to \( \lambda \):

\[ 1 - \sum_{i=1}^{m} q_i^2 = 0 \]

Rocchio’s Algorithm (5)

The first equation gives:

\[ q_i = \frac{1}{2\lambda} \left( \frac{1}{|C_{+}|} \sum_{d \in C_{+}} d - \frac{1}{|C_{-}|} \sum_{d \in C_{-}} d \right) \]
– Note that all possible choices for \( q \) only differ in their length
– The second equation just expresses the “length 1” constraint
– Therefore, the choice of \( q \) having length 1 is the right one

Rocchio’s Algorithm (6)

We arrive at:

\[ q_{\text{opt}}(\lambda) = \frac{1}{2\lambda} \left( \frac{1}{|C_{+}|} \sum_{d \in C_{+}} d - \frac{1}{|C_{-}|} \sum_{d \in C_{-}} d \right) \]

Because of the constraint \(|q| = 1\), the optimal solution points in the same direction as \( q_{\text{opt}}(\lambda) \) but has unit length:

\[ q_{\text{opt}} = \frac{q_{\text{opt}}(\lambda)}{|q_{\text{opt}}(\lambda)|} \]

Note that \( q_{\text{opt}} \) is a scaled version of the difference vector between \( C_{+} \)'s centroid and \( C_{-} \)'s centroid

Rocchio’s Algorithm (7)

Optimal query

\( x \) non-relevant documents
\( o \) relevant documents

Origin of space
Rocchio’s Algorithm (8)

• Problems:
  – The user’s judgments are biased by the initial result set
  – We cannot trust the user’s judgments ultimately
• Therefore, in practice a modified approach is used
• Idea: Modify the initial query vector!

\[ q_{opt} = \alpha \cdot q_0 + \beta \cdot \sum_{d \in C_r} d - \gamma \cdot \sum_{d \in C_n} d \]

– \( q_0 \): Initial query
– \( \alpha, \beta, \gamma \): Weighting factors

Rocchio’s Algorithm (9)

• How to choose \( \alpha, \beta, \) and \( \gamma \)?
  – Only if we have a lot of judged documents, we want \( \beta \) and \( \gamma \) to be larger than \( \alpha \)
  – Positive feedback usually is more valuable than negative feedback, so set \( \beta > \gamma \)
  – Reasonable values might be:
    – \( \alpha = 1 \)
    – \( \beta = 0.75 \)
    – \( \gamma = 0.15 \)

Rocchio’s Algorithm (10)

\[ q_{opt} = \alpha \cdot q_0 + \beta \cdot \sum_{d \in C_r} d - \gamma \cdot \sum_{d \in C_n} d \]

• Pros:
  – Intuitive approach to automatic query refinement
  – Positive and negative feedback can be exploited
  – Pseudo relevance feedback can enhance result quality without any user interaction

• Cons:
  – Requires the initial query to be “good enough”
  – Relies on the cluster hypothesis:
    • Relevant documents are similar
    • Relevant documents are dissimilar from nonrelevant ones
  – Change of results often is hard to explain to the user

What’s Document Classification?

• Task:
  Automate assigning a given document to one or more categories, based on its contents

• Typical applications in IR:
  – Spam detection
  – E-mail sorting (friends and family, job, study, …)
  – Detection of sexually explicit content
  – Domain-specific search (e.g., Google Scholar)
  – Language detection
  – Information filtering (standing queries)

Lecture 8: Feedback and Classification

1. Relevance Feedback
2. Document Classification

Origin of space
Centroid of relevant documents
Centroid of nonrelevant documents
New query
\[ \alpha = 1 \]
\[ \beta = 1.3 \]
\[ \gamma = 0.5 \]
Supervised Classification

- We will focus on supervised classification here, which is the most common type.

Some fundamental definitions:
- Let $X$ be the document space (e.g., $\mathbb{R}^n$ in vector space retrieval).
- Let $C = \{c_1, \ldots, c_m\}$ be a fixed set of classes (aka categories, labels).
- Let $D$ be a set of training pairs $(d, c) \in X \times C$ (training set).

Task in supervised learning:
- Using a learning algorithm, find a classification function $(aka \ classifier) f : X \rightarrow C$, which maps documents to classes.

Supervised Classification (2)

- The learning algorithm takes the training set $D$ as input and returns the learned classification function $f$.

Training set $D$  Learning algorithm  Classifier $f$

The quality of a learned classification function can be evaluated using a test set, which also consists of correctly labeled training pairs $(d, c) \in X \times C$.
Consequently, the training and test set should be similar (or from the same distribution).

