Learning to Rank from Relevance Feedback

Peter Lubell-Doughtie

University of Amsterdam

peter@helioid.com

supervised by Maarten van Someren and Katja Hofmann

August 29th, 2011

Learning to Rank from Relevance Feedback

- Search is inherently interactive, how can we use this to make search better?

Learning to Rank from Relevance Feedback

- Search is inherently interactive, how can we use this to make search better?
- Our method uses judged documents for query expansion and then for learning to rank

Learning to Rank from Relevance Feedback

- Search is inherently interactive, how can we use this to make search better?
- Our method uses judged documents for query expansion and then for learning to rank
- We improve search result ordering beyond that achieved when using only query expansion

Learning to Rank from Relevance Feedback

- Search is inherently interactive, how can we use this to make search better?
- Our method uses judged documents for query expansion and then for learning to rank
- We improve search result ordering beyond that achieved when using only query expansion
- Success depends on how we extract features

Learning to Rank from Relevance Feedback

- Search is inherently interactive, how can we use this to make search better?
- Our method uses judged documents for query expansion and then for learning to rank
- We improve search result ordering beyond that achieved when using only query expansion
- Success depends on how we extract features
- And on how we request document judgements

Contents

- Motivation
- Background
- Method
- Experiments and Results
- Conclusions

Motivation: Learn from Users

Motivation: Learn from Users

Search needs extra help when

Motivation: Learn from Users

Search needs extra help when

- Queries and documents contain ambiguous or domain specific terms

Motivation: Learn from Users

Search needs extra help when

- Queries and documents contain ambiguous or domain specific terms
- The user wants to retrieve many or all relevant documents

Motivation: Learn from Users

Search needs extra help when

- Queries and documents contain ambiguous or domain specific terms
- The user wants to retrieve many or all relevant documents
- There are multiple interactions with the search system

Motivation: Learn from Users

Search needs extra help when

- Queries and documents contain ambiguous or domain specific terms
- The user wants to retrieve many or all relevant documents
- There are multiple interactions with the search system

Search is interactive, let's learn from this

Motivation: Learn from Users

Search needs extra help when

- Queries and documents contain ambiguous or domain specific terms
- The user wants to retrieve many or all relevant documents
- There are multiple interactions with the search system

Search is interactive, let's learn from this

- Interaction is especially useful for difficult search tasks

Motivation: Learn from Users

Search needs extra help when

- Queries and documents contain ambiguous or domain specific terms
- The user wants to retrieve many or all relevant documents
- There are multiple interactions with the search system

Search is interactive, let's learn from this

- Interaction is especially useful for difficult search tasks
- But providing explicit feedback is costly for the user

Motivation: Learn from Users

Search needs extra help when

- Queries and documents contain ambiguous or domain specific terms
- The user wants to retrieve many or all relevant documents
- There are multiple interactions with the search system

Search is interactive, let's learn from this

- Interaction is especially useful for difficult search tasks
- But providing explicit feedback is costly for the user
- We must use the feedback we receive well

Overarching Research Question

How can a system that is aware of the interactive nature of search exploit this interactivity to better serve the user's needs?

Background: Learning to Rank from *Relevance Feedback*

Background: Learning to Rank from *Relevance Feedback*

The Rocchio Algorithm and its Discontents

The Rocchio Algorithm and its Discontents

- Combines an initial query, a positive contribution from relevant documents, and a negative contribution from non-relevant documents

Background: Learning to Rank from *Relevance Feedback*

The Rocchio Algorithm and its Discontents

- Combines an initial query, a positive contribution from relevant documents, and a negative contribution from non-relevant documents
- Judged documents only affect ordering through the expanded query

Background: Learning to Rank from *Relevance Feedback*

The Rocchio Algorithm and its Discontents

- Combines an initial query, a positive contribution from relevant documents, and a negative contribution from non-relevant documents
- Judged documents only affect ordering through the expanded query
- Negative feedback is often ignored because it is hard to use

Background: Learning to Rank from *Relevance Feedback*

The Rocchio Algorithm and its Discontents

- Combines an initial query, a positive contribution from relevant documents, and a negative contribution from non-relevant documents
- Judged documents only affect ordering through the expanded query
- Negative feedback is often ignored because it is hard to use
- But negative feedback is found to be especially beneficial for difficult queries (Wang and Wang et al.)

Background: Learning to Rank from *Relevance Feedback*

Our method addresses these challenges

Background: Learning to Rank from *Relevance Feedback*

Our method addresses these challenges

- To make better use of judged documents

Background: Learning to Rank from *Relevance Feedback*

Our method addresses these challenges

- To make better use of judged documents
- We use feedback in both learning to rank and query expansion

Background: Learning to Rank from *Relevance Feedback*

Our method addresses these challenges

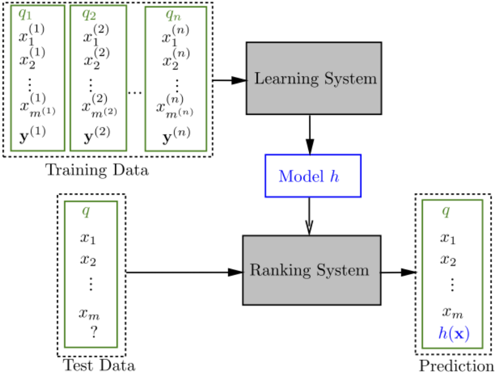
- To make better use of judged documents
- We use feedback in both learning to rank and query expansion
- To avoid the problems of negative feedback

