Online Testing Learning to Rank with Solr Interleaving

Alessandro Benedetti, Director

24th March 2022

Who We Are

HAYSTACK

Alessandro Benedetti

- Born in Tarquinia(ancient Etruscan city in Italy)
- R&D Software Engineer
- Director
- Master in Computer Science
- PC member for ECIR, SIGIR and Desires
- Apache Lucene/Solr PMC member/committer
- Elasticsearch expert
- Semantic, NLP, Machine Learning technologies passionate
- Beach Volleyball player and Snowboarder

Sease

SEArch SErvices

HAYSTACK

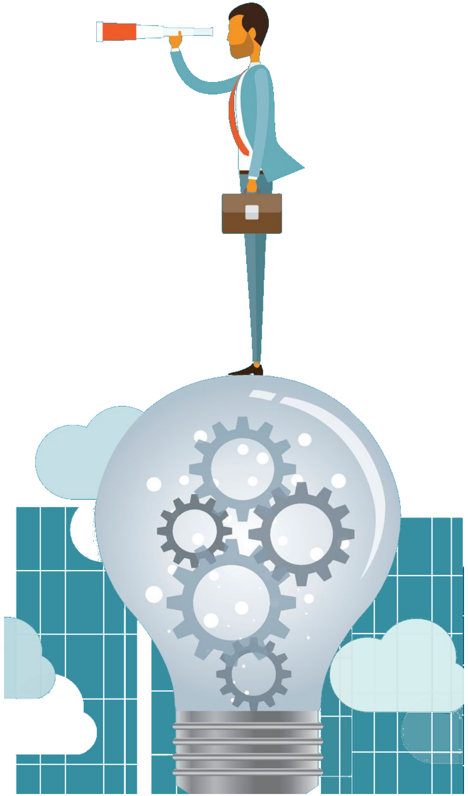
www.sease.io

- Headquarter in London/distributed
- Open Source Enthusiasts
- Apache Lucene/Solr experts
- Elasticsearch experts
- Community Contributors
- Active Researchers
- **Hot Trends** : Neural Search,
  Natural Language Processing
  Learning To Rank,
  Document Similarity,
  Search Quality Evaluation,
  Relevancy Tuning

Sease
INFORMATION RETRIEVAL APPLIED

Overview

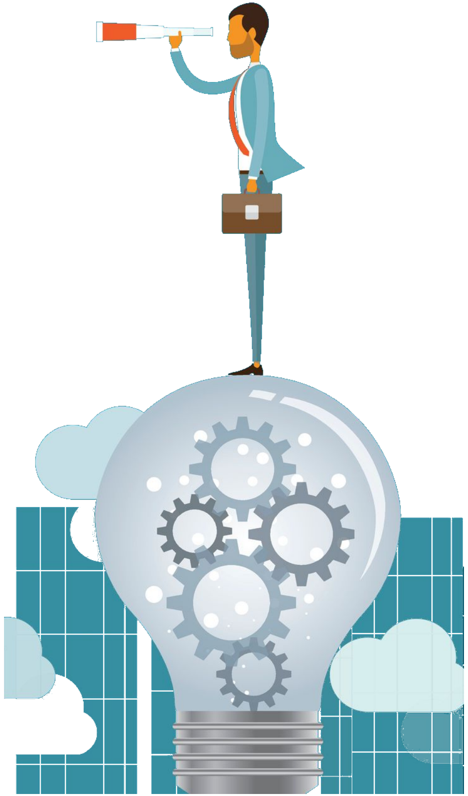
- Learning to Rank
- Online Testing for Business
- Interleaving
- Apache Solr Implementation
- DEMO



Overview

HAYSTACK

- Learning to Rank
- Online Testing for Business
- Interleaving
- Apache Solr Implementation
- DEMO



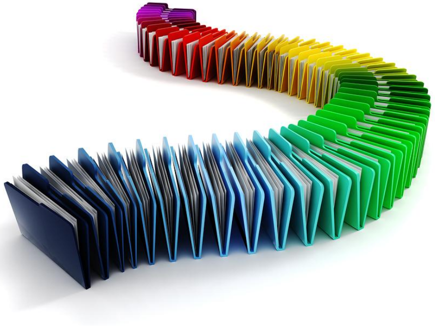
Sease

What is it?

Learning from user implicit/explicit feedback

To

Rank documents (sensu lato)



These types of models focus more on the relative ordering of items rather than the individual label (classification) or score (regression), and are categorized as **Learning To Rank** models.

