Introduction to Information Retrieval
http://informationretrieval.org

IIR 15-2: Learning to Rank

Hinrich Schütze

Institute for Natural Language Processing, Universität Stuttgart

2011-08-29

Models and Methods

1. Boolean model and its limitations (30)
2. Vector space model (30)
3. Probabilistic models (30)
4. Language model-based retrieval (30)
5. Latent semantic indexing (30)
6. Learning to rank (30)

Take-away

Take-away

- Machine-learned relevance: We use machine learning to learn the relevance score (retrieval status value) of a document with respect to a query.

Take-away

- Machine-learned relevance: We use machine learning to learn the relevance score (retrieval status value) of a document with respect to a query.
- Learning to rank: A machine-learning method that directly optimizes the ranking (as opposed to classification or regression accuracy)

Outline

Machine-learned relevance: Basic idea

Machine-learned relevance: Basic idea

- Given: A training set of examples, each of which is a tuple of: a query q, a document d, a relevance judgment for d on q

Machine-learned relevance: Basic idea

- Given: A training set of examples, each of which is a tuple of: a query q, a document d, a relevance judgment for d on q
- Learn weights from this training set, so that the learned scores approximate the relevance judgments in the training set

Machine-learned relevance vs. Text classification

Machine-learned relevance vs. Text classification

- Both are machine learning approaches

Machine-learned relevance vs. Text classification

- Both are machine learning approaches
- Text classification (if used for information retrieval, e.g., in relevance feedback) is query-specific.

Machine-learned relevance vs. Text classification

- Both are machine learning approaches
- Text classification (if used for information retrieval, e.g., in relevance feedback) is query-specific.
  - We need a query-specific training set to learn the ranker.

Machine-learned relevance vs. Text classification

- Both are machine learning approaches
- Text classification (if used for information retrieval, e.g., in relevance feedback) is query-specific.
  - We need a query-specific training set to learn the ranker.
  - We need to learn a new ranker for each query.

Machine-learned relevance vs. Text classification

- Both are machine learning approaches
- Text classification (if used for information retrieval, e.g., in relevance feedback) is query-specific.
  - We need a query-specific training set to learn the ranker.
  - We need to learn a new ranker for each query.
- Machine-learned relevance and learning to rank usually refer to query-independent ranking.

Machine-learned relevance vs. Text classification

- Both are machine learning approaches
- Text classification (if used for information retrieval, e.g., in relevance feedback) is query-specific.
  - We need a query-specific training set to learn the ranker.
  - We need to learn a new ranker for each query.
- Machine-learned relevance and learning to rank usually refer to query-independent ranking.
- We learn a single classifier or ranker.

Machine-learned relevance vs. Text classification

- Both are machine learning approaches
- Text classification (if used for information retrieval, e.g., in relevance feedback) is query-specific.
  - We need a query-specific training set to learn the ranker.
  - We need to learn a new ranker for each query.
- Machine-learned relevance and learning to rank usually refer to query-independent ranking.
- We learn a single classifier or ranker.
- We can then rank documents for a query that we don't have any relevance judgments for.

Two typical features used in machine-learned relevance

Two typical features used in machine-learned relevance

- The vector space cosine similarity between query and document (denoted α)

Two typical features used in machine-learned relevance

- The vector space cosine similarity between query and document (denoted α)
- The minimum window width within which the query terms lie (denoted ω)

Two typical features used in machine-learned relevance

- The vector space cosine similarity between query and document (denoted α)
- The minimum window width within which the query terms lie (denoted ω)
- Thus, we have

Two typical features used in machine-learned relevance

- The vector space cosine similarity between query and document (denoted α)
- The minimum window width within which the query terms lie (denoted ω)
- Thus, we have
  - one feature (α) that captures overall query-document similarity

Two typical features used in machine-learned relevance

- The vector space cosine similarity between query and document (denoted α)
- The minimum window width within which the query terms lie (denoted ω)
- Thus, we have
  - one feature (α) that captures overall query-document similarity
  - one feature (ω) that captures query term proximity (often indicative of topical relevance) □