Background: Learning to Rank from *Relevance Feedback*

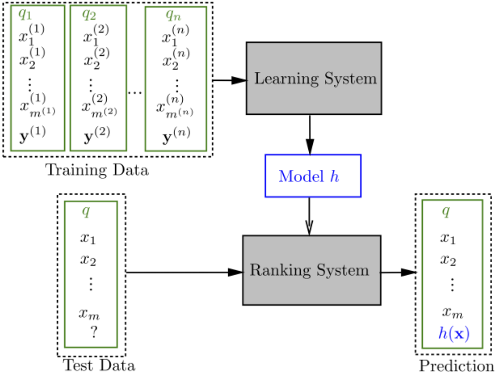
Our method addresses these challenges

- To make better use of judged documents
- We use feedback in both learning to rank and query expansion
- To avoid the problems of negative feedback
- We use non-relevant documents in learning to rank, not query expansion

Background: *Learning to Rank* from Relevance Feedback

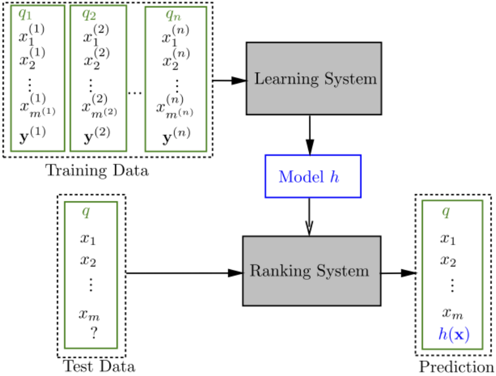


Background: *Learning to Rank* from Relevance Feedback



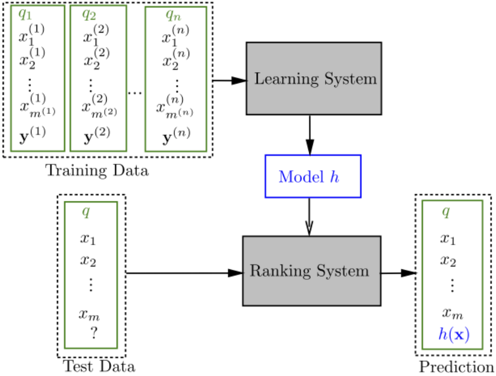
- Extract feature vectors from data

Background: *Learning to Rank* from Relevance Feedback



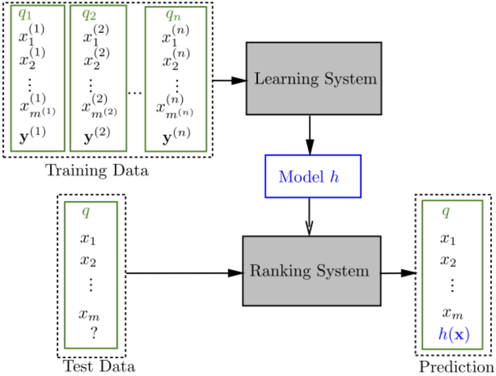
- Extract feature vectors from data
- Use the training data to learn a model *h*

Background: *Learning to Rank* from Relevance Feedback



- Extract feature vectors from data
- Use the training data to learn a model h
- Use this model to predict the relevance of unlabeled test data

Background: *Learning to Rank* from Relevance Feedback



- Extract feature vectors from data
- Use the training data to learn a model h
- Use this model to predict the relevance of unlabeled test data
- Sort documents by their predicted relevance

Background: *Learning to Rank* from Relevance Feedback

Pairwise Learning to Rank

Background: *Learning to Rank* from Relevance Feedback

Pairwise Learning to Rank

- Use document judgements to construct a preference pair set

Background: *Learning to Rank* from Relevance Feedback

Pairwise Learning to Rank

- Use document judgements to construct a preference pair set
- (i, j) in preference pairs iff document i prefered to document j

Background: *Learning to Rank* from Relevance Feedback

Pairwise Learning to Rank

- Use document judgements to construct a preference pair set
- (i, j) in preference pairs iff document i prefered to document j
- E.g. if $\mathbf{d}_1$ and $\mathbf{d}_2$ are preferred to $\mathbf{d}_3$, then $\mathcal{P} = \{(\mathbf{d}_1, \mathbf{d}_3), (\mathbf{d}_2, \mathbf{d}_3)\}$

Background: *Learning to Rank* from Relevance Feedback

Pairwise Learning to Rank

- Use document judgements to construct a preference pair set
- (i, j) in preference pairs iff document i prefered to document j
- E.g. if $\mathbf{d}_1$ and $\mathbf{d}_2$ are preferred to $\mathbf{d}_3$, then $\mathcal{P} = \{(\mathbf{d}_1, \mathbf{d}_3), (\mathbf{d}_2, \mathbf{d}_3)\}$

Minimize Incorrectly Ordered Pairs

Background: *Learning to Rank* from Relevance Feedback

Pairwise Learning to Rank

- Use document judgements to construct a preference pair set
- (i, j) in preference pairs iff document i prefered to document j
- E.g. if $\mathbf{d}_1$ and $\mathbf{d}_2$ are preferred to $\mathbf{d}_3$, then $\mathcal{P} = \{(\mathbf{d}_1, \mathbf{d}_3), (\mathbf{d}_2, \mathbf{d}_3)\}$

Minimize Incorrectly Ordered Pairs

- Using feature vectors for documents in the preference pair set

Background: *Learning to Rank* from Relevance Feedback

Pairwise Learning to Rank

- Use document judgements to construct a preference pair set
- (i, j) in preference pairs iff document i prefered to document j
- E.g. if $\mathbf{d}_1$ and $\mathbf{d}_2$ are preferred to $\mathbf{d}_3$, then $\mathcal{P} = \{(\mathbf{d}_1, \mathbf{d}_3), (\mathbf{d}_2, \mathbf{d}_3)\}$

