# Pre-indexing pruning strategies

## SPIRE 2020

## October 14 2020

**Soner Altin - Universitat Pompeu Fabra**
**Ricardo Baeza-Yates - Northeastern University, Universitat Pompeu Fabra, Universidad de Chile**
**Barla Cambazoglu - RMIT University**

# Agenda

- Motivation
- Pruning Heuristics
- Experimental Setup
- Experimental Results
- Conclusions

# Motivation

## Index pruning & post-indexing

- Index pruning is creating a so-called pruned inverted index which stores less information than the full web index while attaining the search result quality obtained with a full index as much as possible. A pruned index has lower space requirements and leads to faster query processing since fewer postings are stored and processed. The main challenge is to prevent degradation in search quality and query coverage due to the absence of potentially useful indexed content in the pruned index.

- All existing approaches so far assume the presence of a full web index to facilitate the construction of the pruned index. That is, a pruned index is created by removing postings from the inverted lists in the full web index, entirely or selectively, by using statistical techniques, with the objective of maintaining the search quality.

# Motivation

## Pre-index pruning

- In certain scenarios, however, the resource constraints are really tight (e.g., a low-budget web search engine), and it may not be feasible to build and/or store the full web index. Therefore, pruning decisions need to be made without constructing a full web index first, perhaps even at crawling time to reduce the storage requirements. Hence, how much we lose if we decide a priori which documents or parts of them should be indexed?

- Motivated by these scenarios and that few documents may cover most queries, here we design document pruning strategies which do not require the presence of a full web index.

.

# Pruning Heuristics

## Pruning approaches

- Document level pruning
  - Document size
  - Set cover
  - aDCP
- Sentence level pruning
  - Summarization
- Term level pruning
  - Query terms
  - Stop word
  - Stemming

# Pruning Heuristics

## Document-Level Pruning

- **Document size**: Very large documents may take too much space in the index. Yet, their contribution to search quality is often not likely to be very different from medium-size documents. Based on this idea, we exclude largest L% of documents in the collection from the index and we index the remaining documents.

- **Set cover:** Essentially, this heuristic tries to select a minimal subset D of documents from a given collection such that the number of queries whose ideal top k results (obtained using a full web index) contain at least one document from D is maximized.

- **Access-based Document-Centric Pruning**: Altingovde et al. proposed an access based pruning strategy to prune documents directly from the collection. Documents with low access count are removed from the collection until a fraction of pruned documents is reached.

# Pruning Heuristics

## Sentence-Level Pruning

- **Summarization:** Certain sentences in a document are relatively more important or represent the document better than the others. It may be more beneficial to index such sentences as they are more likely to be of interest to users and match their queries. To this end, we apply the Textteaser summarizer to the content of documents to obtain the most important or representative S sentences and then index only the terms occurring in those sentences. If the document contains less than S sentences, we do not apply summarization, i.e., we index all of the terms in the document.

# Pruning Heuristics

## Term-Level Pruning

- **Query terms popularity:** It is usually important to index terms that often appear in web queries. In this heuristic, we extract the set of terms occurring in a web query log to obtain a representative set of such useful terms. When processing documents, we index only the terms which appear in this set.

- **Stopword removal and stemming:** We also considered these standard term pro- cessing operations as baseline cases for term-level pruning. We have three cases, just stopwords, just stemming, or use both of them.

# Pruning Heuristics

## Properties of the heuristics

**Table 1.** Properties of the heuristics.

| Index | Uses training queries | Uses training documents | Index | Uses training queries | Uses training documents |
|---|---|---|---|---|---|
| `Full` | No | No | `TermPopularity` | Yes | No |
| `DocumentSize` | No | No | `Stemming` | No | No |
| `SetCover` | Yes | Yes | `Stopwords` | No | No |
| `aDCP` | Yes | Yes | `StemStop` | No | No |
| `Summarization` | No | No | | | |