Supervised Classification (3)

Example from (Manning et al., 2008):
- Document contains the word "Beijing".
- Document contains the word "Stuttgart".
- Document contains the word "wall".

$Pr(C) = \frac{\#C}{\#docs}$

$Pr(W) = \frac{\#W}{\#docs}$

$Pr(B) = \frac{\#B}{\#docs}$

$Pr(W|C) = \frac{\#(W \text{ and } C)}{\#C}$

$Pr(W|\neg C) = \frac{\#(W \text{ and } \neg C)}{\#\neg C}$

All these probabilities can be estimated from the training set (possibly using smoothing).

Supervised Classification (4)

- There are several popular learning algorithms, which we will have a look at in this and the next lecture:
  - Naïve Bayes: A simple probabilistic approach.
  - Rocchio: Classes are represented by centroids.
  - K-nearest neighbors: Look at the nearest neighbors of a new document to determine class membership.
  - Support vector machines: Use hyperplanes to cut the document space into slices; each slice corresponds to a class.

Naïve Bayes

A simple Bayesian network:

- $Pr(C = \text{China} | \text{have} \text{Beijing})$
- $Pr(C = \text{China} | \text{have} \text{Stuttgart})$
- $Pr(W = \text{wall} | \text{China})$

All these probabilities can be estimated from the training set (possibly using smoothing).
Classifying a new document:

- We know whether each of the events B, S, and W occurred
- We want to find out whether event C is true

This can be done using Bayes' Theorem:

$$P(A|B) = \frac{P(A) \cdot P(B|A)}{P(B)}$$

Assume that the document to be classified contains the word “Beijing” but neither “Stuttgart” nor “wall”

Consequently, we want to find $P(C | B, \neg S, \neg W)$

Bayes Theorem yields:

$$P(C|B, \neg S, \neg W) = \frac{P(C) \cdot P(B|C) \cdot P(\neg S|C) \cdot P(\neg W|C)}{P(B)}$$

In naïve Bayes (sometimes called idiot Bayes), statistical independence is assumed:

$$P(C|B, \neg S, \neg W) = \frac{P(C) \cdot P(B|C) \cdot P(\neg S|\neg C) \cdot P(\neg W|\neg C)}{P(B)}$$

How to classify a new document d?

- Estimate $P(c|d)$, for any class $c \in C$
- Assign $d$ to the class having the highest probability

Example (from Manning et al., 2008; modified):

- Estimation for $P(China)$: 3/4
- Estimation for $P(Chinese | China)$: 2/3
- Estimation for $P(China | China)$: 1/3
- Estimation for $P(China)$: 1/3
- Estimation for $P(China | China)$: 1/3
- Estimation for $P(China)$: 1/3
- Estimation for $P(China | China)$: 1/3

$$P(China | Chinese, Tokyo, Japan, \neg Shanghai, \neg Beijing) = 3/4 \cdot 2/3 \cdot 1/3 \cdot 2/3 \cdot 1/3 = 64/243 \approx 0.26$$

Since $P(China | \ldots) > P(China | \ldots)$, let’s classify doc 5 as “China”

Well, obviously, we need some smoothing here…

For example, estimate $P(Chinese | \neg China)$ by a linear blend of

$$\frac{P(Chinese)}{P(China)} \cdot 0.8 + \frac{P(China)}{P(China)} \cdot 0.2$$

From now on, we estimate $P(Chinese | China)$ by $0.8 \cdot 0 + 0.2 \cdot 0.25 = 0.15$
Using the smoothed estimates, we get the following:

- \( \Pr(\text{China} \mid \text{Chinese}, \text{Tokyo}, \text{Japan}, \neg \text{Shanghai}, \neg \text{Beijing}) \approx 0.34 \)
- \( \Pr(\neg \text{China} \mid \text{Chinese}, \text{Tokyo}, \text{Japan}, \neg \text{Shanghai}, \neg \text{Beijing}) \approx 0.37 \)

Since \( \Pr(\text{China} \mid \ldots) < \Pr(\neg \text{China} \mid \ldots) \), let’s classify doc 5 as “\neg \text{China}”

Typically, when using naïve Bayes, one considers only positive events, that is, only probabilities of terms that actually occur in the document:

- \( \Pr(\text{China} \mid \text{Chinese}, \text{Tokyo}, \text{Japan}) \approx 0.51 \)
- \( \Pr(\neg \text{China} \mid \text{Chinese}, \text{Tokyo}, \text{Japan}) \approx 0.65 \)