Minimize Incorrectly Ordered Pairs

- Using feature vectors for documents in the preference pair set
- Calculate a weight vector over features

Background: *Learning to Rank* from Relevance Feedback

Pairwise Learning to Rank

- Use document judgements to construct a preference pair set
- (i, j) in preference pairs iff document i prefered to document j
- E.g. if $\mathbf{d}_1$ and $\mathbf{d}_2$ are preferred to $\mathbf{d}_3$, then $\mathcal{P} = \{(\mathbf{d}_1, \mathbf{d}_3), (\mathbf{d}_2, \mathbf{d}_3)\}$

Minimize Incorrectly Ordered Pairs

- Using feature vectors for documents in the preference pair set
- Calculate a weight vector over features
- Score unlabeled documents using this weight vector

Background: A Difficult Recall Oriented Task

Background: A Difficult Recall Oriented Task

e-Discovery and TREC Legal

Background: A Difficult Recall Oriented Task

e-Discovery and TREC Legal

- Ideally the search system returns *all* documents relevant to the query

Background: A Difficult Recall Oriented Task

e-Discovery and TREC Legal

- Ideally the search system returns *all* documents relevant to the query
- Ordered with the most relevant first

Background: A Difficult Recall Oriented Task

e-Discovery and TREC Legal

- Ideally the search system returns *all* documents relevant to the query
- Ordered with the most relevant first

- A hard task, the best systems have trouble with some queries

Background: A Difficult Recall Oriented Task

e-Discovery and TREC Legal

- Ideally the search system returns *all* documents relevant to the query
- Ordered with the most relevant first

- A hard task, the best systems have trouble with some queries
- Learning is important, previous TREC Legal participants' systems benefit substantially from incorporating feedback

Background: A Difficult Recall Oriented Task

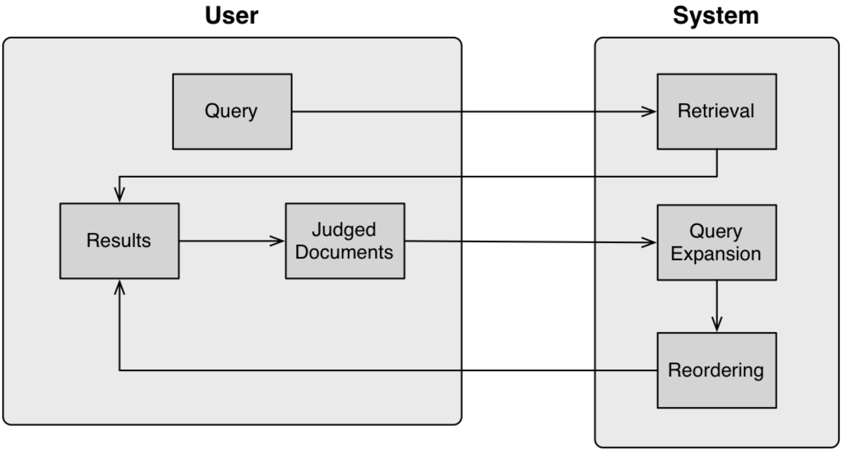
e-Discovery and TREC Legal

- Ideally the search system returns *all* documents relevant to the query
- Ordered with the most relevant first

- A hard task, the best systems have trouble with some queries
- Learning is important, previous TREC Legal participants' systems benefit substantially from incorporating feedback
- We can simulate feedback by using already judged documents

Method Overview

Method Overview



Open Questions

Open Questions

- What documents should the user judged?

Open Questions

- What documents should the user judged?
- How do we represent documents for learning?

How to Request Judgements: Exploitative

Exploitative Sampling Strategy

How to Request Judgements: Exploitative

Exploitative Sampling Strategy

- Suppose we have a set of *seed* documents we expect are informative

How to Request Judgements: Exploitative

Exploitative Sampling Strategy

- Suppose we have a set of *seed* documents we expect are informative
- Sample these documents descending from the highest ranked

How to Request Judgements: Exploitative

Exploitative Sampling Strategy

- Suppose we have a set of *seed* documents we expect are informative
- Sample these documents descending from the highest ranked

Example

- Suppose the returned documents are ordered: $\mathbf{d}_1^* > \mathbf{d}_2 > \mathbf{d}_3^*$

How to Request Judgements: Exploitative

Exploitative Sampling Strategy

- Suppose we have a set of *seed* documents we expect are informative
- Sample these documents descending from the highest ranked

Example

- Suppose the returned documents are ordered: $\mathbf{d}_1^* > \mathbf{d}_2 > \mathbf{d}_3^*$
- Those marked $*$ are expected to be informative

How to Request Judgements: Exploitative

Exploitative Sampling Strategy

- Suppose we have a set of *seed* documents we expect are informative
- Sample these documents descending from the highest ranked

Example

- Suppose the returned documents are ordered: $\mathbf{d}_1^* > \mathbf{d}_2 > \mathbf{d}_3^*$
- Those marked * are expected to be informative
- We would request judgements on $\mathbf{d}_1$ and then $\mathbf{d}_3$

How to Request Judgements: Exploitative

Exploitative Sampling Strategy

- Suppose we have a set of *seed* documents we expect are informative
- Sample these documents descending from the highest ranked

Example

- Suppose the returned documents are ordered: $\mathbf{d}_1^* > \mathbf{d}_2 > \mathbf{d}_3^*$
- Those marked * are expected to be informative
- We would request judgements on $\mathbf{d}_1$ and then $\mathbf{d}_3$

- Close to the way documents are judged in a real-life search system

How to Request Judgements: Exploitative

Exploitative Sampling Strategy

- Suppose we have a set of *seed* documents we expect are informative
- Sample these documents descending from the highest ranked

