CS54701: Information Retrieval

CS-54701
Information Retrieval

Course Review

Luo Si

Department of Computer Science
Purdue University
Basic Concepts of Information Retrieval:

- Task definition of Ad-hoc IR
  - Terminologies and concepts
  - Overview of retrieval models

- Text representation
  - Indexing
  - Text preprocessing

- Evaluation
  - Evaluation methodology
  - Evaluation metrics
Ad-hoc IR: Terminologies

Terminologies:

- **Query**
  - Representative data of user’s information need: text (default) and other media

- **Document**
  - Data candidate to satisfy user’s information need: text (default) and other media

- **Database|Collection|Corpus**
  - A set of documents

- **Corpora**
  - A set of databases
  - Valuable corpora from TREC (Text Retrieval Evaluation Conference)
AD-hoc IR: Basic Process

1. Information Need
2. Representation
   - Query
   - Retrieval Model
   - Retrieved Objects
   - Evaluation/Feedback
3. Representation
   - Indexed Objects
Zipf’s law: relate a term’s frequency to its rank

- Rank all terms with their frequencies in descending order, for a term at a specific rank (e.g., \( r \)) collects and calculates:

\[
\begin{align*}
    f_r & : \text{term frequency} \quad p_r = \frac{f_r}{N} : \text{relative term frequency} \\
    & \text{Relative term frequency} \\
    & \text{Total number of words}
\end{align*}
\]

- Zipf’s law (by observation):

\[
p_r = \frac{A}{r} \quad A \approx 0.1
\]

So \[ p_r = \frac{f_r}{N} = \frac{A}{r} \Rightarrow rf_r = AN \Rightarrow \log(r) = -\log(f_r) + \log(AN) \]

So Rank \times Frequency = Constant
Application of Zipf’s law

- In a 1,000,000 word corpus, rank of a term that occur 100 times?
  \[ r \times 100 = 0.1 \times N \Rightarrow r = 1000 \]

- In a 1,000,000 word corpus, estimate the number of terms that occur 100 times?
  - Assume rank \( r_n \) associates to the last word that occur \( n \) times
    \[ r_n = \frac{AN}{n} \quad \text{and} \quad r_{n+1} = \frac{AN}{n+1} \]

  So: the number is about
  \[ r_n - r_{n+1} = \frac{AN}{n(n+1)} = 0.1 \times 1,000,000/(100 \times 101) = 10 \]
Text Preprocessing: extract representative index terms

- Parse query/document for useful structure
  - E.g., title, anchor text, link, tag in XML...

- Tokenization
  - For most western languages, words separated by spaces; deal with punctuation, capitalization, hyphenation
  - For Chinese, Japanese: more complex word segmentation...

- Remove stopwords: (remove “the”, “is”,..., existing standard list)

- Morphological analysis (e.g., stemming):
  - Stemming: determine stem form of given inflected forms

- Other: extract phrases; decompounding for some European languages
Evaluation

Evaluation criteria

- Effectiveness
  - Favor returned document ranked lists with more relevant documents at the top
  - Objective measures
    - Recall and Precision
    - Mean-average precision
    - Rank based precision

For documents in a subset of a ranked lists, if we know the truth

<table>
<thead>
<tr>
<th></th>
<th>Retrieved</th>
<th>Not retrieved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relevant</td>
<td>Relevant docs retrieved</td>
<td>Relevant docs not retrieved</td>
</tr>
<tr>
<td>Irrelevant</td>
<td>Irrelevant docs retrieved</td>
<td>Irrelevant docs not retrieved</td>
</tr>
</tbody>
</table>

\[
\text{Precision} = \frac{\text{Relevant docs retrieved}}{\text{Retrieved docs}}
\]

\[
\text{Recall} = \frac{\text{Relevant docs retrieved}}{\text{Relevant docs}}
\]
Evaluation

Pooling Strategy

- Retrieve documents using multiple methods
- Judge top n documents from each method
- Whole retrieved set is the union of top retrieved documents from all methods
- Problems: the judged relevant documents may not be complete
- It is possible to estimate size of true relevant documents by randomly sampling
Evaluation

Single value metrics

- Mean average precision
  - Calculate precision at each relevant document; average over all precision values