Machine-learned relevance: Setup for these two features

Machine-learned relevance: Setup for these two features

Training set

Example	DocID	Query	α	ω	Judgment
Φ_1	37	linux ...	0.032	3	relevant
Φ_2	37	penguin ...	0.02	4	nonrelevant
Φ_3	238	operating system	0.043	2	relevant
Φ_4	238	runtime ...	0.004	2	nonrelevant
Φ_5	1741	kernel layer	0.022	3	relevant
Φ_6	2094	device driver	0.03	2	relevant
Φ_7	3191	device driver	0.027	5	nonrelevant

α is the cosine score. ω is the window width.

Machine-learned relevance: Setup (2)

Machine-learned relevance: Setup (2)

- Two classes: relevant $= 1$ and nonrelevant $= 0$

Machine-learned relevance: Setup (2)

- Two classes: relevant $= 1$ and nonrelevant $= 0$
- We now seek a scoring function that combines the values of the features to generate a value that is (close to) 0 or 1.

Machine-learned relevance: Setup (2)

- Two classes: relevant = 1 and nonrelevant = 0
- We now seek a scoring function that combines the values of the features to generate a value that is (close to) 0 or 1.
- We wish this function to be in agreement with our set of training examples as much as possible.

Machine-learned relevance: Setup (2)

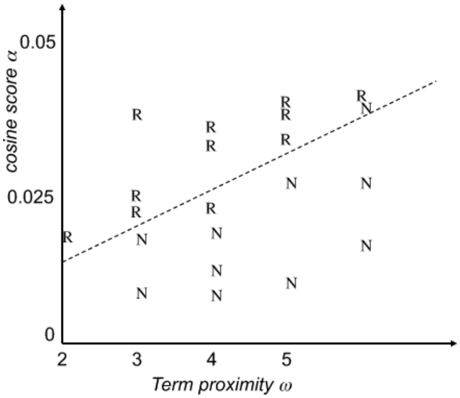
- Two classes: relevant = 1 and nonrelevant = 0
- We now seek a scoring function that combines the values of the features to generate a value that is (close to) 0 or 1.
- We wish this function to be in agreement with our set of training examples as much as possible.
- The simplest classifier is a linear classifier, defined by an equation of the form:

$$Score(d, q) = Score(\alpha, \omega) = a\alpha + b\omega + c,$$

where we learn the coefficients a, b, c from training data. □

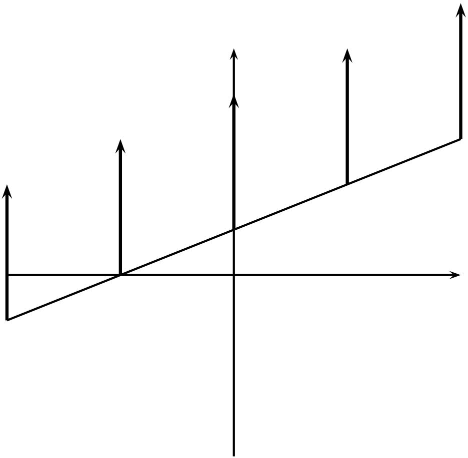
Graphic representation of the training set

Graphic representation of the training set



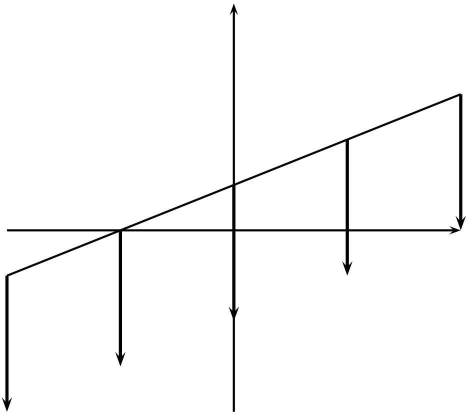
In this case, we learn a linear classifier in 2D

In this case, we learn a linear classifier in 2D



- A linear classifier in 2D is a line described by the equation $w_1 d_1 + w_2 d_2 = \theta$
- Example for a 2D linear classifier
- Points $(d_1 \ d_2)$ with $w_1 d_1 + w_2 d_2 \geq \theta$ are in the class c.
- Points $(d_1 \ d_2)$ with $w_1 d_1 + w_2 d_2 < \theta$ are in the complement class $\overline{c}$.

