Queries

Overview

Concepts covered:

- (Pseudo-)Relevance feedback
- Query/document expansion/reduction
- Query performance prediction

Queries

Relevance Feedback Overview

Idea: Modify the query given additional documents such that the query is more like (potentially) relevant documents.

Two possible directions:

- For retrieval models that use vectors to represent queries, can use Rocchio's update formula.
- For all other retrieval models that represent queries as text, perform some kind of query expansion conditioned on documents.

What could we do if we don't have explicit relevance assessments?

Queries

Relevance Feedback Overview

Idea: Modify the query given additional documents such that the query is more like (potentially) relevant documents.

Two possible directions:

- For retrieval models that use vectors to represent queries, can use Rocchio's update formula.
- For all other retrieval models that represent queries as text, perform some kind of query expansion conditioned on documents.

What could we do if we don't have explicit relevance assessments?

Pseudo-relevance Feedback

- Explicit relevance judgements are hard to collect.
- Instead: assume top-k documents initially retrieved contain relevance signals.

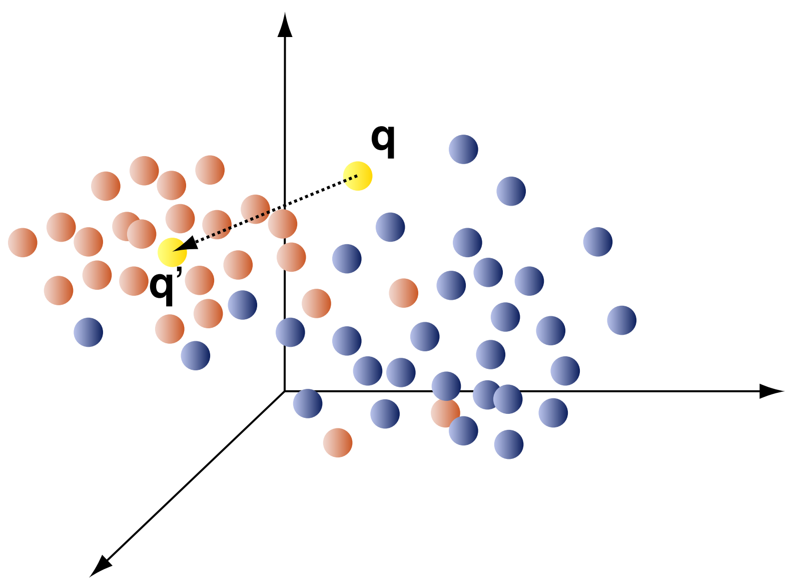
Queries

Rocchio's Update Formula

Given a result set R for a query q, and subsets $R^+ \subseteq R$ and $R^- \subseteq R$ of relevant and non-relevant documents, where $R^+ \cap R^- = \emptyset$, the query representation $\mathbf{q}$ can be refined with the document representations $\mathbf{R}$ using Rocchio's update formula:

$$\mathbf{q}' = \quad \alpha \cdot \mathbf{q} \quad + \quad \beta \cdot \frac{1}{|\mathbf{R}^+|} \sum_{\mathbf{d}^+ \in \mathbf{R}^+} \mathbf{d}^+ \quad - \quad \gamma \cdot \frac{1}{|\mathbf{R}^-|} \sum_{\mathbf{d}^- \in \mathbf{R}^-} \mathbf{d}^-,$$

where α, β, and γ adjust the impact of original query and (non-)relevant documents.



Queries

Rocchio's Update Formula

Given a result set R for a query q, and subsets $R^+ \subseteq R$ and $R^- \subseteq R$ of relevant and non-relevant documents, where $R^+ \cap R^- = \emptyset$, the query representation $\mathbf{q}$ can be refined with the document representations $\mathbf{R}$ using Rocchio's update formula:

$$\mathbf{q}' = \quad \alpha \cdot \mathbf{q} \quad + \quad \beta \cdot \frac{1}{|\mathbf{R}^+|} \sum_{\mathbf{d}^+ \in \mathbf{R}^+} \mathbf{d}^+ \quad - \quad \gamma \cdot \frac{1}{|\mathbf{R}^-|} \sum_{\mathbf{d}^- \in \mathbf{R}^-} \mathbf{d}^-,$$

where α, β, and γ adjust the impact of original query and (non-)relevant documents.

Observations:

- Terms not in query q may get added; often a limit is imposed (say, 50).
- Terms may accrue negative weight; such weights are set to 0.
- Moves the query vector closer to the centroid of relevant documents.
- Works well if relevant documents cluster; less suited for multi-faceted topics.

Relevance feedback can be obtained directly from the user, indirectly through user interaction, or automatically assuming the top-retrieved documents as relevant.