- 11-point interpolated average precision
  - Calculate precision at standard recall points (e.g., 10%, 20%...); smooth the values; estimate 0 % by interpolation
  - Average the results

- Rank based precision
  - Calculate precision at top ranked documents (e.g., 5, 10, 15…)
  - Desirable when users care more for top ranked documents
Retrieval Models: Outline

Retrieval Models

- Exact-match retrieval method
  - Unranked Boolean retrieval method
  - Ranked Boolean retrieval method

- Best-match retrieval method
  - Vector space retrieval method
  - Latent semantic indexing
Unranked Boolean: Exact match method

Selection Model
- Retrieve a document iff it matches the precise query
- Often return unranked documents (or with chronological order)

Operators
- Logical Operators: AND OR, NOT
- Approximately operators: #1(white house) (i.e., within one word distance, phrase) #sen(Iraq weapon) (i.e., within a sentence)
- String matching operators: Wildcard (e.g., ind* for india and indonesia)
- Field operators: title(information and retrieval)…
Retrieval Models: Unranked Boolean

Advantages:
- Work well if user knows exactly what to retrieve
- Predicable; easy to explain
- Very efficient

Disadvantages:
- It is difficult to design the query; high recall and low precision for loose query; low recall and high precision for strict query
- Results are unordered; hard to find useful ones
- Users may be too optimistic for strict queries. A few very relevant but a lot more are missing
Retrieval Models: Ranked Boolean

Ranked Boolean: Exact match

- Similar as unranked Boolean but documents are ordered by some criterion

Retrieve docs from Wall Street Journal Collection

Query: (Thailand AND stock AND market)

Which word is more important?

Many “stock” and “market”, but fewer “Thailand”. Fewer may be more indicative

Term Frequency (TF): Number of occurrence in query/doc; larger number means more important

Inversed Document Frequency (IDF):
Larger means more important

Total number of docs

Number of docs contain a term

There are many variants of TF, IDF: e.g., consider document length
Retrieval Models: Ranked Boolean

Ranked Boolean: Calculate doc score

- Term evidence: Evidence from term i occurred in doc j: $(tf_{ij})$ and $(tf_{ij} \cdot idf_i)$
- AND weight: minimum of argument weights
- OR weight: maximum of argument weights

Query: (Thailand AND stock AND market)
Retrieval Models: Ranked Boolean

Advantages:

- All advantages from unranked Boolean algorithm
  - Works well when query is precise; predictive; efficient
- Results in a ranked list (not a full list); easier to browse and find the most relevant ones than Boolean
- Rank criterion is flexible: e.g., different variants of term evidence

Disadvantages:

- Still an exact match (document selection) model: inverse correlation for recall and precision of strict and loose queries
- Predictability makes user overestimate retrieval quality
Vector space model

- Any text object can be represented by a term vector
  - Documents, queries, passages, sentences
  - A query can be seen as a short document

- Similarity is determined by distance in the vector space
  - Example: cosine of the angle between two vectors

- The SMART system
  - Developed at Cornell University: 1960-1999
  - Still quite popular
Retrieval Models: Vector Space Model

Vector representation
Retrieval Models: Vector Space Model

Give two vectors of query and document

- query as \( \vec{q} = (q_1, q_2, ..., q_n) \)
- document as \( \vec{d}_j = (d_{j1}, d_{j2}, ..., d_{jn}) \)
- calculate the similarity

Cosine similarity: Angle between vectors

\[
sim(q, \vec{d}_j) = \cos(\theta(q, \vec{d}_j))
\]

\[
\cos(\theta(q, \vec{d}_j)) = \frac{\vec{q} \cdot \vec{d}_j}{\|q\| \|d\|} = \frac{q_1d_{j1} + q_2d_{j2} + ... + q_jd_{jn}}{\sqrt{q_1^2 + ... + q_n^2} \sqrt{d_{j1}^2 + ... + d_{jn}^2}}
\]
Retrieval Models: Vector Space Model

Common vector weight components:

- Inc.ltc: widely used term weight
  - “l”: log(tf)+1
  - “n”: no weight/normalization
  - “t”: log(N/df)
  - “c”: cosine normalization

\[
\frac{q_1 d_{j_1} + q_2 d_{j_2} \ldots + q_n d_{j_n}}{\|q\|\|d_j\|} = \frac{\sum_k \left[ (\log(tf_q(k) + 1) \log(tf_j(k) + 1) \log \frac{N}{df(k)} \right]}{\sqrt{\sum_k [(\log(tf_q(k) + 1)^2] \sqrt{\sum_k \left[ (\log(tf_j(k) + 1) \log \frac{N}{df(k)} \right]^2}}}
\]
Retrieval Models: Vector Space Model

Advantages:

- Best match method; it does not need a precise query
- Generated ranked lists; easy to explore the results
- Simplicity: easy to implement
- Effectiveness: often works well
- Flexibility: can utilize different types of term weighting methods
- Used in a wide range of IR tasks: retrieval, classification, summarization, content-based filtering...
Disadvantages:

- Hard to choose the dimension of the vector ("basic concept"); terms may not be the best choice
- Assume independent relationship among terms
- Heuristic for choosing vector operations
  - Choose of term weights
  - Choose of similarity function
- Assume a query and a document can be treated in the same way
Latent Semantic Indexing (LSI): Explore correlation between terms and documents

- Two terms are correlated (may share similar semantic concepts) if they often co-occur
- Two documents are correlated (share similar topics) if they have many common words

Latent Semantic Indexing (LSI): Associate each term and document with a small number of semantic concepts/topics
Retrieval Models: Latent Semantic Indexing

Using singular value decomposition (SVD) to find the small set of concepts/topics

\( m \): number of concepts/topics

\[
X = USV^T
\]

\( U^TU = I_m \)

\( V^TV = I_m \)

Representation of document in concept space

Representation of term in concept space
Retrieval Models: Latent Semantic Indexing

Retrieval with respect to a query

- Map (fold-in) a query into the representation of the concept space
  \[ \vec{q}' = \vec{q}^T U_k Inv(S_k) \]

- Use the new representation of the query to calculate the similarity between query and all documents
  - Cosine Similarity
Query Expansion: Outline

Query Expansion via Relevant Feedback
- Relevance Feedback
- Blind/Pseudo Relevance Feedback

Query Expansion via External Resources
- Thesaurus
  - “Industrial Chemical Thesaurus”, “Medical Subject Headings” (MeSH)
- Semantic network
  - WordNet
**Goal:** Move new query close to relevant documents and far away from irrelevant documents

**Approach:** New query is a weighted average of original query, and relevant and non-relevant document vectors

\[
\tilde{q}' = \tilde{q} + \alpha \frac{1}{|R|} \sum_{\tilde{d}_i \in R} \tilde{d}_i - \beta \frac{1}{|NR|} \sum_{\tilde{d}_i \in NR} \tilde{d}_i \quad \text{(Rocchio formula)}
\]

How to set the desired weights?
Probability and Statistics: Outline

- **Probability**
  - Basic concepts of probability
  - Conditional probability and Independence
  - Common probability distributions
  - Bayes’ Rule

- **Statistics Inference**
  - Statistical learning
  - Maximum likelihood estimation (MLE)
  - Maximum a posterior (MAP) estimation

- **Introduction to optimization**
Independence

- Two events A and B are independent iff
  \[ \Pr(A, B) = \Pr(A)\Pr(B) \]
  - The probability of both A and B happens is: probability of A happens times probability of B happens
  - Two events do not have influence on each other

- Example:

<table>
<thead>
<tr>
<th>Department</th>
<th>Male</th>
<th>Female</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Admitted</td>
<td>Not admitted</td>
</tr>
<tr>
<td>Dept1</td>
<td>40</td>
<td>360</td>
</tr>
<tr>
<td>Dept2</td>
<td>20</td>
<td>80</td>
</tr>
</tbody>
</table>

- Is admission independent from gender?
  \[ \Pr(\text{admitted}, \text{male}) = \frac{60}{800} = 7.5\% \]
  \[ \Pr(\text{admitted}) \cdot \Pr(\text{male}) = \frac{110}{800} \cdot \frac{500}{800} = 8.5\% \]
  - Not independent
Conditional Independence

- Events A and B are conditionally independent given C iff $\Pr(A,B|C) = \Pr(A|C)\Pr(B|C)$
  - If we know the outcome of event C, then outcomes of event A and B are independent