What is not

- ***[sci-fi]*** Sentient system that learn by itself

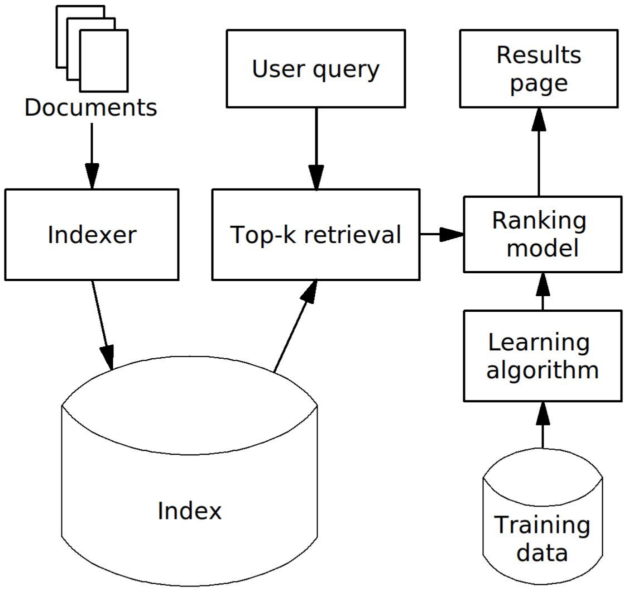
  "Machine Learning stands for that, doesn't it?" Unknown

- ***[Integration]*** Easy to set up and tune it -> it takes patience, time and multiple experiments

- *[**Explainability**]* Easy to give a human understandable explanation of why the model operates in certain ways

Application

- Ranking is a central part of many information retrieval problems, such as document retrieval, collaborative filtering, sentiment analysis, and online advertising.

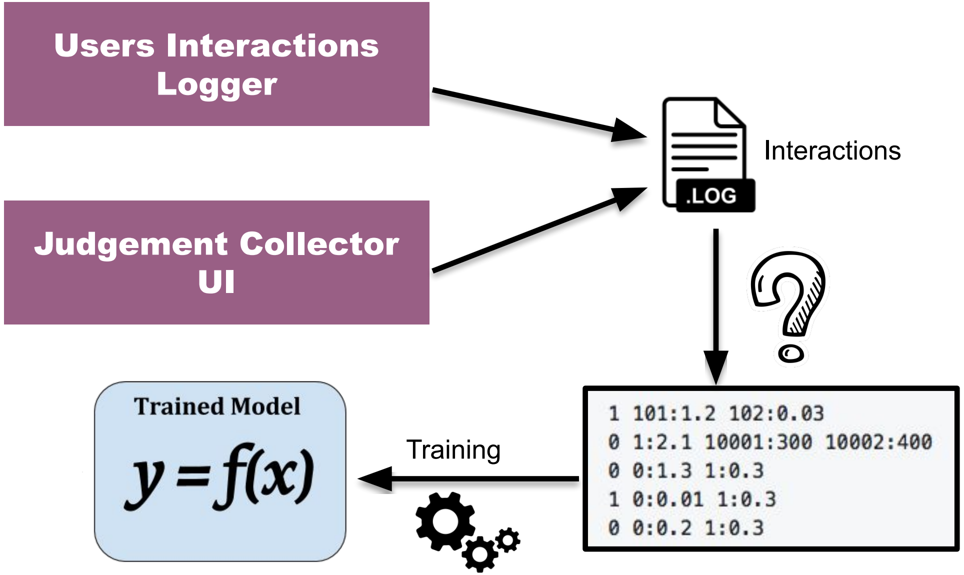


Training Data

HAYSTACK

Learning To Rank

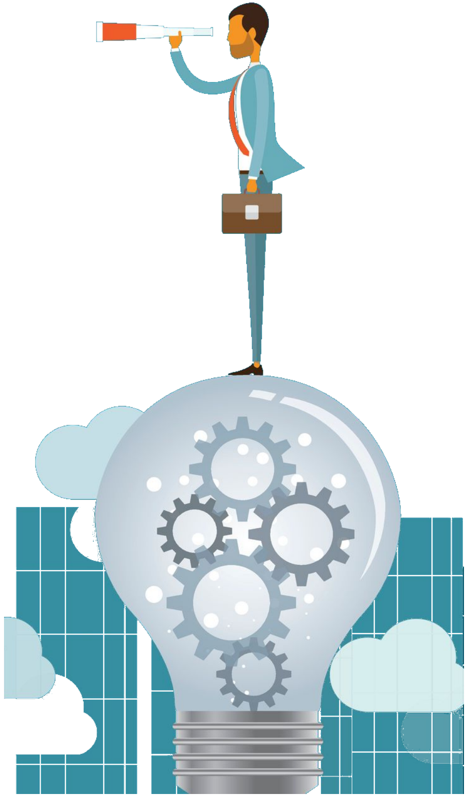
"Learning to rank is the application of machine learning, typically supervised, semi-supervised or reinforcement learning, in the construction of ranking models for information retrieval systems." Wikipedia



Sease

Overview

- Learning to Rank
- Online Testing for Business
- Interleaving
- Apache Solr Implementation
- DEMO



There are several problems that are **hard to be detected** with an *offline evaluation:*

- An **incorrect** or **imperfect test set** brings us model evaluation results that aren’t reflecting the real model improvement/regressions
  e.g.
  We may get an extremely high evaluation metric offline, but only because we improperly designed the test, even if the model is unfortunately not a good fit.