Queries

Relevance Feedback for Bi-Encoders [Li et al. 2022]

Average of the query and document vectors:

$$\mathbf{q}' = \frac{1}{1 + |\mathbf{R}^+|} \quad \mathbf{q} + \sum_{\mathbf{d}^+ \in \mathbf{R}^+} \mathbf{d}^+$$

Adapt the original Rocchio method to bi-encoders:

$$\mathbf{q}' = \alpha \cdot \mathbf{q} \quad + \quad \beta \cdot \frac{1}{|\mathbf{R}^+|} \sum_{\mathbf{d}^+ \in \mathbf{R}^+} \mathbf{d}^+$$

Queries

Relevance Feedback for Bi-Encoders [Li et al. 2022]

Average of the query and document vectors:

$$\mathbf{q}' = \frac{1}{1 + |\mathbf{R}^+|} \quad \mathbf{q} + \sum_{\mathbf{d}^+ \in \mathbf{R}^+} \mathbf{d}^+$$

Adapt the original Rocchio method to bi-encoders:

$$\mathbf{q}' = \alpha \cdot \mathbf{q} \quad + \quad \beta \cdot \frac{1}{|\mathbf{R}^+|} \sum_{\mathbf{d}^+ \in \mathbf{R}^+} \mathbf{d}^+$$

Observations:

- Rocchio method generally leads to more effective rankings.
- Both approaches do not model negative feedback.
- Big gains in effectiveness with negligible query latency trade-off.
  - Computing $\mathbf{q}'$ is cheap, and do not need to use encoder for second round.

Language Models

Relevance Function ρ: Summary

$$\rho(\mathbf{d},\mathbf{q}) \quad = \quad P(\mathbf{d} \mid \mathbf{q}) \quad \propto \quad P(\mathbf{d}) \cdot \prod_{i=1}^{|q|} \frac{\mathit{tf}(t_i,d) + \alpha \cdot \frac{\sum_{d \in D} \mathit{tf}(t_i,d)}{\sum_{d \in D} |d|}}{|d| + \alpha}$$

Assumptions:

1. The user has a mental model of the desired document and generates the query from that model.
2. The equation represents a probability estimate that the document the user had in mind was in fact this one.
3. Independence of word occurrence in documents.
4. Terms not in query q are equally likely to occur in relevant and irrelevant documents.
5. The prior $P(\mathbf{d})$ may be chosen uniform for all documents, or to boost more important documents.

Queries

Language Model Relevance Feedback

Given a query q, let R^* denote the subset of relevant documents from document collection D. Every $d \in R^*$ and q are samples drawn from their relevance model $\mathbf{R}^*$.

$$P(\mathbf{d} \mid \mathbf{q}) \quad \stackrel{\text{rank}}{=} \quad -\mathit{KL}(\mathbf{R}^* \,||\, \mathbf{d}) \tag{1}$$

(1) Rank-preserving approximation by measuring the negative statistical difference between the language model $\mathbf{d}$ and that of the set $\mathbf{R}^*$ of relevant documents to query q using the Kullback–Leibler (KL) divergence measure.

Queries

Language Model Relevance Feedback

Given a query q, let R^* denote the subset of relevant documents from document collection D. Every $d \in R^*$ and q are samples drawn from their relevance model $\mathbf{R}^*$.

$$P(\mathbf{d} \mid \mathbf{q}) \quad \stackrel{\text{rank}}{=} \quad -\sum_{t \in T} P(t \mid \mathbf{R}^*) \log \frac{P(t \mid \mathbf{R}^*)}{P(t \mid \mathbf{d})} \qquad (1)$$

(1) Rank-preserving approximation by measuring the negative statistical difference between the language model $\mathbf{d}$ and that of the set $\mathbf{R}^*$ of relevant documents to query q using the Kullback–Leibler (KL) divergence measure.

Queries

Language Model Relevance Feedback

Given a query q, let R^* denote the subset of relevant documents from document collection D. Every $d \in R^*$ and q are samples drawn from their relevance model $\mathbf{R}^*$.

$$P(\mathbf{d} \mid \mathbf{q}) \stackrel{\text{rank}}{=} -\sum_{t \in T} P(t \mid \mathbf{R}^*) \log \frac{P(t \mid \mathbf{R}^*)}{P(t \mid \mathbf{d})} \quad (1)$$

$$= \sum_{t \in T} P(t \mid \mathbf{R}^*) \log P(t \mid \mathbf{d}) - \sum_{t \in T} P(t \mid \mathbf{R}^*) \log P(t \mid \mathbf{R}^*) \quad (2)$$

(1) Rank-preserving approximation by measuring the negative statistical difference between the language model $\mathbf{d}$ and that of the set $\mathbf{R}^*$ of relevant documents to query q using the Kullback–Leibler (KL) divergence measure.

(2) Rearrangement.