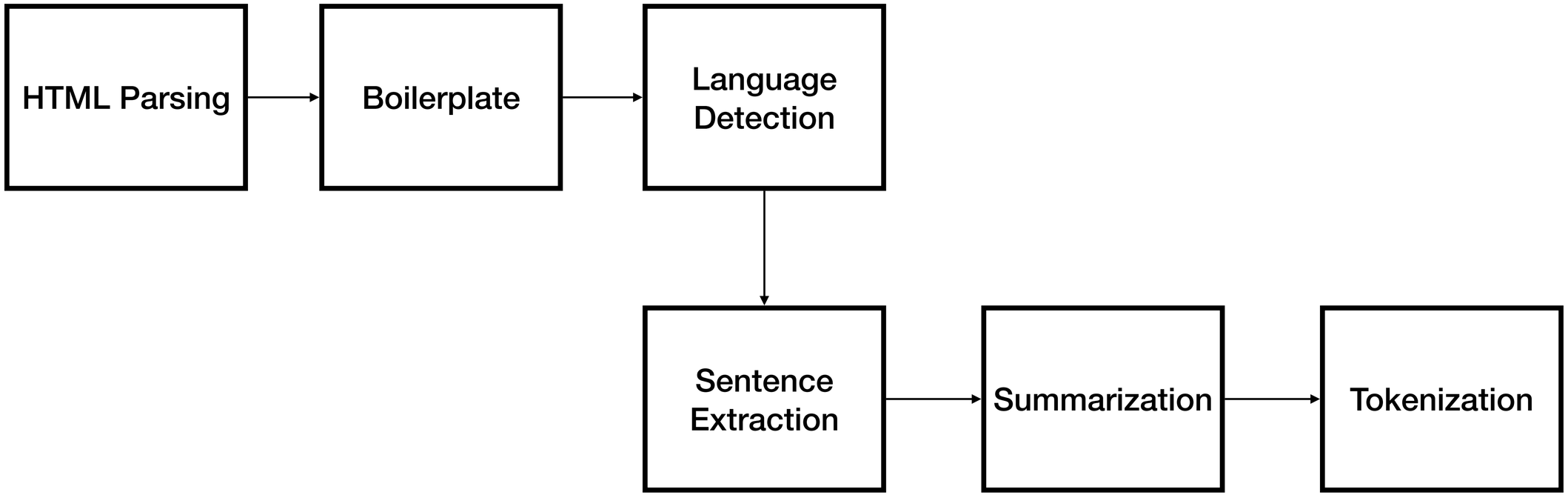
# Experimental Setup

## Document Collection

- **Web document collection**: Common Crawl, November 2017. 4TB of disk space when compressed. The full web index constructed using this collection contains 50.3M English documents. The average document size is 3.5KB before parsing the HTML content, and after processing the average number of sentences and terms in a document is around 26 and 628, respectively. This collection is quite diverse in that it contains pages crawled from more than 1.3M different web domains.

# Experimental Setup

## Document Processing



# Experimental Setup

## Indexing

- We index documents using an open source version of Elasticsearch. The documents are indexed in an incremental fashion. All indexes were created with 15 shards with a replication factor of 3. Full is the full index constructed using the entire collection for comparison purposes. Most of the evaluation metrics computed are relative to this index.

- We used Elasticsearch's built-in stopword remover and its built-in stemmer, which removes only possessive suffixes, to obtain our three baselines: Stemming, Stopwords, and StemStop). In all other cases, we do not perform stopword removal nor stemming.

# Experimental Setup

## Indexing

- In total we created 18 different indices
- For the DocumentSize heuristic, we set L to 1% and 10%. These thresholds result in pruning of documents that are larger than 7KB and 33KB, respectively.
- In the SetCover heuristic, we set the C parameter to 1, 5, or 10, resulting in approximately 3.1, 1.7 and 1.3 million documents being indexed, respectively.
- In the Summarization heuristic, we set the S parameter to 10, 20, 40, 80, and 160.
- For the aDCP heuristic, we set the μ parameter to 0.1, 0.2 and 0.3 resulting in 5, 10 and 15 million documents being indexed, respectively.
- In the case of the Term Popularity index, we prune all terms which do not appear in a given query log.