- Example

<table>
<thead>
<tr>
<th>Department</th>
<th>Male</th>
<th>Female</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Admitted</td>
<td>Not admitted</td>
</tr>
<tr>
<td>Dept1</td>
<td>40</td>
<td>360</td>
</tr>
<tr>
<td>Dept2</td>
<td>20</td>
<td>80</td>
</tr>
</tbody>
</table>

Pr(Male, Admitted | Dept1) = 40/500 = 8%
Pr(Admitted|Dept1)*Pr(male|Dept1) = 50/500*400/500 = 8%
Common Probability Distribution
Multinomial

- Model multiple outcomes: side of a dice; topic of documents; occurrences of terms appear within a document;

  - Multinomial: $n$ outcomes of a variable with multiple values ($v_1..v_n$), the probability $p_1$ to be $v_1$, …, the probability $p_k$ to be $v_k$, what is probability of $v_1$ appear $x_1$ times, … $v_k$ appear $x_k$ times

  \[
P(X_1 = x_1, \ldots, X_K = x_K \mid n, p_1, \ldots, p_k) = \frac{n!}{x_1! \ldots x_K!} p_1^{x_1} \ldots p_K^{x_K}; \sum_{i=1}^{K} p_k = 1; 0 \leq p_k \leq 1
\]
Examples:

Three words in vocabulary (sport, basketball, finance), a multinomial model generates the words by probabilities as \((p_s=0.5, p_b=0.4, p_f=0.1)\) (represented by the first character of each word)

A document generated by this model contains 10 words.

Question:

- What is the expectation of occurrences of word “sport”? 
  \(10 \times 0.5 = 5\)

- What is the probability of generating 5 “sport”, 3 “basketball” and 2 “finance”?
  \[
  \frac{10!}{5!3!2!} \times 0.5^5 \times 0.4^3 \times 0.1^2
  \]

Does the word order matter here? Bag of words representation…
Bayes’s Rule

- Interpretation of Bayes’ Rule

  Hypothesis space: \( H = \{H_1, \ldots, H_n\} \)
  
  Observed Data: \( D \)

  \[
P(H_i \mid D) = \frac{P(D \mid H_i)P(H_i)}{P(D)}
  \]

  To pick the most likely hypothesis \( H^* \), \( p(D) \) can be dropped

  \[
P(H_i \mid D) \propto P(D \mid H_i)P(H_i)
  \]

  Posterior probability of \( H_i \)

  Prior probability of \( H_i \)

  Likelihood of data if \( H_i \) is true
Maximum Likelihood Estimation (MLE)

Maximum Likelihood Estimation:

- Find model parameters that make generation likelihood reach maximum:
  \[ M^* = \text{argmax}_M \Pr(D|M) \]

There are K words in vocabulary, \( w_1 \ldots w_K \) (e.g., 5)

Data: documents \( \vec{d}_1, \ldots, \vec{d}_I \)

For \( \vec{d}_i \) with counts \( c_i(w_1), \ldots, c_i(w_K) \), and length \( |\vec{d}_i| \)

Model: multinomial M with parameters \( \{p(w_k)\} \)

Likelihood: \( \Pr(\vec{d}_1, \ldots, \vec{d}_I|M) \)

\[ M^* = \text{argmax}_M \Pr(\vec{d}_1, \ldots, \vec{d}_I|M) \]
Maximum Likelihood Estimation (MLE)

\[
p(d_1,..,d_L | M) = \prod_{i=1}^{L} \left\{ \left( \frac{d_i}{c_i(w_1)\ldots c_i(w_K)} \right) \prod_{k=1}^{K} p_k^{c_i(w_k)} \right\} \propto \prod_{i=1}^{L} \prod_{k} p_k^{c_i(w_k)}
\]

\[
l(d_1,..,d_L | M) = \log p(d_1,..,d_L | M) = \sum_{i=1}^{L} \sum_{k} c_i(w_k) \log p_k
\]

\[
l'(d_1,..,d_L | M) = \sum_{i=1}^{L} \sum_{k} c_i(w_k) \log p_k + \lambda(\sum_{k} p_k - 1)
\]

\[
\frac{\partial l'}{\partial p_k} = \sum_{i=1}^{L} c_i(w_k) + \lambda = 0 \quad \Rightarrow \quad p_k = -\frac{\sum_{i=1}^{L} c_i(w_k)}{\lambda}
\]