Queries

Language Model Relevance Feedback

Given a query q, let R^* denote the subset of relevant documents from document collection D. Every $d \in R^*$ and q are samples drawn from their relevance model $\mathbf{R}^*$.

$$
\begin{aligned}
P(\mathbf{d} \mid \mathbf{q}) \quad & \overset{\text{rank}}{=} -\sum_{t \in T} P(t \mid \mathbf{R}^*) \log \frac{P(t \mid \mathbf{R}^*)}{P(t \mid \mathbf{d})} && (1) \\
& = \sum_{t \in T} P(t \mid \mathbf{R}^*) \log P(t \mid \mathbf{d}) - \sum_{t \in T} P(t \mid \mathbf{R}^*) \log P(t \mid \mathbf{R}^*) && (2) \\
& \overset{\text{rank}}{=} \sum_{t \in T} P(t \mid \mathbf{R}^*) \log P(t \mid \mathbf{d}) && (3)
\end{aligned}
$$

(1) Rank-preserving approximation by measuring the negative statistical difference between the language model $\mathbf{d}$ and that of the set $\mathbf{R}^*$ of relevant documents to query q using the Kullback–Leibler (KL) divergence measure.

(2) Rearrangement.

(3) Rank-preserving omission of the second sum; it does not depend on $\mathbf{d}$.

Remarks:

- The Kullback-Leibler divergence $\mathit{KL}(P \,||\, Q)$ (also called relative entropy) is a measure of how probability distribution Q diverges from an expected probability distribution P. It is a distribution-wise asymmetric measure. A Kullback-Leibler divergence of 0 indicates that we can expect similar, if not the same, behavior of two different distributions, while a Kullback-Leibler divergence of 1 indicates that the two distributions behave in such a different manner that the expectation given the first distribution approaches zero.

  In applications, P typically represents the "true" distribution of data, observations, or a precisely calculated theoretical distribution, while Q typically represents a theory, model, description, or approximation of P. [Wikipedia]

- Idea: Estimate $P(d \mid \mathbf{R}^*)$ directly, i.e., the probability that the relevance model generates document d. This is called the document likelihood model, but it does not work in practice: the estimates for d heavily depend on its length $|d|$ and are therefore hardly comparable for documents of different lengths.

- Idea: Simply estimating $P(t \mid \mathbf{R}^*)$ with the maximum likelihood estimate of t occurring in q:

$$P(t \mid \mathbf{R}^*) = \frac{\mathit{tf}(t,q)}{|q|}$$

  yields

$$P(\mathbf{d} \mid \mathbf{q}) \quad \overset{\text{rank}}{=} \quad \sum_{t \in T} \frac{\mathit{tf}(t,q)}{|q|} \log P(t \mid \mathbf{d}) \quad = \quad \frac{1}{|q|} \sum_{t \in T} \log P(t \mid \mathbf{d})^{\mathit{tf}(t,q)}$$

  which is equivalent to the query likelihood model.

Queries

Language Model Relevance Feedback

Since R^* is unknown at query time, we cannot approximate its language model directly. But we can exploit that, by definition, q has been sampled from $\mathbf{R}^*$.

$$P(t \mid \mathbf{R}^*) \quad \approx \quad P(t \mid \mathbf{q}) \quad = \quad P(t \mid t_1, \ \ldots, \ t_{|q|}) \quad \text{for } t_i \in \mathbf{q} \qquad (4)$$

(4) Approximation as probability of observing term t given query q: sampling the sequence $\mathbf{q} = (t_1, \ \ldots, \ t_{|q|})$ from $\mathbf{R}^*$, what is the probability of sampling t next?

Queries

Language Model Relevance Feedback

Since R^* is unknown at query time, we cannot approximate its language model directly. But we can exploit that, by definition, q has been sampled from $\mathbf{R}^*$.

$$P(t \mid \mathbf{R}^*) \approx P(t \mid \mathbf{q}) = P(t \mid t_1, \ldots, t_{|q|}) \quad \text{for } t_i \in \mathbf{q} \tag{4}$$

$$= \frac{P(t, t_1, \ldots, t_{|q|})}{P(t_1, \ldots, t_{|q|})} \tag{5}$$

(4) Approximation as probability of observing term t given query q: sampling the sequence $\mathbf{q} = (t_1, \ldots, t_{|q|})$ from $\mathbf{R}^*$, what is the probability of sampling t next?

(5) Definition of conditional probability.

Queries

Language Model Relevance Feedback

Since R^* is unknown at query time, we cannot approximate its language model directly. But we can exploit that, by definition, q has been sampled from $\mathbf{R}^*$.

$$P(t \mid \mathbf{R}^*) \approx P(t \mid \mathbf{q}) = P(t \mid t_1, \ldots, t_{|q|}) \quad \text{for } t_i \in \mathbf{q} \tag{4}$$

$$= \frac{P(t, t_1, \ldots, t_{|q|})}{P(t_1, \ldots, t_{|q|})} \tag{5}$$

$$= \frac{P(t, t_1, \ldots, t_{|q|})}{\sum_{t \in T} P(t, t_1, \ldots, t_{|q|})} \tag{6}$$

(4) Approximation as probability of observing term t given query q: sampling the sequence $\mathbf{q} = (t_1, \ldots, t_{|q|})$ from $\mathbf{R}^*$, what is the probability of sampling t next?