[Online] A Business Perspective

There are several problems that are **hard to be detected** with an *offline evaluation:*

- An incorrect **test set** allow us to obtain model evaluation results that aren’t reflecting the real model improvement.
  - **One sample** per query group
  - **One relevance label** for all the samples of a query group
  - **Interactions** considered for the data set creation

[Online] A Business Perspective

There are several problems that are **hard to be detected** with an *offline evaluation:*

- A **incorrect test set** allow us to obtain model evaluation results that aren’t reflecting the real model improvement.
- **Finding a direct correlation** between the offline evaluation metrics and the parameters used for the online model performance evaluation (e.g. revenues, click through rate…).

[Online] A Business Perspective

There are several problems that are **hard to be detected** with an *offline evaluation:*

- A **incorrect test set** allow us to obtain model evaluation results that aren’t reflecting the real model improvement.
- **Finding a direct correlation** between the offline evaluation metrics and the parameters used for the online model performance evaluation (e.g. revenues).
- Is based on **generated relevance labels** that not always reflect the real user need.

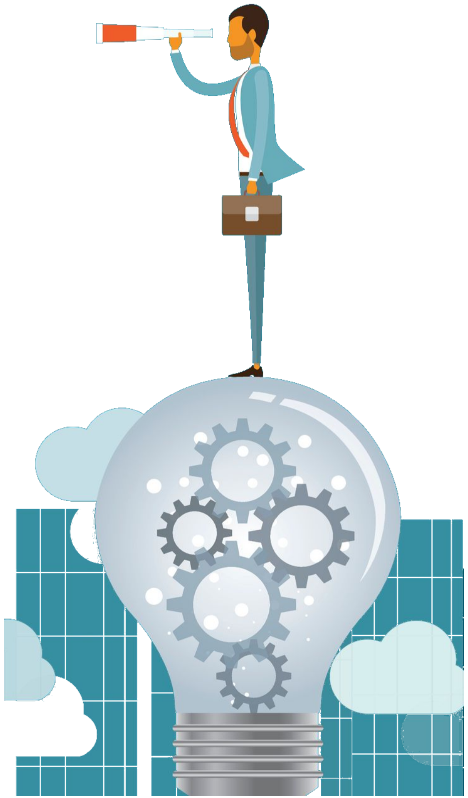
[Online] Business Advantages

Using *online testing* can lead to many **advantages**:

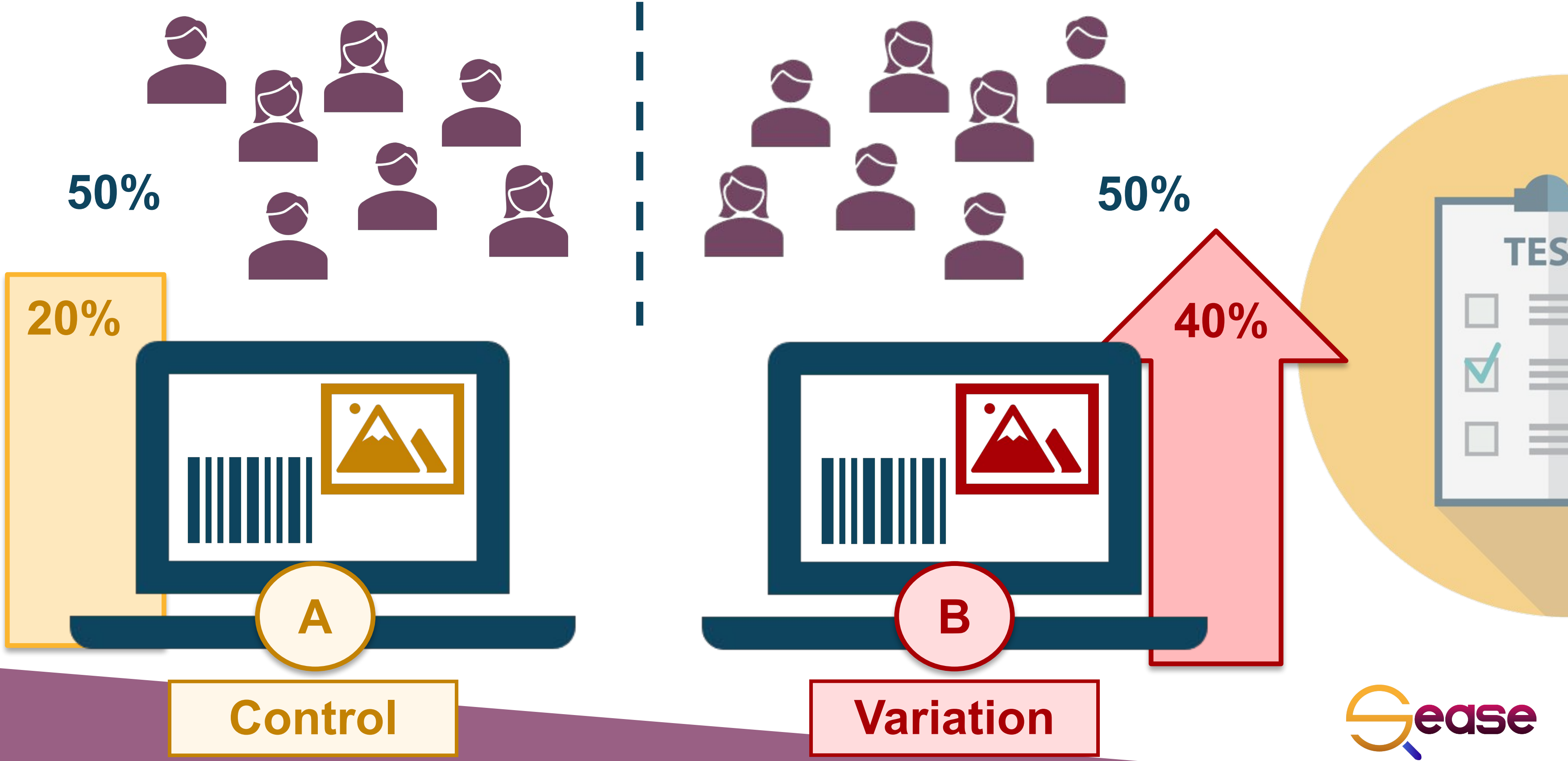
- The **reliability** of the results: we directly observe the user behaviour.
- The **interpretability** of the results: we directly observe the impact of the model in terms of online metrics the business cares.
- The possibility to observe the **model behavior**: we can see how the user interact with the model and figure out how to improve it.