In this case, we learn a linear classifier in 2D



- A linear classifier in 2D is a line described by the equation $w_1 d_1 + w_2 d_2 = \theta$
- Example for a 2D linear classifier
- Points $(d_1 \ d_2)$ with $w_1 d_1 + w_2 d_2 \geq \theta$ are in the class c.
- Points $(d_1 \ d_2)$ with $w_1 d_1 + w_2 d_2 < \theta$ are in the complement class $\overline{c}$.

Summary

- Machine-learned relevance

Summary

- Machine-learned relevance
  - Assemble a training set of query-document-judgment triples

Summary

- Machine-learned relevance
  - Assemble a training set of query-document-judgment triples
  - Train classification or regression model on training set

Summary

- Machine-learned relevance
  - Assemble a training set of query-document-judgment triples
  - Train classification or regression model on training set
  - For a new query, apply model to all documents (actually: a subset)

Summary

- Machine-learned relevance
  - Assemble a training set of query-document-judgment triples
  - Train classification or regression model on training set
  - For a new query, apply model to all documents (actually: a subset)
  - Rank documents according to model's decisions

Summary

- Machine-learned relevance
  - Assemble a training set of query-document-judgment triples
  - Train classification or regression model on training set
  - For a new query, apply model to all documents (actually: a subset)
  - Rank documents according to model's decisions
  - Return the top K (e.g., $K = 10$) to the user

Summary

- Machine-learned relevance
  - Assemble a training set of query-document-judgment triples
  - Train classification or regression model on training set
  - For a new query, apply model to all documents (actually: a subset)
  - Rank documents according to model's decisions
  - Return the top K (e.g., $K = 10$) to the user
- In principle, any classification/regression method can be used.

Summary

- Machine-learned relevance
  - Assemble a training set of query-document-judgment triples
  - Train classification or regression model on training set
  - For a new query, apply model to all documents (actually: a subset)
  - Rank documents according to model's decisions
  - Return the top K (e.g., $K = 10$) to the user
- In principle, any classification/regression method can be used.
- Big advantage: we avoid hand-tuning scoring functions and simply learn them from training data.

Summary

- Machine-learned relevance
  - Assemble a training set of query-document-judgment triples
  - Train classification or regression model on training set
  - For a new query, apply model to all documents (actually: a subset)
  - Rank documents according to model's decisions
  - Return the top K (e.g., $K = 10$) to the user
- In principle, any classification/regression method can be used.
- Big advantage: we avoid hand-tuning scoring functions and simply learn them from training data.
- Bottleneck: we need to maintain a representative set of training examples whose relevance assessments must be made by humans.

Machine-learned relevance for more than two features

- The approach can be readily generalized to a large number of features.

Machine-learned relevance for more than two features

- The approach can be readily generalized to a large number of features.
- Any measure that can be calculated for a query-document pair is fair game for this approach.

LTR features used by Microsoft Research (1)

LTR features used by Microsoft Research (1)

- Features derived from standard IR models: query term number, query term ratio, length, idf, sum/min/max/mean/variance of term frequency, sum/min/max/mean/variance of length normalized term frequency, sum/min/max/mean/variance of tf-idf weight, boolean model, BM25, LM-absolute-discounting, LM-dirichlet, LM-jelinek-mercer

LTR features used by Microsoft Research (1)

- Features derived from standard IR models: query term number, query term ratio, length, idf, sum/min/max/mean/variance of term frequency, sum/min/max/mean/variance of length normalized term frequency, sum/min/max/mean/variance of tf-idf weight, boolean model, BM25, LM-absolute-discounting, LM-dirichlet, LM-jelinek-mercer
- Most of these features can be computed for different zones: body, anchor, title, url, whole document

LTR features used by Microsoft Research (2)

LTR features used by Microsoft Research (2)