(5) Definition of conditional probability.

(6) Ensures additivity of the model.

Queries

Language Model Relevance Feedback

Let $R^+ \subseteq R^*$ be a set of documents relevant to query q, which have been obtained via relevance feedback.

$$P(t,\ t_1,\ \ldots,\ t_{|q|}) \quad \approx \sum_{d \in R^+} P(\mathbf{d}) \cdot P(t,\ t_1,\ \ldots,\ t_{|q|} \mid \mathbf{d}) \qquad (7)$$

(7) Approximation based on the law of total probability, using the language models of the individual relevant documents.

Queries

Language Model Relevance Feedback

Let $R^+ \subseteq R^*$ be a set of documents relevant to query q, which have been obtained via relevance feedback.

$$
\begin{aligned}
P(t,\, t_1,\, \ldots,\, t_{|q|}) \quad &\approx \sum_{d \in R^+} P(\mathbf{d}) \cdot P(t,\, t_1,\, \ldots,\, t_{|q|} \mid \mathbf{d}) && (7)\\
&= \sum_{d \in R^+} P(\mathbf{d}) \cdot P(t \mid \mathbf{d}) \cdot \prod_{i=1}^{|q|} P(t_i \mid \mathbf{d}) && (8)
\end{aligned}
$$

(7) Approximation based on the law of total probability, using the language models of the individual relevant documents.

(8) Assumption that the query's terms $t_1,\, \ldots,\, t_{|q|}$ are independent of one another, as well as from t.

Queries

Language Model Relevance Feedback

$$\rho(\mathbf{d},\mathbf{q}) \;=\; P(\mathbf{d}\,|\,\mathbf{q}) \;\propto\; \sum_{t\in T} \frac{\sum_{d'\in R^+} P(\mathbf{d'}) \cdot P(t \mid \mathbf{d'}) \cdot \prod_{i=1}^{|q|} P(t_i \mid \mathbf{d'})}{\sum_{t\in T}\sum_{d'\in R^+} P(\mathbf{d'}) \cdot P(t \mid \mathbf{d'}) \cdot \prod_{i=1}^{|q|} P(t_i \mid \mathbf{d'})} \cdot \log P(t \mid \mathbf{d})$$

Retrieval:

1. Given query q, rank the documents in D by their query likelihood score.
2. Use the top-ranked 10–50 documents as pseudo-relevance feedback R^+.
3. Compute the relevance model probabilities.
4. Rank documents by their KL divergence score as computed above.

Queries

Relevance Feedback for Cross-Encoders [Li et al. 2023]

Concatenate and truncate. Generate new query by concatenating text from original query with text of the of the top-k feedback documents, separating each with a space. Resulting query is truncated to fit into input length.

$$\mathrm{q}' = \lceil \mathrm{q} + \text{␣} + \mathrm{d}_1, \ldots, + \text{␣} + \mathrm{d}_k \rceil_{256}$$

Concatenate and aggregate. Generate k new queries by concatenating original query with each of the top-k passages. Re-rank all documents with new query, and aggregate scores of all the re-scored documents.

$$\mathrm{q}'_1 = \mathrm{q} + \text{␣} + \mathrm{d}_1$$
$$\ldots$$
$$\mathrm{q}'_k = \mathrm{q} + \text{␣} + \mathrm{d}_\mathrm{k}$$

Sliding window. Generate j new queries by concatenating the top-k feedback documents and slide a window to partition the text and combine with original query like in concatenate and aggregate.

Last two require aggregation. Possible aggregation mechanisms include: Average, Max, Fusion.

Queries

Relevance Feedback for Cross-Encoders [Li et al. 2023]

Observations:

- Cross-encoders are already expensive, can only do re-ranking.
- Shallow pools of pseudo-relevance feedback are better than deeper.
  - More documents lead to query drift.
- Concatenate and aggregate is generally the best method.
- When aggregation is needed, Average is generally the best, Max is worst.
- Relative gains compared to bi-encoders relevance feedback much lower.
- Combining relevance feedback for bi-encoder and cross-encoder in a pipeline improves effectiveness further.

Queries

Query/Document Expansion/Reduction

Idea: Use a neural model to add or remove terms to queries or documents to make lexical matching more effective.