Since \( \sum_{k} p_k = 1, \lambda = -\sum_{k} \sum_{i=1}^{L} c_i(w_k) = -\sum_{i=1}^{L} |d_i| \) \quad \text{So,} \quad p_k = p(w_k) = \frac{\sum_{i=1}^{L} c_i(w_k)}{\sum_{i=1}^{L} |d_i|}

Use Lagrange multiplier approach
Set partial derivatives to zero
Get maximum likelihood estimate
Maximum Likelihood Estimation (MLE)

Example:
- Given a document topic model, which is a multinomial distribution

Five words in vocabulary (sport, basketball, ticket, finance, stock)

Observe two documents

\( \vec{d}_1 : (\text{sport basketball ticket}) \)
\( \vec{d}_2 : (\text{sport basketball sport}) \)

Maximum likelihood parameters of multinomial distribution

\( (p_{sp}, p_{b}, p_{t}, p_{f}, p_{st}) = (3/6, 2/6, 1/6, 0/6, 0/6, 0/6) \)

so \( (p_{sp} = 0.5, p_{b} = 0.33, p_{t} = 0.17, p_{f} = 0, p_{st} = 0) \)
Maximum Likelihood Estimation:
- Zero probabilities with small sample (e.g., 0 for finance)
- Purely data driven, cannot incorporate prior belief/knowledge

Maximum A Posterior Estimation:
- Select a model that maximizes the probability of model given observed data

\[ M^* = \arg\max_M \Pr(M|D) = \arg\max_M \Pr(D|M)\Pr(M) \]

- \( \Pr(M) \): Prior belief/knowledge
- Use prior \( \Pr(M) \) to avoid zero probabilities
Maximum A Posterior (MAP) Estimation

- Dirichlet Prior is the conjugate prior for multinomial distribution
- For the topic model estimation example, MAP estimator is:
  \[
  p_k = \frac{\sum_{i=1}^{l} c_i (w_k) + (\alpha_k - 1)}{\sum_{i=1}^{l} |d_i| + \sum_{k}(\alpha_k - 1)}
  \]
  
  Pseudo count

- \(\vec{d}_1\): (sport basketball ticket)
- \(\vec{d}_2\): (sport basketball sport)

\(\alpha_k = 2\) Maximum a posterior parameters of multinomial distribution

\((p_{sp}, p_{b}, p_{t}, p_{f}, p_{st}) = (3+1)/(6+5), (2+1)/(6+5), (1+1)/(6+5), 1/(6+5), 1/(6+5)\)

so \((p_{sp} = 0.364, p_{b} = 0.27, p_{t} = 0.18, p_{f} = 0.091, p_{st} = 0.091)\)
Retrieval Model: Language Model

- Introduction to language model
- Unigram language model
- Document language model estimation
  - Maximum Likelihood estimation
  - Maximum a posterior estimation
  - Jelinek Mercer Smoothing
- Model-based feedback
Introduction to Language Models:

- A document **language model** defines a probability distribution over indexed terms
  - E.g., the probability of generating a term
  - Sum of the probabilities is 1

- A query can be seen as observed data from unknown models
  - Query also defines a language model (more on this later)

- How might the models be used for IR?
  - Rank documents by $\text{Pr}(\vec{q} \mid \vec{d}_i)$
  - Rank documents by language models of $\vec{q}$ and $\vec{d}_i$ based on kullback-Leibler (KL) divergence between the models (come later)
Language Model for IR: Example

Estimate the generation probability of $\Pr(\vec{q} | \vec{d}_i)$

Generate retrieval results

Estimating language model for each document

$\vec{q}$
sport, basketball

Language Model for $\vec{d}_1$
sport, basketball, ticket, sport

Language Model for $\vec{d}_2$
basketball, ticket, finance, ticket, sport

Language Model for $\vec{d}_3$
stock, finance, finance, stock
Text Categorization (I)

Outline

● Introduction to the task of text categorization
  ➢ Manual v.s. automatic text categorization
● Text categorization applications
● Evaluation of text categorization
● K nearest neighbor text categorization method
Text Categorization