- Web-specific features: number of slashes in url, length of url, inlink number, outlink number, PageRank, SiteRank

LTR features used by Microsoft Research (2)

- Web-specific features: number of slashes in url, length of url, inlink number, outlink number, PageRank, SiteRank
- Spam features: QualityScore

LTR features used by Microsoft Research (2)

- Web-specific features: number of slashes in url, length of url, inlink number, outlink number, PageRank, SiteRank
- Spam features: QualityScore
- Usage-based features: query-url click count, url click count, url dwell time

LTR features used by Microsoft Research (2)

- Web-specific features: number of slashes in url, length of url, inlink number, outlink number, PageRank, SiteRank
- Spam features: QualityScore
- Usage-based features: query-url click count, url click count, url dwell time
- All of these features can be assembled into a big feature vector and then fed into the machine learning algorithm. □

Shortcoming of what we've presented so far

- Approaching IR ranking like we have done so far is not necessarily the right way to think about the problem.

Shortcoming of what we've presented so far

- Approaching IR ranking like we have done so far is not necessarily the right way to think about the problem.
- Statisticians normally first divide problems into classification problems (where a categorical variable is predicted) versus regression problems (where a real number is predicted).

Shortcoming of what we've presented so far

- Approaching IR ranking like we have done so far is not necessarily the right way to think about the problem.
- Statisticians normally first divide problems into classification problems (where a categorical variable is predicted) versus regression problems (where a real number is predicted).
- In between: specialized field of ordinal regression

Shortcoming of what we've presented so far

- Approaching IR ranking like we have done so far is not necessarily the right way to think about the problem.
- Statisticians normally first divide problems into classification problems (where a categorical variable is predicted) versus regression problems (where a real number is predicted).
- In between: specialized field of ordinal regression
- Machine learning for ad hoc retrieval is most properly thought of as an ordinal regression problem.

Shortcoming of what we've presented so far

- Approaching IR ranking like we have done so far is not necessarily the right way to think about the problem.
- Statisticians normally first divide problems into classification problems (where a categorical variable is predicted) versus regression problems (where a real number is predicted).
- In between: specialized field of ordinal regression
- Machine learning for ad hoc retrieval is most properly thought of as an ordinal regression problem.
- Next up: ranking SVMs, a machine learning method that learns an ordering directly.

Outline

Basic setup for ranking SVMs

Basic setup for ranking SVMs

- As before we begin with a set of judged query-document pairs.

Basic setup for ranking SVMs

- As before we begin with a set of judged query-document pairs.
- But we do not represent them as query-document-judgment triples.

Basic setup for ranking SVMs

- As before we begin with a set of judged query-document pairs.
- But we do not represent them as query-document-judgment triples.
- Instead, we ask judges, for each training query q, to order the documents that were returned by the search engine with respect to relevance to the query.

Basic setup for ranking SVMs

- As before we begin with a set of judged query-document pairs.
- But we do not represent them as query-document-judgment triples.
- Instead, we ask judges, for each training query q, to order the documents that were returned by the search engine with respect to relevance to the query.
- We again construct a vector of features $\psi_j = \psi(d_j, q)$ for each document-query pair – exactly as we did before.

Basic setup for ranking SVMs

- As before we begin with a set of judged query-document pairs.
- But we do not represent them as query-document-judgment triples.
- Instead, we ask judges, for each training query q, to order the documents that were returned by the search engine with respect to relevance to the query.
- We again construct a vector of features $\psi_j = \psi(d_j, q)$ for each document-query pair – exactly as we did before.
- For two documents d_i and d_j, we then form the vector of feature differences:

$$\Phi(d_i, d_j, q) = \psi(d_i, q) - \psi(d_j, q)$$

Training a ranking SVM

- Vector of feature differences: $\Phi(d_i, d_j, q) = \psi(d_i, q) - \psi(d_j, q)$

Training a ranking SVM

- Vector of feature differences: $\Phi(d_i, d_j, q) = \psi(d_i, q) - \psi(d_j, q)$
- By hypothesis, one of d_i and d_j has been judged more relevant.