Document Expansion

- doc2query [Nogueira et al. 2019]

Document Reduction

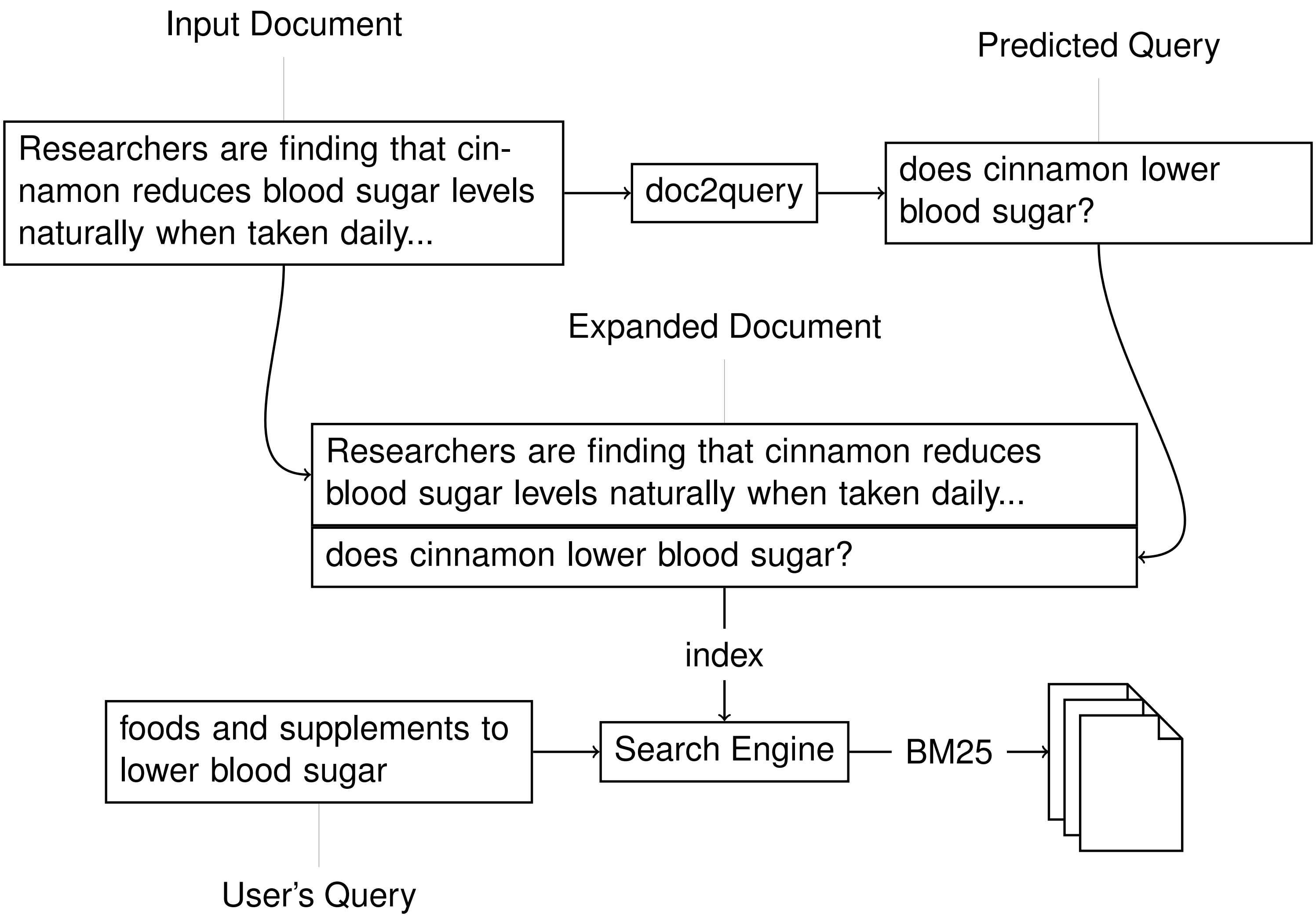
- DeepCT [Dai and Callan 2020]

Query+Document Expansion+Reduction

- SPLADE [Formal et al. 2021]

Queries

Document Expansion with doc2query [Nogueira et al. 2019]



Queries

Document Expansion with doc2query [Nogueira et al. 2019]

doc2query attempts to address the “vocabulary mismatch” problem of lexical retrieval models by using a neural model to expand the terms of a document.

- For each document, predict queries where the document would be relevant.
- Encoder-Decoder transformer model (T5) fine-tuned to do prediction.
  - Modelled as a sequence-to-sequence task, e.g., translation.
  - Translate document ➜ query.

Queries

Document Expansion with doc2query [Nogueira et al. 2019]

doc2query attempts to address the “vocabulary mismatch” problem of lexical retrieval models by using a neural model to expand the terms of a document.

- For each document, predict queries where the document would be relevant.
- Encoder-Decoder transformer model (T5) fine-tuned to do prediction.
  - Modelled as a sequence-to-sequence task, e.g., translation.
  - Translate document ➜ query.