# Experimental Setup

## Pruning Performance

**Table 2.** Index sizes and corresponding pruning ratios ($PR$) shown as percentages.

| Index | | Parameter Size (GB) | $PR$ (%) |
|---|---|---|---|
| `Full` | – | 180.200 | 0.00 |
| `DocumentSize` (DS#) | $L = 1\%$ | 54.147 | 69.95 |
| `DocumentSize` | $L = 10\%$ | 32.226 | 82.12 |
| `SetCover` (SC#) | $C = 1$ | 9.400 | 94.78 |
| `SetCover` | $C = 5$ | 7.000 | 96.12 |
| `SetCover` | $C = 10$ | 6.600 | **96.34** |
| `aDCP` (aDECP#) | $\mu = 0.1$ | 22.900 | 87.29 |
| `aDCP` | $\mu = 0.2$ | 31.600 | 81.90 |
| `aDCP` | $\mu = 0.3$ | 41.100 | 77.19 |
| `Summarization` (S#) | $S = 10$ | 23.725 | 86.83 |
| `Summarization` | $S = 20$ | 32.930 | 81.73 |
| `Summarization` | $S = 40$ | 42.526 | 76.40 |
| `Summarization` | $S = 80$ | 50.713 | 71.86 |
| `Summarization` | $S = 160$ | 82.58 | **54.17** |
| `TermPopularity` (TP) | – | 52.014 | 71.14 |
| `Stemming` (ST) | – | 69.791 | 61.27 |
| `Stopwords` (SW) | – | 64.338 | 64.30 |
| `StemStop` (BOTH) | – | 60.907 | 66.20 |

# Experimental Setup

## Ranking

- We use two different ranking techniques in the experimental comparison. The simplest one just uses the well-known BM25 technique which is native to ElasticSearch, as it is one of the best baselines based just in textual content. A more sophisticated version uses a two phase ranking approach, first using BM25 to obtain a pool of 2,000 candidates and then using learning- to-rank (LTR), LambdaMart, to do the final ranking. The LTR variant uses more than 200 features that include query-document similarity (44%), link analysis (20%), query-document relevance (16%), URL name features (10%) and textual content (10%) features.

# Experimental Setup

## Query Logs

- **Query log A**: Which contains 7.3 million queries submitted in 2006. The query log is split, in temporal order, into training and test sets, which contain 6.4 million and 900K queries, respectively. The training set is used to compute the set cover as well as terms' query frequencies. The test queries are used for evaluation.

- **Query log B**: Contains about 300K queries with relevance judgments for the top 50 results of our LTR model, where 125K queries are used for training the model and the 175K others are left for relevance evaluation.

- **Query log C**: Contains almost 2.7M queries and they are used to study how sensitive are our pruning techniques to a query log from a different search engine and from a different time (notice that the year of this query log matches the year of the web collection).

# Experimental Setup

## Evaluation Measures

- Pruning ratio
- Average precision
- Average recall
- Average result similarity
- Unique query coverage
- Query volume coverage
- Normalized discounted cumulative gain

# Experimental Setup

## Common Crawl and BM25

- Precision, recall, and result similarity, as expected, less aggressive pruning strategies tend to yield the highest precision values. For example, with query log A, the highest precision value (90%) is obtained using the Summarization strategy with S = 160, while the lowest value (35%) is obtained using the SetCover strategy with C = 10. In terms of the recall and result similarity metrics, we observe very similar values. The metrics obtained by using the query log C (Table 3) confirm the validity of the results since they exhibit similar behavior.