Example

- Suppose the returned documents are ordered: $\mathbf{d}_1^* > \mathbf{d}_2 > \mathbf{d}_3^*$
- Those marked * are expected to be informative
- We would request judgements on $\mathbf{d}_1$ and then $\mathbf{d}_3$

- Close to the way documents are judged in a real-life search system
- Biases judged documents towards those ranked higher

How to Request Judgements: Random Seeds

Random Seed Sampling Strategy

How to Request Judgements: Random Seeds

Random Seed Sampling Strategy

- Randomly sample from the set of informative documents

How to Request Judgements: Random Seeds

Random Seed Sampling Strategy

- Randomly sample from the set of informative documents
- Same document set as in exploitative sampling

How to Request Judgements: Random Seeds

Random Seed Sampling Strategy

- Randomly sample from the set of informative documents
- Same document set as in exploitative sampling
- No rank based bias

Open Questions

Open Questions

- What documents should the user judge?
  1. Exploitative
  2. Random

Open Questions

- What documents should the user judge?
  1. Exploitative
  2. Random
- How do we represent documents for learning?

Representing Documents: Cumulative Features

Representing Documents: Cumulative Features

- Features are the retrieval scores for queries expanded with different sets of judged documents

Representing Documents: Cumulative Features

- Features are the retrieval scores for queries expanded with different sets of judged documents

Example

Representing Documents: Cumulative Features

- Features are the retrieval scores for queries expanded with different sets of judged documents

Example

- Suppose there are three documents

Representing Documents: Cumulative Features

- Features are the retrieval scores for queries expanded with different sets of judged documents

Example

- Suppose there are three documents
- The retrieval scores before expansion are $\langle 1, 2, 3 \rangle$

Representing Documents: Cumulative Features

- Features are the retrieval scores for queries expanded with different sets of judged documents

Example

- Suppose there are three documents
- The retrieval scores before expansion are $\langle 1, 2, 3 \rangle$
- After judging one document on the first iteartion

Representing Documents: Cumulative Features

- Features are the retrieval scores for queries expanded with different sets of judged documents

Example

- Suppose there are three documents
- The retrieval scores before expansion are $\langle 1, 2, 3 \rangle$
- After judging one document on the first iteartion
- The document profile is $P = \{\mathbf{d}\}$ and the expansion term is t_1

Representing Documents: Cumulative Features

- Features are the retrieval scores for queries expanded with different sets of judged documents

Example

- Suppose there are three documents
- The retrieval scores before expansion are $\langle 1, 2, 3 \rangle$
- After judging one document on the first iteartion
- The document profile is $P = \{\mathbf{d}\}$ and the expansion term is t_1
- Retrieval scores with the expanded query are $\langle 1, 1, 3 \rangle$

Representing Documents: Cumulative Features

- Features are the retrieval scores for queries expanded with different sets of judged documents

Example

- Suppose there are three documents
- The retrieval scores before expansion are $\langle 1, 2, 3 \rangle$
- After judging one document on the first iteartion
- The document profile is $P = \{\mathbf{d}\}$ and the expansion term is t_1
- Retrieval scores with the expanded query are $\langle 1, 1, 3 \rangle$
- We are only collecting features up to and including the first iteration

Representing Documents: Cumulative Features

- Features are the retrieval scores for queries expanded with different sets of judged documents

Example

- Suppose there are three documents
- The retrieval scores before expansion are $\langle 1, 2, 3 \rangle$
- After judging one document on the first iteartion
- The document profile is $P = \{\mathbf{d}\}$ and the expansion term is t_1
- Retrieval scores with the expanded query are $\langle 1, 1, 3 \rangle$
- We are only collecting features up to and including the first iteration
- We are reordering the top two documents

Example: Cumulative Features

The input to the *cumulative* feature extractor $\mathbf{\Phi}$ will be:

$$\begin{bmatrix} \{\mathbf{q}_0, \emptyset, \langle 1,2,3 \rangle\} \\ \{\mathbf{q}_0 \circ t_1, \{\mathbf{d}\}, \langle 1,1,3 \rangle\} \\ \vdots \end{bmatrix},$$

Example: Cumulative Features

The input to the *cumulative* feature extractor $\mathbf{\Phi}$ will be:

$$\begin{bmatrix} \{\mathbf{q}_0, \emptyset, \langle 1, 2, 3 \rangle\} \\ \{\mathbf{q}_0 \circ t_1, \{\mathbf{d}\}, \langle 1, 1, 3 \rangle\} \\ \vdots \end{bmatrix},$$

and the output features will be:

$$\begin{bmatrix} 1, 2 \\ 1, 1 \end{bmatrix}.$$

Example: Cumulative Features

The input to the *cumulative* feature extractor $\mathbf{\Phi}$ will be:

$$\begin{bmatrix} \{\mathbf{q}_0, \emptyset, \langle 1,2,3 \rangle\} \\ \{\mathbf{q}_0 \circ t_1, \{\mathbf{d}\}, \langle 1,1,3 \rangle\} \\ \vdots \end{bmatrix},$$

and the output features will be:

$$\begin{bmatrix} 1,2 \\ 1,1 \end{bmatrix}.$$

- The top row are the retrieval scores without expansion

Example: Cumulative Features

The input to the *cumulative* feature extractor $\boldsymbol{\Phi}$ will be:

$$\begin{bmatrix} \{\mathbf{q}_0, \emptyset, \langle 1,2,3 \rangle\} \\ \{\mathbf{q}_0 \circ t_1, \{\mathbf{d}\}, \langle 1,1,3 \rangle\} \\ \vdots \end{bmatrix},$$

and the output features will be:

$$\begin{bmatrix} 1,2 \\ 1,1 \end{bmatrix}.$$

- The top row are the retrieval scores without expansion
- The bottom row are retrieval scores with expansion