Observations:

- Expanded documents significantly improve BM25 ranking.
  - No drops in query latency.
- When used in retrieval pipeline, also improves late-stage effectiveness.
- Downside: very expensive indexing process.
  - Encoder-decoder models are the most computationally intensive.
  - MSMARCOv1 (102k passages): from 10 minutes to 760 hours.

Queries

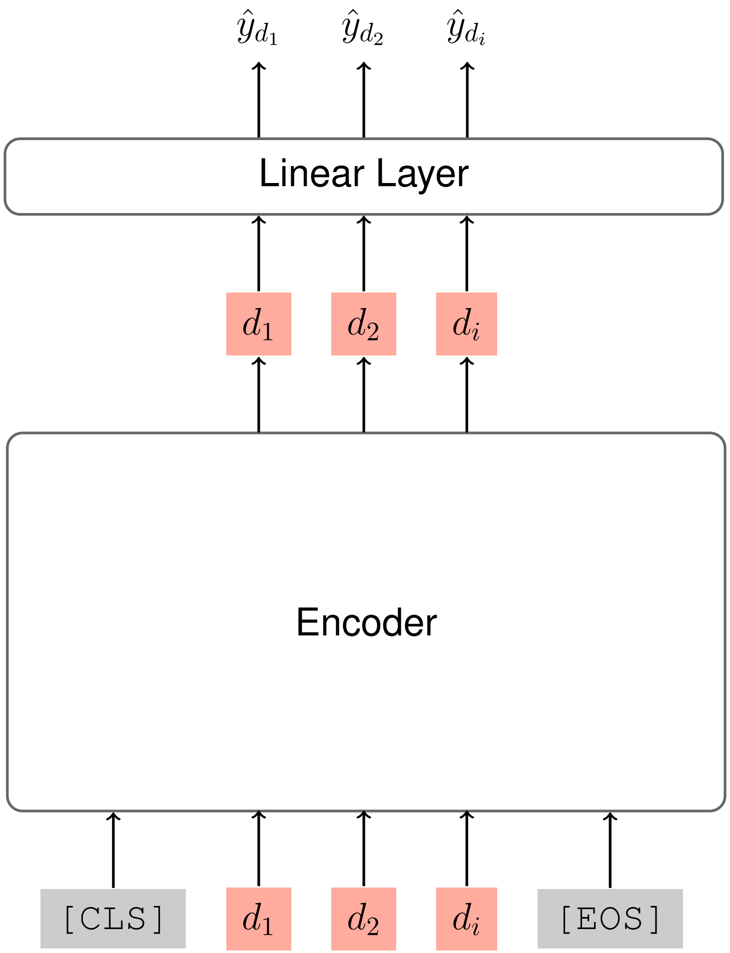
Document Expansion with doc2query [Nogueira et al. 2019]

Related methods:

- TILDEv2 [Zhuang and Zuccon 2021]
  - Uses term likelihoods for each document representation to decide as expansion terms.
  - Since expansion term prediction requires only a single forward pass through the model, indexing is much faster than doc2query, which requires a forward pass for each query token.

Queries

Document Reduction with DeepCT [Dai and Callan 2020]



Queries

Document Reduction with DeepCT [Dai and Callan 2020]

DeepCT aims to compute "deep contextualised term weights" for each term in a document. Intuition: use learnt term weights instead of relying on term frequency and score with BM25.

- Compute an importance score for each term as the linear combination of a term's contextualised embedding.
- Model is trained end-to-end using a regression task to predict a score per contextualised embedding. Minimise mean square error between predicted and ground truth weights.

$$\mathcal{L} = \sum_{d \in D} \sum_{d_i \in d} (y_{d_i} - \hat{y}_{d_i})^2$$

- How to obtain ground truth term weights? ➜ query term recall:

$$y_{d_i} = \frac{Q_{d_i}}{Q_d}$$

Q_d: set of queries where d is relevant. Q_{d_i} Subset of Q_d that contain d_i. Therefore, y_{d_i} estimated as percentage of d's queries that mention token d_i.

Queries

Document Reduction with DeepCT [Dai and Callan 2020]

Observations:

- Like doc2query, DeepCT is query independent.
  - Queries are only used to generate training labels.
  - Term weight estimations can be done offline.
  - Query latency not impacted.