Training a ranking SVM

- Vector of feature differences: $\Phi(d_i, d_j, q) = \psi(d_i, q) - \psi(d_j, q)$
- By hypothesis, one of d_i and d_j has been judged more relevant.
- Notation: We write $d_i \prec d_j$ for "d_i precedes d_j in the results ordering".

Training a ranking SVM

- Vector of feature differences: $\Phi(d_i, d_j, q) = \psi(d_i, q) - \psi(d_j, q)$
- By hypothesis, one of d_i and d_j has been judged more relevant.
- Notation: We write $d_i \prec d_j$ for "d_i precedes d_j in the results ordering".
- If d_i is judged more relevant than d_j, then we will assign the vector $\Phi(d_i, d_j, q)$ the class $y_{ijq} = +1$; otherwise -1.

Training a ranking SVM

- Vector of feature differences: $\Phi(d_i, d_j, q) = \psi(d_i, q) - \psi(d_j, q)$
- By hypothesis, one of d_i and d_j has been judged more relevant.
- Notation: We write $d_i \prec d_j$ for "d_i precedes d_j in the results ordering".
- If d_i is judged more relevant than d_j, then we will assign the vector $\Phi(d_i, d_j, q)$ the class $y_{ijq} = +1$; otherwise -1.
- This gives us a training set of pairs of vectors and "precedence indicators".

Training a ranking SVM

- Vector of feature differences: $\Phi(d_i, d_j, q) = \psi(d_i, q) - \psi(d_j, q)$
- By hypothesis, one of d_i and d_j has been judged more relevant.
- Notation: We write $d_i \prec d_j$ for "d_i precedes d_j in the results ordering".
- If d_i is judged more relevant than d_j, then we will assign the vector $\Phi(d_i, d_j, q)$ the class $y_{ijq} = +1$; otherwise -1.
- This gives us a training set of pairs of vectors and "precedence indicators".
- We can then train an SVM on this training set with the goal of obtaining a classifier that returns

$$\vec{w}^{\mathsf{T}}\Phi(d_i, d_j, q) > 0 \quad \text{iff} \quad d_i \prec d_j$$

Advantages of Ranking SVMs vs. Classification/regression

Advantages of Ranking SVMs vs. Classification/regression

- Documents can be evaluated relative to other candidate documents for the same query . . .

Advantages of Ranking SVMs vs. Classification/regression

- Documents can be evaluated relative to other candidate documents for the same query ...
- ...rather than having to be mapped to a global scale of goodness.

Advantages of Ranking SVMs vs. Classification/regression

- Documents can be evaluated relative to other candidate documents for the same query . . .
- . . . rather than having to be mapped to a global scale of goodness.
- This often is an easier problem to solve since just a ranking is required rather than an absolute measure of relevance.

Why simple ranking SVMs don't work that well

Why simple ranking SVMs don't work that well

- Ranking SVMs treat all ranking violations alike.

Why simple ranking SVMs don't work that well

- Ranking SVMs treat all ranking violations alike.
    - But some violations are minor problems, e.g., getting the order of two relevant documents wrong.

Why simple ranking SVMs don't work that well

- Ranking SVMs treat all ranking violations alike.
  - But some violations are minor problems, e.g., getting the order of two relevant documents wrong.
  - Other violations are big problems, e.g., ranking a nonrelevant document ahead of a relevant document.

Why simple ranking SVMs don't work that well

- Ranking SVMs treat all ranking violations alike.
  - But some violations are minor problems, e.g., getting the order of two relevant documents wrong.
  - Other violations are big problems, e.g., ranking a nonrelevant document ahead of a relevant document.
- In most IR settings, getting the order of the top documents right is key.

Why simple ranking SVMs don't work that well

- Ranking SVMs treat all ranking violations alike.
  - But some violations are minor problems, e.g., getting the order of two relevant documents wrong.
  - Other violations are big problems, e.g., ranking a nonrelevant document ahead of a relevant document.
- In most IR settings, getting the order of the top documents right is key.
  - In the simple setting we have described, top and bottom ranks will not be treated differently.