# Experimental Results

## Common Crawl and BM25

spire VIRTUAL CONFERENCE 2020

**Table 3.** Evaluation metrics in percentages (CC collection, BM25, query log A).

| Index | Parameter | $PR$ | $AP@k$ | $AR@k$ | $ARS@k$ | $QC@k$ | $QVC@k$ |
|---|---|---|---|---|---|---|---|
| `Full` | | | 100 | 100 | 100 | 82.91 | 96.54 |
| `DocumentSize` | $L = 1\%$ | 69.95 | 78.91 | 94.59 | 75.21 | 78.56 | 95.66 |
| `DocumentSize` | $L = 10\%$ | 82.12 | 58.84 | 87.27 | 51.76 | 72.54 | 94.19 |
| `SetCover` | $C = 1$ | 94.78 | 41.95 | 51.99 | 34.07 | 82.80 | 96.53 |
| `SetCover` | $C = 5$ | 96.12 | 36.27 | 47.46 | 29.63 | 82.79 | 96.52 |
| `SetCover` | $C = 10$ | **96.34** | 35.34 | 46.62 | 28.99 | 82.78 | 96.52 |
| `aDCP` | $\mu = 0.1$ | 87.29 | 51.92 | 53.19 | 40.53 | 82.15 | 96.15 |
| `aDCP` | $\mu = 0.2$ | 81.90 | 63.32 | 53.86 | 51.52 | 82.52 | 96.33 |
| `aDCP` | $\mu = 0.3$ | 77.19 | 64.97 | 64.09 | 52.28 | 82.69 | 96.43 |
| `Summarization` | $S = 10$ | 86.83 | 39.99 | 65.94 | 31.31 | 70.90 | 93.57 |
| `Summarization` | $S = 20$ | 81.73 | 52.01 | 75.19 | 43.14 | 73.81 | 94.46 |
| `Summarization` | $S = 40$ | 76.40 | 64.27 | 83.91 | 56.64 | 76.21 | 95.11 |
| `Summarization` | $S = 80$ | 71.86 | 73.83 | 90.22 | 68.22 | 78.08 | 95.54 |
| `Summarization` | $S = 160$ | 54.17 | **89.75** | **97.21** | **87.90** | **82.91** | **96.54** |
| `TermPopularity` | | 71.14 | 71.57 | 79.85 | 61.39 | 79.69 | 95.60 |
| `Stemming` | | 61.27 | 66.13 | 58.42 | 49.94 | **85.92** | **97.13** |
| `Stopwords` | | 64.30 | **93.70** | **93.64** | **88.86** | 82.91 | 96.55 |
| `StemStop` | | 66.20 | 64.47 | 56.73 | 47.64 | **85.92** | **97.13** |

**Table 4.** Evaluation metrics in percentages (CC collection, BM25, query log C).

| Index | Parameter | $PR$ | $AP@k$ | $AR@k$ | $ARS@k$ | $QC@k$ | $QVC@k$ |
|---|---|---|---|---|---|---|---|
| `Full` | | | 100 | 100 | 100 | 73.43 | 78.09 |
| `DocumentSize` | $L = 1\%$ | 69.95 | 75.16 | 81.36 | 69.03 | 67.39 | 75.96 |
| `DocumentSize` | $L = 10\%$ | 82.12 | 56.05 | 66.86 | 45.99 | 59.81 | 72.89 |
| `SetCover` | $C = 1$ | 94.78 | 44.39 | 47.82 | 34.08 | 72.15 | 77.41 |
| `SetCover` | $C = 5$ | 96.12 | 40.02 | 43.94 | 30.55 | 71.97 | 77.27 |
| `SetCover` | $C = 10$ | **96.34** | 39.33 | 43.31 | 30.10 | 71.95 | 77.24 |
| `aDCP` | $\mu = 0.1$ | 87.29 | 54.53 | 55.51 | 43.65 | 72.92 | 77.79 |
| `aDCP` | $\mu = 0.2$ | 81.90 | 63.97 | 64.37 | 52.73 | 73.15 | 77.92 |
| `aDCP` | $\mu = 0.3$ | 77.19 | 64.69 | 64.78 | 53.59 | 73.27 | 77.99 |
| `Summarization` | $S = 10$ | 86.83 | 41.61 | 51.01 | 30.91 | 59.37 | 72.58 |
| `Summarization` | $S = 20$ | 81.73 | 51.48 | 59.92 | 40.55 | 62.16 | 73.73 |
| `Summarization` | $S = 40$ | 76.40 | 61.87 | 69.33 | 52.01 | 64.74 | 74.75 |
| `Summarization` | $S = 80$ | 71.86 | 70.44 | 76.98 | 62.53 | 67.01 | 75.66 |
| `Summarization` | $S = 160$ | 54.17 | **84.75** | **84.39** | **78.82** | **73.43** | **78.09** |
| `TermPopularity` | | 71.14 | 69.18 | 73.12 | 58.17 | 69.21 | 75.80 |
| `Stemming` | | 61.27 | 63.97 | 60.49 | 50.91 | 77.14 | **79.45** |
| `Stopwords` | | 64.30 | **90.56** | **90.51** | **84.65** | 73.44 | 78.09 |
| `StemStop` | | 66.20 | 61.55 | 58.04 | 47.79 | **77.15** | **79.45** |