- Predicted term weights need to be scaled to be used in BM25:

$$\textit{tf}_{\text{DeepCT}} = \text{round}(\hat{y}_{d_i} \cdot N)$$

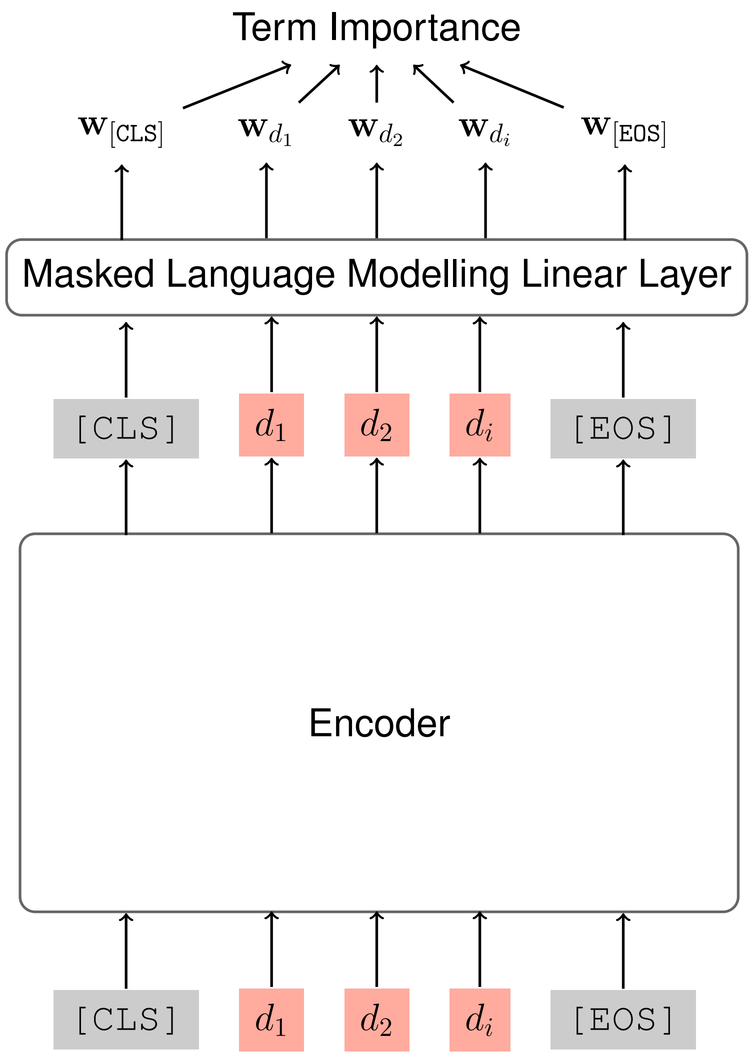
  If $N = 100$, and $\hat{y}_{d_i} < 0.005$, this term will be ignored as it has a weight of 0.

- Scaling scores to be useable in scoring functions like BM25 has the effect of document reduction, or pruning.

- The combined effect of re-weighting document terms and ignoring terms that do not contribute to relevance makes DeepCT more effective than doc2query.

Queries

Document+Query Expansion+Reduction with SPLADE [Formal et al. 2021]



Queries

Document+Query Expansion+Reduction with SPLADE [Formal et al. 2021]

SPLADE attempts to learn sparse representations of documents and queries in a straight-forward way that exploits the masked language modelling task of BERT.

- For every token in the document, the masked language model layer provides a probability distribution over all terms in the vocabulary.
- To obtain a single sparse representation, the individual term probability distributions are aggregated into a single importance estimation distribution.

$$\mathbf{w}' = \sum_{\mathbf{w}_{d_i} \in \mathbf{w}_d} \log(1 + \text{ReLU}(\mathbf{w}_{d_i}))$$

  - ReLU enforces sparsity, any value below 0 is set to 0.
  - Regularisation in loss function further enforces sparsity (non-zero entries are penalised).

- Final sparse representation may contain new semantically related terms, and may remove unimportant terms.

Queries

Document+Query Expansion+Reduction with SPLADE [Formal et al. 2021]

Observations:

- SPLADE is parametrised to control level of sparsity in queries and documents.
- Since SPLADE is sparse, and it learns which terms are important, index size can be greatly compressed.
- Compared to doc2query and DeepCT, SPLADE is both more effective, while maintaining query latency speeds.

Queries

Document+Query Expansion+Reduction with SPLADE [Formal et al. 2021]

Related Models:

- SparTerm [Bai et al. 2020]
  - Precursor to SPLADE.
- SNRM [Zamani et al. 2018]
  - One of the first methods to do learned sparse representations.

Queries
Query Performance Prediction

Idea: Measure the retrieval effectiveness of a query without ground truth labels.

Two classes of Query Performance Predictors (QPPs):

- Pre-retrieval: use signals only from the query itself.
- Post-retrieval: use signals from the query and the ranking of documents.

For both classes of QPPs, predictors can either be:

- Unsupervised: use signals to make prediction directly.
- Supervised: learn a predictor from signals to make a prediction.

Applications: [Carmel and Kurland 2012]

- Feedback to users on whether they should reformulate their query.
- Feedback to search engine on whether to reformulate query automatically.
- Feedback for engineers to identify potential difficult queries.
- For IR applications, e.g., weighted rank fusion from multiple systems.