Overview

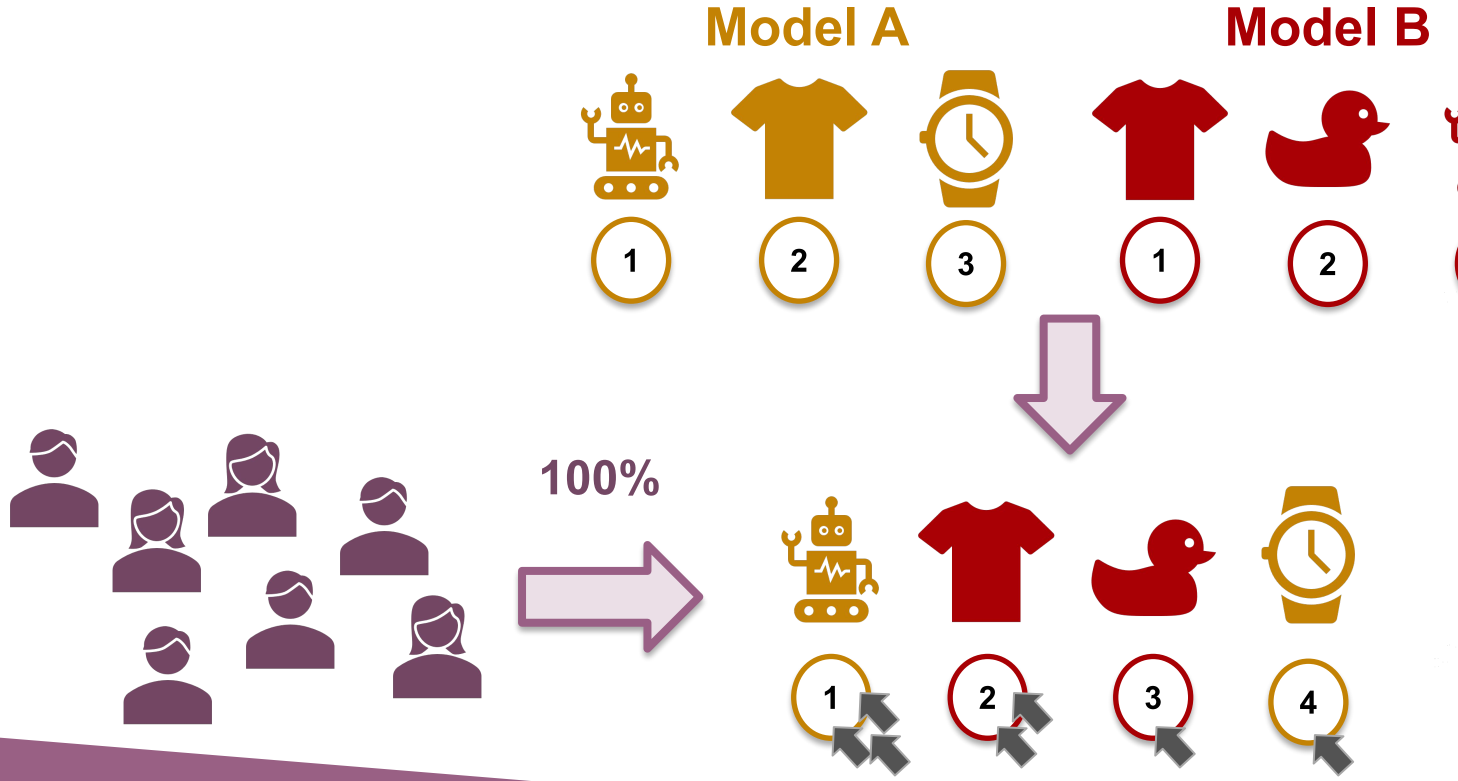
- Learning to Rank
- Online Testing for Business
- Interleaving
- Apache Solr Implementation
- DEMO