Why simple ranking SVMs don't work that well

- Ranking SVMs treat all ranking violations alike.
  - But some violations are minor problems, e.g., getting the order of two relevant documents wrong.
  - Other violations are big problems, e.g., ranking a nonrelevant document ahead of a relevant document.
- In most IR settings, getting the order of the top documents right is key.
  - In the simple setting we have described, top and bottom ranks will not be treated differently.
- $\rightarrow$ Learning-to-rank frameworks actually used in IR are more complicated than what we have presented here. □

Example for superior performance of LTR

Example for superior performance of LTR

SVM algorithm that directly optimizes MAP (as opposed to ranking).

Example for superior performance of LTR

SVM algorithm that directly optimizes MAP (as opposed to ranking).
Proposed by: Yue, Finley, Radlinski, Joachims, ACM SIGIR 2002.

Example for superior performance of LTR

SVM algorithm that directly optimizes MAP (as opposed to ranking).

Proposed by: Yue, Finley, Radlinski, Joachims, ACM SIGIR 2002.

Performance compared to state-of-the-art models: cosine, tf-idf, BM25, language models (Dirichlet and Jelinek-Mercer)

Example for superior performance of LTR

SVM algorithm that directly optimizes MAP (as opposed to ranking).
Proposed by: Yue, Finley, Radlinski, Joachims, ACM SIGIR 2002.
Performance compared to state-of-the-art models: cosine, tf-idf, BM25, language models (Dirichlet and Jelinek-Mercer)

	TREC 9		TREC 10	
Model	MAP	W/L	MAP	W/L
SVM^{Δ}_{map}	0.242	–	0.236	–
Best Func.	0.204	39/11 **	0.181	37/13 **
2nd Best	0.199	38/12 **	0.174	43/7 **
3rd Best	0.188	34/16 **	0.174	38/12 **

Example for superior performance of LTR

SVM algorithm that directly optimizes MAP (as opposed to ranking).
Proposed by: Yue, Finley, Radlinski, Joachims, ACM SIGIR 2002.
Performance compared to state-of-the-art models: cosine, tf-idf, BM25, language models (Dirichlet and Jelinek-Mercer)

	TREC 9		TREC 10	
Model	MAP	W/L	MAP	W/L
$\text{SVM}^{\Delta}_{map}$	0.242	–	0.236	–
Best Func.	0.204	39/11 **	0.181	37/13 **
2nd Best	0.199	38/12 **	0.174	43/7 **
3rd Best	0.188	34/16 **	0.174	38/12 **

Learning-to-rank clearly better than non-machine-learning approaches

Assessment of learning to rank

Assessment of learning to rank

- The idea of learning to rank is old.

Assessment of learning to rank

- The idea of learning to rank is old.
  - Early work by Norbert Fuhr and William S. Cooper

Assessment of learning to rank

- The idea of learning to rank is old.
    - Early work by Norbert Fuhr and William S. Cooper
- Renewed recent interest due to:

Assessment of learning to rank

- The idea of learning to rank is old.
  - Early work by Norbert Fuhr and William S. Cooper
- Renewed recent interest due to:
  - Better machine learning methods becoming available

Assessment of learning to rank

- The idea of learning to rank is old.
  - Early work by Norbert Fuhr and William S. Cooper
- Renewed recent interest due to:
  - Better machine learning methods becoming available
  - More computational power

Assessment of learning to rank

- The idea of learning to rank is old.
  - Early work by Norbert Fuhr and William S. Cooper
- Renewed recent interest due to:
  - Better machine learning methods becoming available
  - More computational power
  - Willingness to pay for large annotated training sets

Assessment of learning to rank

- The idea of learning to rank is old.
    - Early work by Norbert Fuhr and William S. Cooper
- Renewed recent interest due to:
    - Better machine learning methods becoming available
    - More computational power
    - Willingness to pay for large annotated training sets
- Strengths of learning-to-rank

Assessment of learning to rank

- The idea of learning to rank is old.
  - Early work by Norbert Fuhr and William S. Cooper
- Renewed recent interest due to:
  - Better machine learning methods becoming available
  - More computational power
  - Willingness to pay for large annotated training sets
- Strengths of learning-to-rank
  - Humans are bad at fine-tuning a ranking function with dozens of parameters.