Queries

Pre-retrieval QPPs

An incomplete list of pre-retrieval QPPs:

- Query Length [Mothe and Tanguy 2005]
- Term Length [He and Ounis 2006]
- Inverse Document Frequency [Cronen-Townsend et al. 2002]
- Inverse Collection Term Frequency [Kwok 1996]
- Query Scope [Plachouras 2003]
- Simplified Clarity Score [He and Ounis 2006]
- Collection Query Similarity [Zhao et al 2008]

In general, pre-retrieval QPPs attempt to use statistics about the terms in the query without actually issuing the query to a retrieval system.

Although it is uncommon for any one given pre-retrieval QPP to correlate with retrieval effectiveness, the combination of them may provide a better signal.

Queries

Post-retrieval QPPs

An incomplete list of post-retrieval QPPs:

- Clarity Score [Cronen-Townsend et al. 2002]
- Weighted Information Gain [Zhou and Croft 2007]
- Weighted Expansion Gain [Khwileh et al. 2007]
- Normalised Query Commitment [Shtok et al. 2009]

In general, post-retrieval QPPs attempt to use statistics about the documents retrieved by the query to predict retrieval effectiveness.

Although it is more common for post-retrieval QPPs to correlate with retrieval effectiveness, the combination of them and in combination with pre-retrieval QPPs may provide a better signal.

Queries

Post-retrieval QPPs

Example: Clarity Score

$$\text{clarity score} = \sum_{t \in V} P(t|\mathbf{q}) \log \frac{P(t|\mathbf{q})}{P(t|C)},$$

Computed as the relative entropy, or Kullback-Leibler divergence between the query and the collection language models, where

$$P(t|\mathbf{q}) = \sum_{\mathbf{d} \in R} P(t|\mathbf{d}) P(\mathbf{d}|\mathbf{q}).$$

How to estimate the probabilities is left as an exercise to the reader.

Queries

Neural QPPs

Supervised:

- BERT-QPP [Arabzadeh et al. 2021]

Unsupervised:

- Dense-QPP [Arabzadeh et al. 2023]

Queries

QPP using BERT-QPP [Arabzadeh et al. 2021]

Fine-tune cross-encoder to optimise an evaluation measure with cross-entropy loss:

$$\begin{aligned}\mathcal{L} =& M(q,d) \cdot QPP(q,d) \quad + \\ & (1 - M(q,d)) \cdot 1 - QPP(q,d),\end{aligned}$$

Where M is the evaluation measure to optimise, and QPP is the model's prediction.

Alternatively, fine-tune bi-encoder using euclidean, or L2 loss:

$$\mathcal{L} = ||M(q,d) - QPP(q,d)||_2,$$

Where, $QPP = sim(q,d)$.

Both instantiations attempt to fine-tune an encoder model to predict a score given a query and a document.

Queries

QPP using BERT-QPP [Arabzadeh et al. 2021]

Observations:

- Can adapt the regression task of optimising an evaluation measure to a cross-encoder or bi-encoder.
- Cross-encoder is better at predicting the retrieval effectiveness of a query than bi-encoder model.
- Compared to other transformer-based QPP measures, BERT-QPP is much faster to compute. See:
  - NQA-QPP [Hashemi et al. 2019]
  - NeuralQPP [Zamani et al. 2019]
- Also: non-transformer-based supervised QPP method [Datta et al. 2022]

Queries

QPP using Dense-QPP [Arabzadeh et al. 2022]

Create a perturbed query by applying additive Gaussian white noise:

$$\mathbf{q}' = \mathbf{q} + \mathcal{G}(\mathbf{q}, \mu, \sigma^2)$$

- Characteristics of noise controlled by μ (mean) and σ^2 (variance).
- Noise should not lean query representation in any particular direction.
  - Therefore, $\mu = 0$
- In theory, σ^2 can be calculated with signal-to-noise ratio.
  - In practice, a small value, e.g., 0.01 suffices.

Based on the idea of query “robustness”: the effectiveness of a query should change little if perturbations are applied to it.

- Inject noise into dense representation of queries from a bi-encoder.
- A less robust query would experience a noticeable change in retrieval.
- Final QPP metric is calculated as the rank similarity between documents ranked by the original query and the perturbed one.

Queries

QPP using Dense-QPP [Arabzadeh et al. 2022]

Observations:

- Correlation between Dense-QPP and retrieval effectiveness is much higher than all other QPP methods.
- Method is parameterless, amount of noise added to a query is the same for all queries.
- Method is still reasonably efficient to compute, since it uses a bi-encoder model to score documents and the query only needs to be encoded once.

Queries

Questions

1. What do you think has a bigger impact on effectiveness, positive or negative relevance feedback? Do you think this answer changes for classic (e.g., BM25) versus neural (e.g., monoBERT) retrieval models?

2. SPLADE works to expand and reduce queries and documents with sparse representations. How do you think this approach works with short versus long documents?

3. Query performance predictors are measuring something about how good a query is at retrieving documents. How are they different from the evaluation measures we've discussed in this course?