Example: Cumulative Features

The input to the *cumulative* feature extractor $\mathbf{\Phi}$ will be:

$$\begin{bmatrix} \{\mathbf{q}_0, \emptyset, \langle 1, 2, 3 \rangle\} \\ \{\mathbf{q}_0 \circ t_1, \{\mathbf{d}\}, \langle 1, 1, 3 \rangle\} \\ \vdots \end{bmatrix},$$

and the output features will be:

$$\begin{bmatrix} 1, 2 \\ 1, 1 \end{bmatrix}.$$

- The top row are the retrieval scores without expansion
- The bottom row are retrieval scores with expansion
- The first column are features for the first document

Example: Cumulative Features

The input to the *cumulative* feature extractor $\mathbf{\Phi}$ will be:

$$\begin{bmatrix} \{\mathbf{q}_0, \emptyset, \langle 1, 2, 3 \rangle\} \\ \{\mathbf{q}_0 \circ t_1, \{\mathbf{d}\}, \langle 1, 1, 3 \rangle\} \\ \vdots \end{bmatrix},$$

and the output features will be:

$$\begin{bmatrix} 1, 2 \\ 1, 1 \end{bmatrix}.$$

- The top row are the retrieval scores without expansion
- The bottom row are retrieval scores with expansion
- The first column are features for the first document
- The second column are features for the second document

Representing Documents: Constant Features

Features are the retrieval scores for the same set of judged documents but using different numbers of expansion terms

Representing Documents: Constant Features

Features are the retrieval scores for the same set of judged documents but using different numbers of expansion terms

Example

Representing Documents: Constant Features

Features are the retrieval scores for the same set of judged documents but using different numbers of expansion terms

Example

- Suppose there are three documents

Representing Documents: Constant Features

Features are the retrieval scores for the same set of judged documents but using different numbers of expansion terms

Example

- Suppose there are three documents
- The judged document profile is $P = \{\mathbf{d}\}$

Representing Documents: Constant Features

Features are the retrieval scores for the same set of judged documents but using different numbers of expansion terms

Example

- Suppose there are three documents
- The judged document profile is $P = \{\mathbf{d}\}$
- The first two expansion terms are $\{t_1, t_2\}$

Representing Documents: Constant Features

Features are the retrieval scores for the same set of judged documents but using different numbers of expansion terms

Example

- Suppose there are three documents
- The judged document profile is $P = \{\mathbf{d}\}$
- The first two expansion terms are $\{t_1, t_2\}$
- The retrieval scores when expanding with t_1 are $\langle 2, 2, 3\rangle$

Representing Documents: Constant Features

Features are the retrieval scores for the same set of judged documents but using different numbers of expansion terms

Example

- Suppose there are three documents
- The judged document profile is $P = \{\mathbf{d}\}$
- The first two expansion terms are $\{t_1, t_2\}$
- The retrieval scores when expanding with t_1 are $\langle 2, 2, 3 \rangle$
- The retrieval scores when expanding with t_1 and t_2 the retreival scores are $\langle 1, 2, 3 \rangle$

Representing Documents: Constant Features

Features are the retrieval scores for the same set of judged documents but using different numbers of expansion terms

Example

- Suppose there are three documents
- The judged document profile is $P = \{\mathbf{d}\}$
- The first two expansion terms are $\{t_1, t_2\}$
- The retrieval scores when expanding with t_1 are $\langle 2, 2, 3 \rangle$
- The retrieval scores when expanding with t_1 and t_2 the retreival scores are $\langle 1, 2, 3 \rangle$
- We are reordering the top two documents

Example: Constant Features

Our input to the *constant* feature extractor $\mathbf{\Phi}$ will be:

$$\begin{bmatrix} \{\mathbf{q}_0 \circ t_1, \{\mathbf{d}\}, \langle 2, 2, 3\rangle\} \\ \{\mathbf{q}_0 \circ t_1 \circ t_2, \{\mathbf{d}\}, \langle 1, 2, 3\rangle\} \end{bmatrix},$$

Example: Constant Features

Our input to the *constant* feature extractor $\mathbf{\Phi}$ will be:

$$\begin{bmatrix} \{\mathbf{q}_0 \circ t_1, \{\mathbf{d}\}, \langle 2, 2, 3 \rangle\} \\ \{\mathbf{q}_0 \circ t_1 \circ t_2, \{\mathbf{d}\}, \langle 1, 2, 3 \rangle\} \end{bmatrix},$$

and the output features will be:

$$\begin{bmatrix} 2, 2 \\ 1, 2 \end{bmatrix}.$$

Example: Constant Features

Our input to the *constant* feature extractor $\mathbf{\Phi}$ will be:

$$\begin{bmatrix} \{\mathbf{q}_0 \circ t_1, \{\mathbf{d}\}, \langle 2, 2, 3 \rangle\} \\ \{\mathbf{q}_0 \circ t_1 \circ t_2, \{\mathbf{d}\}, \langle 1, 2, 3 \rangle\} \end{bmatrix},$$

and the output features will be:

$$\begin{bmatrix} 2, 2 \\ 1, 2 \end{bmatrix}.$$

- The top row are the retrieval scores for the query $\mathbf{q}_0 \circ t_1$

Example: Constant Features

Our input to the *constant* feature extractor $\mathbf{\Phi}$ will be:

$$\begin{bmatrix} \{\mathbf{q}_0 \circ t_1, \{\mathbf{d}\}, \langle 2, 2, 3 \rangle\} \\ \{\mathbf{q}_0 \circ t_1 \circ t_2, \{\mathbf{d}\}, \langle 1, 2, 3 \rangle\} \end{bmatrix},$$

and the output features will be:

$$\begin{bmatrix} 2, 2 \\ 1, 2 \end{bmatrix}.$$

- The top row are the retrieval scores for the query $\mathbf{q}_0 \circ t_1$
- The bottom row are retrieval scores for the query $\mathbf{q}_0 \circ t_1 \circ t_2$