[Online] A/B testing



Interleaving



[Online] Interleaving results estimator

$$\Delta_{AB} = \frac{wins(A) + \frac{1}{2}ties(A,B)}{wins(A) + wins(B) + ties(A,B)} - 0{,}5$$

>0 winner A
<0 winner B
=0 tie

Interleaving Advantages

- It **reduces** the problem with **users' variance** due to their separation in groups (group A and group B).
- It is more **sensitive** when comparing models.
- It requires **less traffic**.
- It requires **less time** to achieve reliable results.
- It **doesn't** necessarily **expose** a **bad model** to a sub population of users.

Balanced Interleaving

There are different **types of interleaving**:

- **Balanced Interleaving:** alternate insertion with one model having the priority(decided at the beginning of the interleaving().

Input: Rankings $l_a = (a_1, a_2, \dots)$ and $l_b = (b_1, b_2, \dots)$

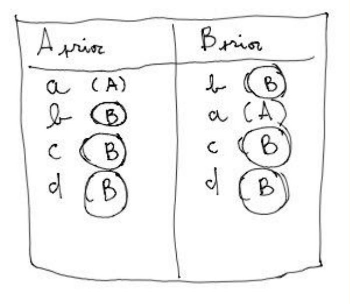
$l_I \leftarrow []$; $k_a \leftarrow 1$; $k_b \leftarrow 1$;
$Afirst \leftarrow randomBit()$
while $(k_a \leq |l_a|) \wedge (k_b \leq |l_b|)$ do
    if $(k_a < k_b) \vee \big((k_a = k_b) \wedge (Afirst = 1)\big)$ do
        if $l_a[k_a] \notin l_I$ then $l_I \leftarrow l_I + l_a[k_a]$
        $k_a \leftarrow k_a + 1$
    else
        if $l_b[k_b] \notin l_I$ then $l_I \leftarrow l_I + l_b[k_b]$
        $k_b \leftarrow k_b + 1$
    end if
end while

Output: Interleaving ranking l_I

HAYSTACK

Ther

DRAWBACK



- When comparing two very similar models.
  - **Model A:** l_A = (*a*, *b*, c, d)
  - **Model B:** l_B = (*b*, *c*, d, a)
- The comparison phase will bring the **Model B** to win more often than **Model A**. This happens regardless of the model chosen as prior.
- This drawback arises due to:
  - the way in which the evaluation of the results is done.
  - the fact that *model_B* rank higher than *model_A* all documents with the exception of *a*.

sease

Team-Draft Interleaving

There are different **types of interleaving**:

- **Balanced Interleaving:** alternate insertion with one model having the priority.
- **Team-Draft Interleaving:** method of team captains in team-matches.

Team-Draft Interleaving

HAYSTACK

Input: Rankings $l_a = (a_1, a_2, \dots)$ and $l_b = (b_1, b_2, \dots)$

1. $l_I \leftarrow []$; $TeamA \leftarrow \emptyset$; $TeamB \leftarrow \emptyset$;
2. while $(\exists i: l_a[i] \notin l_I) \wedge (\exists j: l_b[j] \notin l_I)$ do
3. if $(|TeamA| < |TeamB|) \vee ((|TeamA| = |TeamB|) \wedge (randomBit() = 1))$ then
4. $k \leftarrow \min_i \{i: l_a[i] \notin l_I\}$
5. $l_I \leftarrow l_I + l_a[k]$
6. $TeamA \leftarrow TeamA \cup \{l_a[k]\}$
7. else
8. $k \leftarrow \min_i \{i: l_b[i] \notin l_I\}$
9. $l_I \leftarrow l_I + l_b[k]$
10. $TeamB \leftarrow TeamB \cup \{l_b[k]\}$
11. end if
12. end while

Output: Interleaving ranking $l_I, TeamA, TeamB$

Sease

T

DRAWBACK

- When comparing two very similar models.
  - **Model A:** $l_A = (a, b, c, d)$
  - **Model B:** $l_B = (b, c, d, a)$
- Suppose *c* to be the only relevant document.
- With this approach we can obtain four different interleaved lists:
  - $l_{I1} = (a_A, b_B, c_A, d_B)$
  - $l_{I2} = (b_B, a_A, c_B, d_A)$
  - $l_{I3} = (b_B, a_A, c_A, d_B)$
  - $l_{I4} = (a_A, b_B, c_B, d_A)$
- All of them putting *c* at the same rank.