# Experimental Results

## Common Crawl and BM25

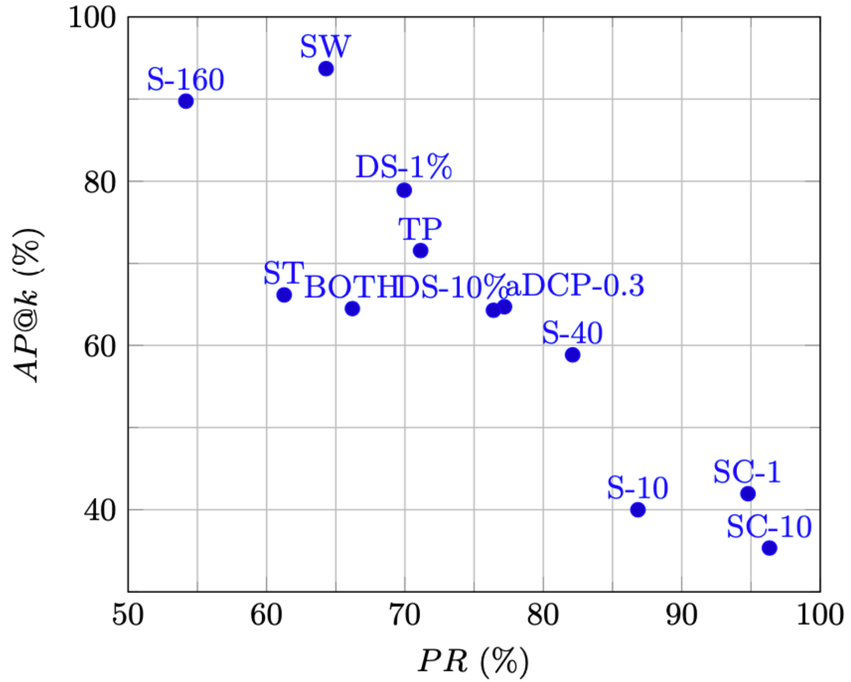



**Fig. 4.** Pruning ratio vs. precision (CC collection, BM25, query log A).

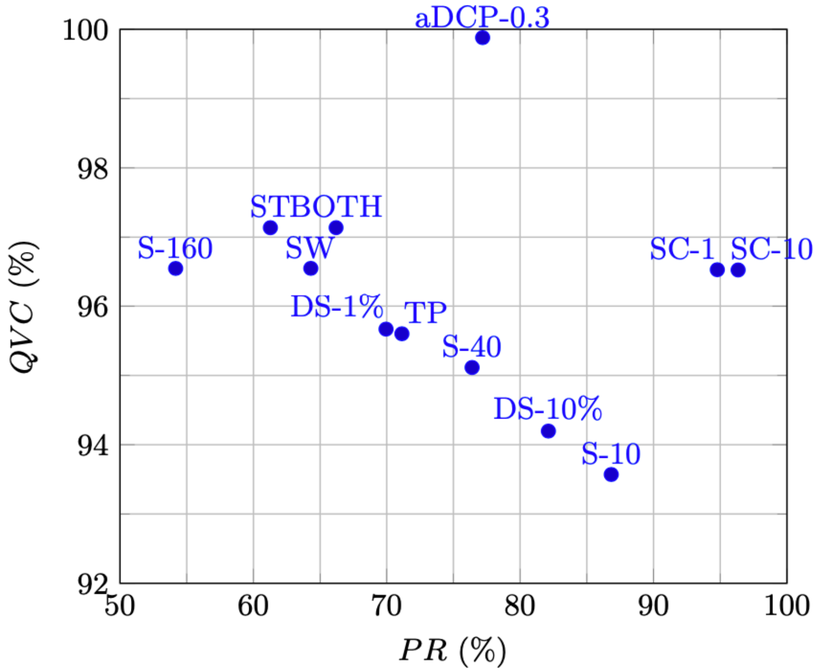



**Fig. 5.** Pruning ratio vs. query volume coverage (CC collection, BM25, query log A).

# Experimental Results

## Relevance

- To understand the relevance loss due to the pruned indexes, here we use the web collection A with our LTR ranking variant and the test queries from query log B. Based on the results of the previous section, we analyze only the heuristics with higher precision score. For example, DocumentSize with L = 1%.

- We make an exception in the case of the Summarization heuristics, to understand better its effect in relevance. In total we try seven cases and we evaluate relevance using NDCG with respect to the full index version. We do not include stemming nor stopwords removal in this case, because the LTR variant has these functionality embedded during the feature extraction process.

# Experimental Results

## Relevance

**Table 5.** Relevance results (collection A, LTR, query log C).

| Index | Parameter | Size (GB) | $PR$ (%) | $NDCG$@10 (%) | $\Delta NDCG$@10 (%) |
|---|---|---|---|---|---|
| `Full` | | 172.1 | 0.00 | 100 | 0 |
| `Summarization` | $S = 40$ | 133.9 | 22.19 | 98.97 | -1.02 |
| `Summarization` | $S = 20$ | 119.6 | 30.50 | 98.63 | -1.36 |
| `Summarization` | $S = 10$ | 107.5 | 37.53 | 98.46 | -1.53 |
| `aDCP` | $\mu = 0.3$ | 47.2 | 72.57 | 79.59 | -20.41 |
| `DocumentSize` | $L = 1\%$ | 145.6 | 15.39 | 69.98 | -30.02 |
| `TermPopularity` | | 95.4 | 44.56 | 30.21 | -69.79 |
| `SetCover` | $C = 1$ | 1.3 | 99.24 | 11.54 | -88.46 |