- Automatic text categorization
  - Learn algorithm to automatically assign predefined categories to text documents /objects
  - automatic or semi-automatic

- Procedures
  - **Training**: Given a set of categories and labeled document examples; learn a method to map a document to correct category (categories)
  - **Testing**: Predict the category (categories) of a new document

- Automatic or semi-automatic categorization can significantly reduce the manual efforts
Text Categorization (I)

Outline

- Naïve Bayes (NB) Classification
- Logistic Regression Classification
Naïve Bayes Classification

- **Representation**
  - Each document is a “bag of words” with weights (e.g., TF.IDF)
  - Each category is a super “bag of words”, which is composed of all words in all the documents associated with the category
  - For all the words in a specific category $c$, it is modeled by a multinominal distribution as
    \[
p(d_{c1}, \ldots, d_{cn_c} | \theta_c)
    \]
  - Each category ($c$) has a prior distribution $P(c)$, which is the probably of choosing category $c$ BEFORE observing the content of a document
Naïve Bayes Classification

- **MLE Estimator: Normalization by simple counting**
  - Train a language model for all the documents in one category

\[
p(w \mid \theta^*_c) = \frac{\sum_{i=1}^{n_c} c_{ci}(w)}{\sum_{i=1}^{n_c} |d_{ci}|}
\]

- **Category Prior:**
  - Number of documents in the category divided by the total number of documents

\[
p(c) = \frac{n_c}{\sum_{c'} n_{c'}}
\]
Naïve Bayes Classification

- Prediction:

\[ c^* = \arg \max_c p(c \mid \vec{d}_i) \]

\[ = \arg \max_c \left\{ \frac{p(c) p(\vec{d}_i \mid c)}{p(\vec{d}_i)} \right\} \]

\[ = \arg \max_c \left\{ p(c) p(\vec{d}_i \mid c) \right\} \quad \text{(Bayes Rule)} \]

\[ = \arg \max_c \left\{ p(c) \prod_{k} p(w_k \mid c)^{c_i(w_k)} \right\} \quad \text{(Multinomial Dist)} \]

\[ = \arg \max_c \left\{ \log(p(c)) + \sum_k c_i(w_k) \log p(w_k \mid c) \right\} \]

Plug in the estimator
Logistic Regression Classification

Directly model probability of generating class conditional on words:  \( p(c | \vec{d}_i) \)

\[
\log \frac{P(C_+ | \vec{d}_i)}{P(C_- | \vec{d}_i)} = \beta_c (0) + \sum_k \beta_c (k) \times c_i (w_k)
\]

\[
P(C_+ | \vec{d}_i) = \frac{\exp \left( \beta_c (0) + \sum_k \beta_c (k) \times c_i (w_k) \right)}{1 + \exp \left( \beta_c (0) + \sum_k \beta_c (k) \times c_i (w_k) \right)}
\]

Sigmod/logistic function:  \( \sigma \left( \beta_c (0) + \sum_k \beta_c (k) \times c_i (w_k) \right) \)

Logistic regression: Tune the parameters to optimize conditional likelihood (class probability predictions)
Collaborative Filtering

Outline

- Introduction to collaborative filtering
- Main framework
- Memory-based collaborative filtering approach
  - Aspect model & Two-way clustering model
  - Flexible mixture model
  - Decouple model
- Model-based collaborative filtering approach
- Unified filtering by combining content and collaborative filtering
Federated Search

Outline

- Introduction to federated search
- Main research problems
  - Resource Representation
  - Resource Selection
  - Results Merging
- A unified utility maximization framework for federated search
- Modeling search engine effectiveness
Information source recommendation: Recommend information sources for users’ text queries (e.g., completeplanet.com): Steps 1 and 2

Federated document retrieval: Also search selected sources and merge individual ranked lists into a single list: Steps 1, 2 and 3
Clustering

Document clustering
- Motivations
- Document representations
- Success criteria

Clustering algorithms
- K-means
- Model-based clustering (EM clustering)
Link Analysis

Outline

- The characteristics of Web structure (small world)
- Hub & Authority Algorithms
  - Authority Value, Hubness Value
- Page Rank Algorithm (Page Rank Value)
- Relation with the computation of eigenvector