Tie!
But Model B should be chosen as the best model!

[Online] Probabilistic Interleaving

There are different **types of interleaving**:

- **Balanced Interleaving:** alternate insertion with one model having the priority.
- **Team-Draft Interleaving:** method of team captains in team-matches.
- **Probabilistic Interleaving:** rely on probability distributions. Every documents have a non-zero probability to be added in the interleaved result list.

[Online] Probabilistic Interleaving

There are different **types of interleaving**:

- **Balanced Interleaving:** alternate insertion with one model having the priority.
- **Team-Draft Interleaving:** method of team captains in team-matches.
- **Probabilistic Interleaving:** rely on probability distributions

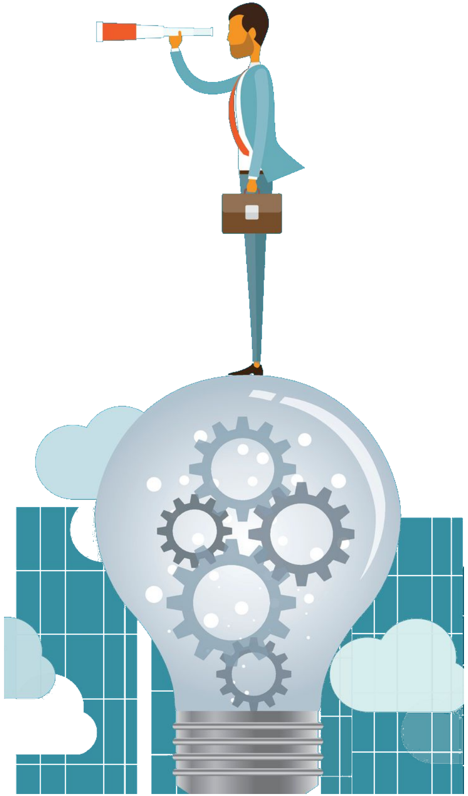
DRAWBACK

- The use of probability distribution could lead to a worse user experience. Less relevant document could be put higher.

https://sease.io/2020/05/online-testing-for-learning-to-rank-interleaving.html

Overview

- Learning to Rank
- Online Testing for Business
- Interleaving
- Apache Solr Implementation
- DEMO



Minimum Requirements

HAYSTACK

- Include the required contrib JARs. Note that by default paths are relative to the Solr core so they may need adjustments to your configuration, or an explicit specification of the $solr.install.dir.

```
<lib dir="${solr.install.dir:../../../..}/dist/" regex="solr-ltr-\d.*\.jar" />
```

- Declaration of the ltr query parser.

```
<queryParser name="ltr" class="org.apache.solr.ltr.search.LTRQParserPlugin"/>
```

- Configuration of the feature values cache.

```
<cache name="QUERY_DOC_FV"
class="solr.search.LRUCache"
size="4096"
initialSize="2048"
autowarmCount="4096"
regenerator="solr.search.NoOpRegenerator" />
```

Sease

Minimum Requirements

- Declaration of the [features] transformer.
```
<transformer name="features"
class="org.apache.solr.ltr.response.transform.LTRFeatureLoggerTransformerFactory">
<str name="fvCacheName">QUERY_DOC_FV</str>
</transformer>
```
- Declaration of the [interleaving] transformer.
```
<transformer name="interleaving"
class="org.apache.solr.ltr.response.transform.LTRInterleavingTransformerFactory"/>
```

Models

HAYSTACK

Implements the ranking function:

General form	Class	Specific examples
Linear	LinearModel	RankSVM, Pranking
Multiple Additive Trees	MultipleAdditiveTreesModel	LambdaMART, Gradient Boosted Regression Trees (GBRT)
Neural Network	NeuralNetworkModel	RankNet
(wrapper)	DefaultWrapperModel	(not applicable)
(custom)	(custom class extending AdapterModel)	(not applicable)
(custom)	(custom class extending LTRScoringModel)	(not applicable)

ease

Linear

HAYSTACK

- Computes the scores using a dot product

Example configuration:

```
{
  "class" : "org.apache.solr.ltr.model.LinearModel",
  "name" : "myModelName",
  "features" : [
    { "name" : "userTextTitleMatch" },
    { "name" : "originalScore" },
    { "name" : "isBook" }
  ],
  "params" : {
    "weights" : {
      "userTextTitleMatch" : 1.0,
      "originalScore" : 0.5,
      "isBook" : 0.1
    }
  }
}
```

https://lucene.apache.org/solr/8_8_0//solr-ltr/org/apache/solr/ltr/model/LinearModel.html