Assessment of learning to rank

- The idea of learning to rank is old.
  - Early work by Norbert Fuhr and William S. Cooper
- Renewed recent interest due to:
  - Better machine learning methods becoming available
  - More computational power
  - Willingness to pay for large annotated training sets
- Strengths of learning-to-rank
  - Humans are bad at fine-tuning a ranking function with dozens of parameters.
  - Machine-learning methods are good at it.

Assessment of learning to rank

- The idea of learning to rank is old.
  - Early work by Norbert Fuhr and William S. Cooper
- Renewed recent interest due to:
  - Better machine learning methods becoming available
  - More computational power
  - Willingness to pay for large annotated training sets
- Strengths of learning-to-rank
  - Humans are bad at fine-tuning a ranking function with dozens of parameters.
  - Machine-learning methods are good at it.
  - Web search engines use a large number of features → web search engines need some form of learning to rank.

Information retrieval models: Pros and Cons

Information retrieval models: Pros and Cons

- Least effort: Boolean system

Information retrieval models: Pros and Cons

- Least effort: Boolean system
  - In general, low user satisfaction

Information retrieval models: Pros and Cons

- Least effort: Boolean system
  - In general, low user satisfaction
- A little bit more effort: Vector space model

Information retrieval models: Pros and Cons

- Least effort: Boolean system
  - In general, low user satisfaction
- A little bit more effort: Vector space model
  - Acceptable performance in many cases

Information retrieval models: Pros and Cons

- Least effort: Boolean system
  - In general, low user satisfaction
- A little bit more effort: Vector space model
  - Acceptable performance in many cases
- State-of-the-art performance: BM25, LMs

Information retrieval models: Pros and Cons

- Least effort: Boolean system
  - In general, low user satisfaction
- A little bit more effort: Vector space model
  - Acceptable performance in many cases
- State-of-the-art performance: BM25, LMs
  - You need to tune parameters.

Information retrieval models: Pros and Cons

- Least effort: Boolean system
  - In general, low user satisfaction
- A little bit more effort: Vector space model
  - Acceptable performance in many cases
- State-of-the-art performance: BM25, LMs
  - You need to tune parameters.
- Best performance: learning to rank

Information retrieval models: Pros and Cons

- Least effort: Boolean system
  - In general, low user satisfaction
- A little bit more effort: Vector space model
  - Acceptable performance in many cases
- State-of-the-art performance: BM25, LMs
  - You need to tune parameters.
- Best performance: learning to rank
  - But you need an expensive training set

Information retrieval models: Pros and Cons

- Least effort: Boolean system
  - In general, low user satisfaction
- A little bit more effort: Vector space model
  - Acceptable performance in many cases
- State-of-the-art performance: BM25, LMs
  - You need to tune parameters.
- Best performance: learning to rank
  - But you need an expensive training set
- Noisy data or vocabulary mismatch queries/documents & no time to custom-build a solution & collection is not too large

Information retrieval models: Pros and Cons

- Least effort: Boolean system
  - In general, low user satisfaction
- A little bit more effort: Vector space model
  - Acceptable performance in many cases
- State-of-the-art performance: BM25, LMs
  - You need to tune parameters.
- Best performance: learning to rank
  - But you need an expensive training set
- Noisy data or vocabulary mismatch queries/documents & no time to custom-build a solution & collection is not too large
  - Use Latent Semantic Indexing

Take-away

- Machine-learned relevance: We use machine learning to learn the relevance score (retrieval status value) of a document with respect to a query.
- Learning to rank: A machine-learning method that directly optimizes the ranking (as opposed to classification or regression accuracy)

Resources

- Chapter 15 of Introduction to Information Retrieval
- Resources at http://informationretrieval.org/essir2011
    - References to learning to rank literature
    - Microsoft learning to rank datasets
    - How Google tweaks ranking

Exercise

Exercise

Write down the training set from the last exercise as a training set for a ranking SVM.