Example: Constant Features

Our input to the *constant* feature extractor $\boldsymbol{\Phi}$ will be:

$$\begin{bmatrix} \{\mathbf{q}_0 \circ t_1, \{\mathbf{d}\}, \langle 2, 2, 3 \rangle\} \\ \{\mathbf{q}_0 \circ t_1 \circ t_2, \{\mathbf{d}\}, \langle 1, 2, 3 \rangle\} \end{bmatrix},$$

and the output features will be:

$$\begin{bmatrix} 2, 2 \\ 1, 2 \end{bmatrix}.$$

- The top row are the retrieval scores for the query $\mathbf{q}_0 \circ t_1$
- The bottom row are retrieval scores for the query $\mathbf{q}_0 \circ t_1 \circ t_2$
- The first column are features for the first document

Example: Constant Features

Our input to the *constant* feature extractor $\boldsymbol{\Phi}$ will be:

$$\begin{bmatrix} \{\mathbf{q}_0 \circ t_1, \{\mathbf{d}\}, \langle 2, 2, 3 \rangle\} \\ \{\mathbf{q}_0 \circ t_1 \circ t_2, \{\mathbf{d}\}, \langle 1, 2, 3 \rangle\} \end{bmatrix},$$

and the output features will be:

$$\begin{bmatrix} 2, 2 \\ 1, 2 \end{bmatrix}.$$

- The top row are the retrieval scores for the query $\mathbf{q}_0 \circ t_1$
- The bottom row are retrieval scores for the query $\mathbf{q}_0 \circ t_1 \circ t_2$
- The first column are features for the first document
- The second column are features for the second document

Example: Term Frequency Features

- Features are the term frequency of the most common terms in judged documents and highly ranked documents

Example: Term Frequency Features

- Features are the term frequency of the most common terms in judged documents and highly ranked documents

Example

Example: Term Frequency Features

- Features are the term frequency of the most common terms in judged documents and highly ranked documents

Example

- Suppose there are three documents

Example: Term Frequency Features

- Features are the term frequency of the most common terms in judged documents and highly ranked documents

Example

- Suppose there are three documents
- Their retrieval scores are $\langle 1, 2, 3 \rangle$

Example: Term Frequency Features

- Features are the term frequency of the most common terms in judged documents and highly ranked documents

Example

- Suppose there are three documents
- Their retrieval scores are $\langle 1, 2, 3 \rangle$
- The term based features for the query are $\mathbf{f_q} = [t_1, t_2]$

Example: Term Frequency Features

- Features are the term frequency of the most common terms in judged documents and highly ranked documents

Example

- Suppose there are three documents
- Their retrieval scores are $\langle 1, 2, 3 \rangle$
- The term based features for the query are $\mathbf{f_q} = [t_1, t_2]$
- For the first document the frequency of t_1 is 1 and that of t_2 is 2, i.e. its term frequency vector is $[1, 2]$

Example: Term Frequency Features

- Features are the term frequency of the most common terms in judged documents and highly ranked documents

Example

- Suppose there are three documents
- Their retrieval scores are $\langle 1, 2, 3 \rangle$
- The term based features for the query are $\mathbf{f_q} = [t_1, t_2]$
- For the first document the frequency of t_1 is 1 and that of t_2 is 2, i.e. its term frequency vector is $[1, 2]$
- For the second document the frequency of t_1 is 1 and that of t_2 is 1, i.e. its term frequency vector is $[1, 1]$

Example: Term Frequency Features

- Features are the term frequency of the most common terms in judged documents and highly ranked documents

Example

- Suppose there are three documents
- Their retrieval scores are $\langle 1, 2, 3 \rangle$
- The term based features for the query are $\mathbf{f_q} = [t_1, t_2]$
- For the first document the frequency of t_1 is 1 and that of t_2 is 2, i.e. its term frequency vector is $[1, 2]$
- For the second document the frequency of t_1 is 1 and that of t_2 is 1, i.e. its term frequency vector is $[1, 1]$
- We are reordering the top two documents

Example: Term Frequency Features

Our input to the term frequency feature extractor $\mathbf{\Phi}$ will be:

$$\left[\{\mathbf{q}, P_{\mathbf{q}}, \langle 1, 2, 3\rangle\}\right],$$

Example: Term Frequency Features

Our input to the term frequency feature extractor $\mathbf{\Phi}$ will be:

$$\left[\{\mathbf{q}, P_{\mathbf{q}}, \langle 1, 2, 3 \rangle\}\right],$$

and we define the output features as the term frequency vectors:

$$\begin{bmatrix} 1, 1 \\ 2, 1 \end{bmatrix}.$$

Example: Term Frequency Features

Our input to the term frequency feature extractor $\mathbf{\Phi}$ will be:

$$\left[\{\mathbf{q}, P_{\mathbf{q}}, \langle 1, 2, 3\rangle\}\right],$$

and we define the output features as the term frequency vectors:

$$\begin{bmatrix} 1, 1 \\ 2, 1 \end{bmatrix}.$$

- The top row are the term frequencies of t_1

Example: Term Frequency Features

Our input to the term frequency feature extractor $\mathbf{\Phi}$ will be:

$$\left[\{\mathbf{q}, P_{\mathbf{q}}, \langle 1, 2, 3 \rangle\}\right],$$

and we define the output features as the term frequency vectors:

$$\begin{bmatrix} 1, 1 \\ 2, 1 \end{bmatrix}.$$

- The top row are the term frequencies of t_1
- The bottom row are the term frequencies of t_2