Sease

Multiple Additive Trees Model

- computes scores based on the summation of multiple weighted trees

```
{
  "class" : "org.apache.solr.ltr.model.MultipleAdditiveTreesModel",
  "name" : "multipleadditivetreesmodel",
  "features":[
      { "name" : "userTextTitleMatch"},
      { "name" : "originalScore"}
  ],
 }
```

```
"params" : {
    "trees" : [
        {
            "weight" : "1",
            "root": {
                "feature" : "userTextTitleMatch",
                "threshold" : "0.5",
                "left" : {
                    "value" : "-100"
                },
                "right" : {
                    "feature" : "originalScore",
                    "threshold" : "10.0",
                    "left" : {
                        "value" : "50"
                    },
                    "right" : {
                        "value" : "75"
                    }
                }
            }
        },
        {
            "weight" : "2",
            "root" : {
                "value" : "-10"
            }
        }
    ]
}
```

https://lucene.apache.org/solr/8_8_0//solr-ltr/org/apache/solr/ltr/model/MultipleAdditiveTreesModel.html

Neural Network

HAYSTACK

- computes scores using a neural network.

```
{
   "class" : "org.apache.solr.ltr.model.NeuralNetworkModel",
   "name" : "rankNetModel",
   "features" : [
      { "name" : "documentRecency" },
      { "name" : "isBook" },
      { "name" : "originalScore" }
   ],
"params" : {
      "layers" : [
         {
            "matrix" : [ [ 1.0, 2.0, 3.0 ],
                         [ 4.0, 5.0, 6.0 ],
                         [ 7.0, 8.0, 9.0 ],
                         [ 10.0, 11.0, 12.0 ] ],
            "bias" : [ 13.0, 14.0, 15.0, 16.0 ],
            "activation" : "sigmoid"
         },
         {
            "matrix" : [ [ 17.0, 18.0, 19.0, 20.0 ],
                         [ 21.0, 22.0, 23.0, 24.0 ] ],
            "bias" : [ 25.0, 26.0 ],
            "activation" : "relu"
         },
         {
            "matrix" : [ [ 27.0, 28.0 ],
                         [ 29.0, 30.0 ] ],
            "bias" : [ 31.0, 32.0 ],
            "activation" : "leakyrelu"
         },
         {
            "matrix" : [ [ 33.0, 34.0 ],
                         [ 35.0, 36.0 ] ],
            "bias" : [ 37.0, 38.0 ],
            "activation" : "tanh"
         },
         {
            "matrix" : [ [ 39.0, 40.0 ] ],
            "bias" : [ 41.0 ],
            "activation" : "identity"
         }
      ]
   }
```

https://lucene.apache.org/solr/8_8_0//solr-ltr/org/apache/solr/ltr/model/MultipleAdditiveTreesModel.html

Sease

Reranking

```
http://localhost:8983/solr/techproducts/query?q=test&rq={!ltr
model=myModel reRankDocs=100}&fl=id,score
```

To obtain the feature values computed during reranking, add [features] to the fl parameter, for example:

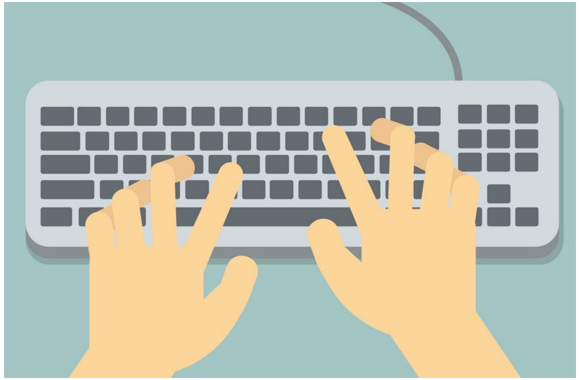
```
http://localhost:8983/solr/techproducts/query?q=test&rq={!ltr
model=myModel reRankDocs=100}&fl=id,score,[features]
```

To rerank using external feature information:

```
http://localhost:8983/solr/techproducts/query?q=test&rq={!ltr
model=myEfiModel efi.text=test efi.preferredManufacturer=Apache
efi.fromMobile=0 efi.answer=13}&fl=id,cat,manu,score,[features]
```

Hands On: Run a reranking query

http://localhost:8983/solr/books/query?q=subjects%3A(%22England%20--%20Fiction%22%20OR%20%22Mystery%20fiction%22)&rq=%7B!ltr%20model=linearModel1%20reRankDocs=100%20efi.favouriteSubject=%22Mystery%20fiction%22%20efi.fromMobile=1%20efi.age=25%20efi.userLanguage=%22en%22%7D&fl=id,title,subjects,downloads,score,[features]&debug=results



Reranking with Interleaving

```
http://localhost:8983/solr/techproducts/query?q=test&rq={!ltr
model=myModelA model=myModelB reRankDocs=100}&fl=id,score
```