# Experimental Results

## Relevance ve Pruning Ratio

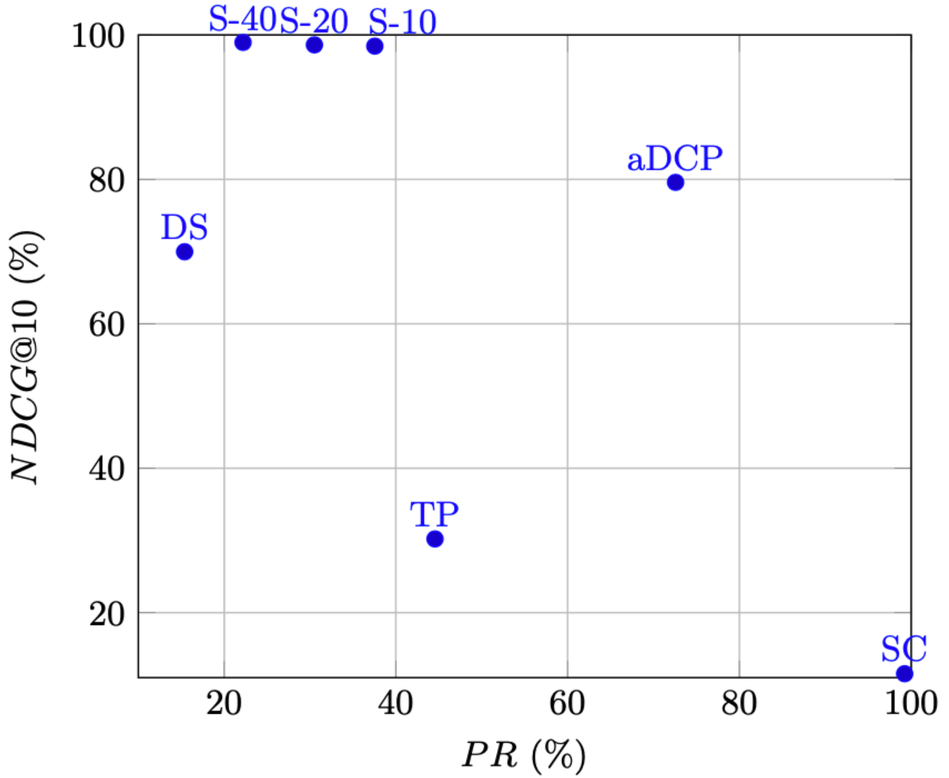



**Fig. 1.** Pruning ratio versus NDCG@10 (collection A, LTR, query log B).

# Discussion

## Trade-off Analysis

- In general, as the pruning ratio increases the precision is observed to decrease, as expected. The Summarization strategy (with S = 160) can be seen to cut the index size by half with around only 10% decrease in precision with respect to the full web index. Therefore, search engines, for which quality is vital, can use this technique to achieve drastic reduction in the hardware needed for storing and processing the web index. The SetCover strategy, on the other hand, results in significant quality loss (around 60%), but can yield huge resource savings (around 95%). Therefore, commercial search engines that operate with very limited resources can employ this strategy to cut their operational costs.

# Discussion

## Trade-off Analysis

- There is a similar trade-off between the pruning ratio and query volume coverage. The Summarization strategy (with S = 160) results in high query volume coverage, again, under the same pruning ratio. The SetCover strategy, however, achieves almost the same query volume coverage (around 96.5%), with a drastic reduction in index size. Therefore, a search engine, which aims to satisfy as many queries as possible (rather than the aggregate performance over many queries) may adopt SetCover as its pruning strategy.

- Regarding relevance, the summarization technique allows to keep parts of all the documents, leading to a marginal NDCG@10 drop, with a reasonable index size reduction. So clearly they are very competitive. The differences that appear in the pruning ratios for some heuristics, such as Summarization and Document size, can be explained by the fact that collection A is of better quality (that is, has much less web spam as has been curated) and also is more homogeneous and hence the fraction of documents removed diminishes.

# Conclusions



- We have shown different index pruning strategies that aim to reduce the index size significantly, while keeping the search quality and query coverage as similar as possible to the case of the full web index. We conducted large-scale experiments demonstrating the feasibility of these approaches. Our results show that there is no clear dominant technique for different query sets and surely the same is true for different web document collections. We have not included the gains in query processing time, but in all the techniques we get significant faster processing times, from 50% more throughput for S160 to almost a 10 times improvement for the most aggressive pruning techniques.

- In practice, each search application has different use cases and different hardware resources. So, we believe that constructing an index that will satisfy all kinds of use cases is not feasible in practice. However, we can recommend different index pruning techniques for different use cases. For example, if we do not have large amounts of hardware and the quality of results is not critical, SetCover or Summarization with a small S parameter (e.g., S = 10) can be a good fit. If we have lots of hardware and search quality is vital, then the Summarization strategy with a large S parameter (e.g., S = 80) or the DocumentSize strategy with a small L parameter (e.g., L = 1%) might be good options.

# Thanks for listening