Example: Term Frequency Features

Our input to the term frequency feature extractor $\mathbf{\Phi}$ will be:

$$\left[\{\mathbf{q}, P_{\mathbf{q}}, \langle 1, 2, 3 \rangle\}\right],$$

and we define the output features as the term frequency vectors:

$$\begin{bmatrix} 1, 1 \\ 2, 1 \end{bmatrix}.$$

- The top row are the term frequencies of t_1
- The bottom row are the term frequencies of t_2
- The first column are features for the first document

Example: Term Frequency Features

Our input to the term frequency feature extractor $\mathbf{\Phi}$ will be:

$$\left[\{\mathbf{q}, P_{\mathbf{q}}, \langle 1, 2, 3\rangle\}\right],$$

and we define the output features as the term frequency vectors:

$$\begin{bmatrix} 1, 1 \\ 2, 1 \end{bmatrix}.$$

- The top row are the term frequencies of t_1
- The bottom row are the term frequencies of t_2
- The first column are features for the first document
- The second column are features for the second document

Open Questions

Open Questions

- What documents should the user judge?
  1. Exploitative: choose highly ranked first
  2. Random: choose at random

Open Questions

- What documents should the user judge?
  1. Exploitative: choose highly ranked first
  2. Random: choose at random
- How do we represent documents for learning?
  1. Cumulative: retrieval scores for different document sets
  2. Constant: retrieval scores with different numbers of expansion terms
  3. Term frequency: vector space model

Learning to Rank improves Query Expansion

Learning to Rank improves Query Expansion

Random Seeds and Constant Features

Metric	Baseline	Expansion	Learning
MAP	0.0674	$^{\triangle}$0.130	$^{\triangle}$**0.137**
NDCG	0.565	$^{\triangle}$0.646	$^{\blacktriangle}$**0.652**

Learning to Rank improves Query Expansion

Random Seeds and Constant Features

Metric	Baseline	Expansion	Learning
MAP	0.0674	$^{\triangle}$0.130	$^{\triangle}$**0.137**
NDCG	0.565	$^{\triangle}$0.646	$^{\blacktriangle}$**0.652**

- *Baseline* is language model retrieval with the original query

Learning to Rank improves Query Expansion

Random Seeds and Constant Features

Metric	Baseline	Expansion	Learning
MAP	0.0674	$^{\triangle}$0.130	$^{\triangle}$**0.137**
NDCG	0.565	$^{\triangle}$0.646	$^{\blacktriangle}$**0.652**

- *Baseline* is language model retrieval with the original query
- *Expansion* is language model retrieval with the expanded query

Learning to Rank improves Query Expansion

Random Seeds and Constant Features

Metric	Baseline	Expansion	Learning
MAP	0.0674	$^{\triangle}$0.130	$^{\triangle}$**0.137**
NDCG	0.565	$^{\triangle}$0.646	$^{\blacktriangle}$**0.652**

- *Baseline* is language model retrieval with the original query
- *Expansion* is language model retrieval with the expanded query
- Learning from 20 relevant and 20 non-relelvant documents per query

Learning to Rank improves Query Expansion

Random Seeds and Constant Features

Metric	Baseline	Expansion	Learning
MAP	0.0674	$^{\triangle}$0.130	$^{\triangle}$**0.137**
NDCG	0.565	$^{\triangle}$0.646	$^{\blacktriangle}$**0.652**

- *Baseline* is language model retrieval with the original query
- *Expansion* is language model retrieval with the expanded query
- Learning from 20 relevant and 20 non-relelvant documents per query
- Scores are averaged over all (8) queries

Learning to Rank improves Query Expansion

Random Seeds and Constant Features

Metric	Baseline	Expansion	Learning
MAP	0.0674	$^{\triangle}$0.130	$^{\triangle}$**0.137**
NDCG	0.565	$^{\triangle}$0.646	$^{\blacktriangle}$**0.652**

- *Baseline* is language model retrieval with the original query
- *Expansion* is language model retrieval with the expanded query
- Learning from 20 relevant and 20 non-relelvant documents per query
- Scores are averaged over all (8) queries
- $^{\triangle}$ are statistically significant at the 0.01 level and $^{\blacktriangle}$ at the 0.001 level

Learning to Rank improves Query Expansion

*By combining traditional query expansion with learning to rank,
a search system can use the interactive nature of search
to better serve the user's needs.*

How to Best Learn from Documents

How to Best Learn from Documents

- The best method for representing documents is codependent with the best method for requesting document judgements

How to Best Learn from Documents

- The best method for representing documents is codependent with the best method for requesting document judgements

Experiment

- Compare evaluation scores when using different feature spaces

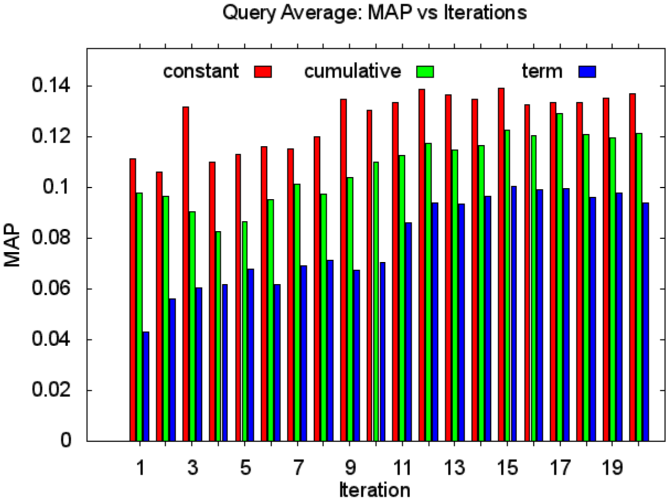
How to Best Learn from Documents

- The best method for representing documents is codependent with the best method for requesting document judgements

Experiment

- Compare evaluation scores when using different feature spaces
- Compare evaluation scores when using different strategies to choose documents for judgement

Retrieval score features outperform term frequency features when using random sampling



Why do retrieval score features outperform term frequency features when using random sampling?