To obtain the model that interleaving picked for a search result, computed during reranking, add [interleaving] to the fl parameter, for example:

```
http://localhost:8983/solr/techproducts/query?q=test&rq={!ltr
model=myModelA model=myModelB reRankDocs=100}&fl=id,score,[interleaving]
```

Reranking with Interleaving


```
{
  "responseHeader":{
    "status":0,
    "QTime":0,
    "params":{
      "q":"test",
      "fl":"id,score,[interleaving]",
      "rq":"{!ltr model=myModelA model=myModelB
reRankDocs=100}"}},

"response":{"numFound":2,"start":0,"maxScore":1.0005897,"docs":[
      {
        "id":"GB18030TEST",
        "score":1.0005897,
        "[interleaving]":"myModelB"},
      {
        "id":"UTF8TEST",
        "score":0.79656565,
        "[interleaving]":"myModelA"}]
  }}
```

Interleaving with Original Score

HAYSTACK

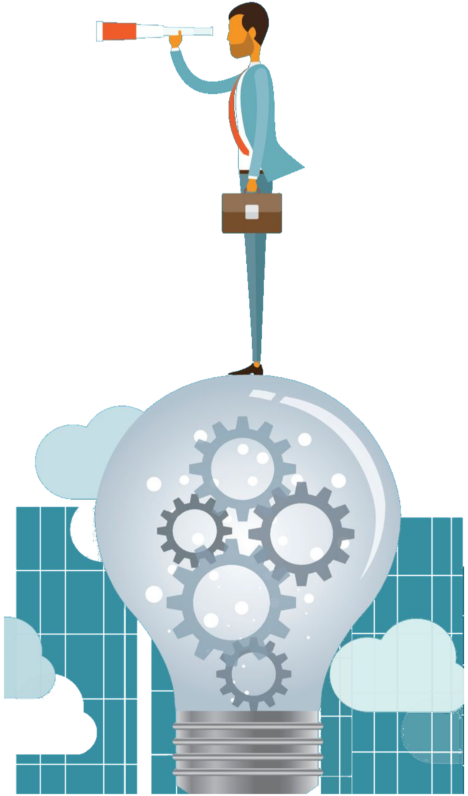
[http://localhost:8983/solr/books/query?q=subjects%3A(%22England%20--%20Fiction%22%20OR%20%22Mystery%20fiction%22)&rq={!ltr%20model=linearModel1%20model=_OriginalRanking_%20reRankDocs=100%20efi.favouriteSubject=%22Mystery%20fiction%22%20efi.fromMobile=1%20efi.age=25%20efi.userLanguage=%22en%22}&fl=id,title,subjects,downloads,score,[features],[interleaving]&debug=results](http://localhost:8983/solr/books/query?q=subjects%3A(%22England%20--%20Fiction%22%20OR%20%22Mystery%20fiction%22)&rq={!ltr%20model=linearModel1%20model=_OriginalRanking_%20reRankDocs=100%20efi.favouriteSubject=%22Mystery%20fiction%22%20efi.fromMobile=1%20efi.age=25%20efi.userLanguage=%22en%22}&fl=id,title,subjects,downloads,score,[features],[interleaving]&debug=results)


```
"response":{"numFound":2,"start":0,"maxScore":1.0005897,"docs
":[
    {
      "id":"GB18030TEST",
      "score":1.0005897,
      "[interleaving]":"_OriginalRanking_"},
    {
      "id":"UTF8TEST",
      "score":0.79656565,
      "[interleaving]":"myModel"}]
 }}
```


Sease

Overview

- Learning to Rank
- Online Testing for Business
- Interleaving
- Apache Solr Implementation
- DEMO



Future Works

```
http://localhost:8983/solr/techproducts/query?q=test&rq={!ltr model=myModelA model=myModelB
reRankDocs=100 interleavingAlgorithm=TeamDraft}&fl=id,score
```

Currently the only (and default) algorithm supported is **'TeamDraft'**.

How to contribute

Do you want to contribute a new Interleaving Algorithm?

You just need to :

- implement the solr/contrib/ltr/src/java/org/apache/solr/ltr/interleaving/Interleaving.java interface in a new class
- add the new algorithm in the package: org.apache.solr.ltr.interleaving.algorithms
- add the new algorithm reference in the org.apache.solr.ltr.interleaving.Interleaving#getImplementation

Limitations

- [Distributed Search] Sharding is not supported

Thanks!

Спасибо
Gracias
شكرا
Obrigado
Спасибо
Dank U
Grazie
Ευχαριστώ
Danke
Ευχαριστώ
شكرا
Merci
Thank You
Ngiyabonga
Dank U
Diolch
Obrigado
Thank You
Ngiyabonga
Tack
Merci
Ευχαριστώ
Ευχαριστώ
Dzięki
Danke
Grazie
תודה
Dank U
Terima Kasih
Diolch
Grazie
Tack
Tack
Dzięki
Diolch
Gracias
Merci
תודה