Since \( \Pr(\text{China} \mid \ldots) < \Pr(\neg \text{China} \mid \ldots) \), let’s classify doc 5 as “\neg \text{China}”

There are many ways to extend naïve Bayes:

- Account for number of occurrences
- Use better smoothing techniques for estimations
- Do not assume independence
- Restrict model to the “most indicative” terms
- Extend model to handle more than two classes
- …

Example (from Manning et al., 2008):

Rocchio relies on the contiguity hypothesis:

“Documents in the same class form a contiguous region and regions of different classes do not overlap”

Rocchio classification

- Requires a vector space representation of documents
- Divides the space into regions centered on centroids

“Documents in the same class form a contiguous region and regions of different classes do not overlap”

Example (from Manning et al., 2008):

A training set with 3 classes: China, UK, and Kenya
Rocchio classification:

Compute centroids and assign new documents to their nearest centroid

Centroid of class “UK”

These lines divide the space into contiguous regions (“Voronoi tessellation”)

Unlike Rocchio, k-nearest neighbor classification (kNN) uses class boundaries based on individual documents (instead of centroids of classes)

Each new documents gets assigned to the majority class of its k closest neighbors, where k is a parameter

For k = 1, the classes correspond to the Voronoi tessellation of the training set

Clearly, kNN for k > 1 is more robust than kNN for k = 1

Example (from Manning et al., 2008):

k = 1

We can also weight the “votes” of the k nearest neighbors by their cosine similarity

The score of class c with respect to some document to be classified d then is:

\[ \sum_{d' \in \text{NN}_k(d)} \frac{d \cdot d'}{||d|| \cdot ||d'||} \]

– \( \text{NN}_k(d) \): The set of the k nearest neighbors of d in the training set
– \( \text{class}(d') \): The class of training document d'

Every document to be classified gets assigned to the class having the highest score

• Another very important classifier:
  – Support vector machines
  – Highly effective but more complicated to explain
  – Next week …

• Each different classification algorithm comes with individual strengths and weaknesses
  – “There ain’t no such thing as a free lunch”

• For hard classification problems, the usual classifiers tend to be weak learners
  – Weak learner = only slightly better than random guessing

• Question:
  – Can a set of weak learners create a single strong learner?

• Answer: YES!
  – Boosting algorithms do the trick!
**Boosting (2)**

- Boosting algorithms are meta-algorithms
  - Basically, a boosting algorithm is a blueprint of how to combine a set of "real" classification algorithms to yield a single combined (and hopefully better) classifier

**Boosting (3)**

- Naïve approach to boosting: Majority vote!
  1. Train base classifiers independently on the training set
  2. For each new object to be classified, independently ask each base classifier and return the answer given by the majority

- Problems:
  - Does only work if the majority is right very often
  - Each base algorithm cannot take advantage of its individual strengths
  - Should expert votes have the same weight as any other vote?

**Boosting (4)**

- Better approach: Adaptive boosting
  1. Train the first base classifier on the training set
  2. Check which training examples cannot be explained by the first case classifier’s underlying model ("errors")
  3. Assign a weight to each training example
     - Low weight = Example fits perfectly into the first classifier’s model
     - High weight = Example fits hardly into the first classifier’s model
  4. Train the second base classifier on the weighted training set
     - Fitting training example with high weights is more important than fitting those with low weights
  5. Reweight as in step (3)
  6. Repeat the steps (4) and (5) for all remaining base classifiers

**Boosting (5)**

- Adaptive boosting (continued)
  - In addition, assign an importance weight to each base classifier, depending on how many training examples fit its model
    - High importance if errors occur only on training examples with low weight
    - Low importance if errors occur on training examples with high weight
  - How does the combined classifier work?
    1. Classify the new example with each base classifier
    2. Use majority vote but weight the individual classifier’s answers by their importance weights; also incorporate each classifier’s confidence if this information is available
  - Typically, the importance weights and the weights of the individual training examples are chosen to be balanced, such that the weighted majority now is right very often

**Boosting (6)**

- Why is adaptive boosting better than "pure" majority vote?
  - Later weak learners focus more on those training examples previous weak learners had problems with
  - Individual weaknesses can be compensated
  - Individual strengths can be exploited

**Boosting: Example**

- Toy example:
Boosting: Example

Round 1:

- Model of classifier 1
- Reweighted training data

Round 2:

- Model of classifier 2
- Reweighted training data

Round 3:

- Model of classifier 3

Combined classifier:

- Model of classifier 1
- Model of classifier 2
- Model of classifier 3

Support vector machines

The bias–variance tradeoff (overfitting)