Why do retrieval score features outperform term frequency features when using random sampling?

- Random sampling judges documents at many different ranks

Why do retrieval score features outperform term frequency features when using random sampling?

- Random sampling judges documents at many different ranks
- This provides an unbiased picture of the relationship between retrieval scores and document judgements

Why do retrieval score features outperform term frequency features when using random sampling?

- Random sampling judges documents at many different ranks
- This provides an unbiased picture of the relationship between retrieval scores and document judgements
- Learning to rank with low ranked documents could increase the rank of other relevant but incorrectly low ranked documents

Why do retrieval score features outperform term frequency features when using random sampling?

- Random sampling judges documents at many different ranks
- This provides an unbiased picture of the relationship between retrieval scores and document judgements
- Learning to rank with low ranked documents could increase the rank of other relevant but incorrectly low ranked documents
- This could significantly improve scores

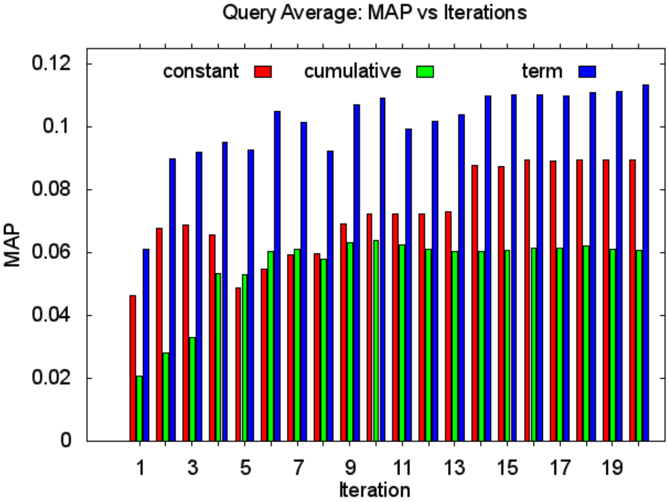
Why do retrieval score features outperform term frequency features when using random sampling?

- Random sampling judges documents at many different ranks
- This provides an unbiased picture of the relationship between retrieval scores and document judgements
- Learning to rank with low ranked documents could increase the rank of other relevant but incorrectly low ranked documents
- This could significantly improve scores
- Term frequency features only consider content and ignore rank

Why do retrieval score features outperform term frequency features when using random sampling?

- Random sampling judges documents at many different ranks
- This provides an unbiased picture of the relationship between retrieval scores and document judgements
- Learning to rank with low ranked documents could increase the rank of other relevant but incorrectly low ranked documents
- This could significantly improve scores
- Term frequency features only consider content and ignore rank
- Content of low ranked documents may be uninformative

Term frequency features outperform retrieval score features for exploitative sampling



Why do term frequency features outperform retrieval score features when using exploitative sampling?

Why do term frequency features outperform retrieval score features when using exploitative sampling?

- Exploitative sampling overrepresents highly ranked judged documents

Why do term frequency features outperform retrieval score features when using exploitative sampling?

- Exploitative sampling overrepresents highly ranked judged documents
- With term frequency features the content of judged documents must be helpful for reordering

Why do term frequency features outperform retrieval score features when using exploitative sampling?

- Exploitative sampling overrepresents highly ranked judged documents
- With term frequency features the content of judged documents must be helpful for reordering
- Our retrieval method orders documents based on the relevance of their terms

Why do term frequency features outperform retrieval score features when using exploitative sampling?

- Exploitative sampling overrepresents highly ranked judged documents
- With term frequency features the content of judged documents must be helpful for reordering
- Our retrieval method orders documents based on the relevance of their terms
- Terms in highly ranked documents are likely to contain more informative terms than those from random documents

Why do term frequency features outperform retrieval score features when using exploitative sampling?

- Exploitative sampling overrepresents highly ranked judged documents
- With term frequency features the content of judged documents must be helpful for reordering
- Our retrieval method orders documents based on the relevance of their terms
- Terms in highly ranked documents are likely to contain more informative terms than those from random documents
- *frequency*(top five terms in exploitative sampled documents) $= 5 \times$ *frequency*(top five terms in random sampled documents)

Summary and Conclusions

- Our method combines learning to rank and query expansion

Summary and Conclusions

- Our method combines learning to rank and query expansion
- This improves search result ordering beyond no expansion and beyond using only query expansion

Summary and Conclusions

- Our method combines learning to rank and query expansion
- This improves search result ordering beyond no expansion and beyond using only query expansion
- Both constant and term frequency features can improve ordering

Summary and Conclusions

- Our method combines learning to rank and query expansion
- This improves search result ordering beyond no expansion and beyond using only query expansion
- Both constant and term frequency features can improve ordering
- TREC Legal is not significantly different from other search task, such as ad-hoc web page search

Summary and Conclusions

- Our method combines learning to rank and query expansion
- This improves search result ordering beyond no expansion and beyond using only query expansion
- Both constant and term frequency features can improve ordering
- TREC Legal is not significantly different from other search task, such as ad-hoc web page search
- Our method's improvement in scores may be a general result that will apply to other corpuses

Future Work

- Design a better hybrid method

Future Work

- Design a better hybrid method
- Apply to additional corpora

Questions?