DAT630
Learning-to-Rank

Search Engines, Section 7.6

30/10/2017

Krisztian Balog | University of Stavanger

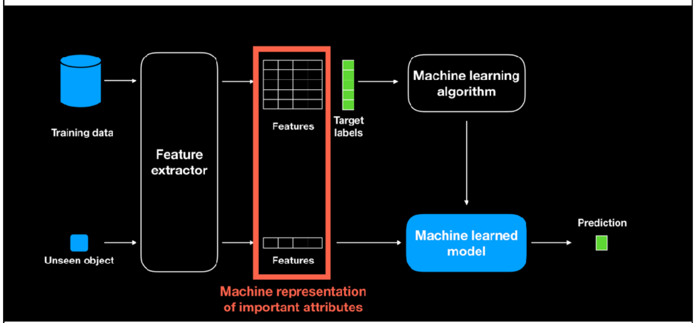
Recap

- Classic retrieval models
  - Vector space model, BM25, LM
- Three main components
  - Term frequency
    - How many times query terms appear in the document
  - Document length
    - Any term is expected to occur more frequently in long document; account for differences in document length
  - Document frequency
    - How often the term appears in the entire collection

Additional factors

- So far: content-based matching
- Many additional signals, e.g.,
  - Document quality
    - PageRank
    - SPAM?
    - …
  - Click-based features
    - How many times users clicked on a document given a query
    - How many times this particular user clicked on a document given the query
    - …

Machine Learning 101



Machine Learning for IR

- The general idea is to use machine learning to combine these *features* to generate the "best" ranking function (referred to as "Learning to Rank")
  - Features can include up to hundreds
  - Impossible to tune by hand
- Modern systems (especially on the Web) use a great number of features
  - In 2008, Google was using over 200 features
    - The New York Times (2008-06-03)

Machine Learning for IR

- Some example features
  - Log frequency of query word in anchor text?
  - Query word in color on page?
  - # of images on page?
  - # of (out) links on page?
  - PageRank of page?
  - URL length?
  - URL contains "~"?
  - Page length?
  - …

Why not earlier?

- Limited training data
  - Especially for other use-cases (i.e., not web search)
- Poor machine learning techniques
- Insufficient customization to IR problem
- Not enough features for ML to show value

Learning-to-Rank (LTR)

- Learn a function automatically to rank items (documents) effectively
- Training data: (item, query, relevance) triples
- Output: ranking function

Pointwise LTR

- The ranking function is based on features of a single object: $h(q,d) = h(\boldsymbol{x})$
  - $\boldsymbol{x}$ is a feature vector
- May be approached classification (relevant vs. non-relevant) or regression (relevance score)
- Note: classic retrieval models are also point-wise: *score(q, d)*

Classification vs. Regression

- Classification
  - Predict a categorical (unordered) output value
- Regression
  - Predict an ordered or continuous output value

Pairwise LTR

- The learning function is based on a pair of items
  - Given two documents, predict partial ranking
- E.g., Ranking SVM
  - Classify which of the two documents should be ranked at a higher position?

Listwise LTR

- The ranking function is based on a ranked list of items
  - Given two ranked list of the same items, which is better?
- Directly optimizes a retrieval metric
  - Need a loss function on a list of documents
- Challenge is scale: huge number of potential lists
- No clear benefits over pairwise ones (so far)

How to?

- **Develop a feature set**
  - The most important step!
  - Usually problem dependent
- **Choose a good ranking algorithm**
  - E.g., Random Forests work well for pairwise LTR
- **Training, validation, and testing**
  - Similar to standard machine learning applications

Exercise

- Design features for an email SPAM classifier
  - You have access to the user's mailbox (all emails, including those that have been classified as SPAM before)
  - For an incoming email, decide whether it is SPAM or not

Features for document retrieval

- Query features
  - Depend only on the query
- Document features
  - Depend only on the document
- Query-document features
  - Express the degree of matching between the query and the document

Query-document features

- Retrieval score of a given document field (e.g., BM25, LM, TF-IDF)
- Retrieval score of the entire document (e.g., BM25F, MLM)
- Sum of TF scores of query terms in a given document field (title, content, anchors, URL, etc)
- ...

Query features

- Query length (number of terms)
- Sum of IDF scores of query terms in a given field (title, content, anchors, etc.)
- Total number of matching documents
- Number of named entities in the query
- ...

Document features

- Length of each document field (title, content, anchors, etc.)
- PageRank score
- Number of inlinks
- Number of outlinks
- Number of slash in URL
- Length of URL
- ...

See also

- LETOR: Learning to Rank for Information Retrieval
  - https://www.microsoft.com/en-us/research/project/letor-learning-rank-information-retrieval/
- Macdonald et al. 2012. On the Usefulness of Query Features for Learning to Rank
  - https://pdfs.semanticscholar.org/dbb5/a414a1168fe2142d9bea2ed561a4a43610bf.pdf

Practical considerations

Feature normalization

- Feature values are often normalized to be in the [0,1] range **for a given query**
  - Esp. matching features that may be on different scales across queries because of query length
- Min-max normalization:

$$\tilde{x}_i = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

- $x_1, \dots, x_n$: original values for a given feature
- $\tilde{x}_i$: normalized value for the *i*th instance

Computation cost

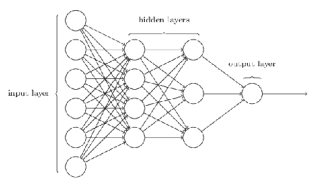
- Implemented as a re-ranking mechanism
  - Retrieve top-N candidate documents using a strong baseline approach (e.g., BM25)
  - Create feature vectors and re-rank these top-N candidates to arrive a the final ranking
- Document features may be computed offline
- Query and query-document features are computed online
  - Avoid using too many expensive features

Class imbalance

- Many more non-relevant than relevant instances
- Classifiers usually do not handle huge imbalance well
- Need to address by over or under sampling

Deep Learning

- Neural networks are a particular type of machine learning, inspired by the architecture of the brain
- Instead of manually engineering features, let the machine learn the representation



Deep Learning