Online Learning to Rank Framework

Anne Schuth - University of Amsterdam
Katja Hofmann - Microsoft Research
Shimon Whiteson - University of Amsterdam
Maarten de Rijke - University of Amsterdam

November 1, 2013, CIKM Living Lab Workshop, San Francisco

bitbucket.org/ilps/lerot

- Learn rankings using user feedback

bitbucket.org/ilps/lerot

- Learn rankings using user feedback

  - clicks are a natural by product of user interaction

bitbucket.org/ilps/lerot

- Learn rankings using user feedback

  - clicks are a natural by product of user interaction

  - no more need for (unnatural) annotations

bitbucket.org/ilps/lerot

- Learn rankings using user feedback

    - clicks are a natural by product of user interaction

    - no more need for (unnatural) annotations

    - personalize

bitbucket.org/ilps/lerot

- Learn rankings using user feedback

    - clicks are a natural by product of user interaction

    - no more need for (unnatural) annotations

    - personalize

- But: we (academics) do not have acces to users

bitbucket.org/ilps/lerot

- Learn rankings using user feedback

    - clicks are a natural by product of user interaction

    - no more need for (unnatural) annotations

    - personalize

- But: we (academics) do not have acces to users

    - we can simulate users using user models

![Lerot logo] bitbucket.org/ilps/lerot

- Learn rankings using user feedback

    - clicks are a natural by product of user interaction

    - no more need for (unnatural) annotations

    - personalize

- But: we (academics) do not have acces to users

    - we can simulate users using user models

- Test base for learning, evaluation, user models

bitbucket.org/ilps/lerot

- Learn rankings using user feedback

  - clicks are a natural by product of user interaction

  - no more need for (unnatural) annotations

  - personalize

- But: we (academics) do not have acces to users

  - we can simulate users using user models

- Test base for learning, evaluation, user models

  - we do not hurt actual users

```
import sys, random
import environment, retrieval_system, evaluation, query
```

```
import sys, random
import environment, retrieval_system, evaluation, query

user_model = environment.CascadeUserModel([…])
```

```python
import sys, random
import environment, retrieval_system, evaluation, query

user_model = environment.CascadeUserModel([...])
learner = retrieval_system.ListwiseLearningSystem(comparison="comparison.TeamDraft")
```

bitbucket.org/ilps/lerot

```python
import sys, random
import environment, retrieval_system, evaluation, query

user_model = environment.CascadeUserModel([...])
learner = retrieval_system.ListwiseLearningSystem(comparison="comparison.TeamDraft")
evaluation = evaluation.NdcgEval([...])
```

bitbucket.org/ilps/lerot

```python
import sys, random
import environment, retrieval_system, evaluation, query

user_model = environment.CascadeUserModel([...])
learner = retrieval_system.ListwiseLearningSystem(comparison="comparison.TeamDraft")
evaluation = evaluation.NdcgEval([...])
train = query.load_queries(sys.argv[1], [...])
test = query.load_queries(sys.argv[2], [...])
```

bitbucket.org/ilps/lerot

```python
import sys, random
import environment, retrieval_system, evaluation, query

user_model = environment.CascadeUserModel([...])
learner = retrieval_system.ListwiseLearningSystem(comparison="comparison.TeamDraft")
evaluation = evaluation.NdcgEval([...])
train = query.load_queries(sys.argv[1], [...])
test = query.load_queries(sys.argv[2], [...])

while True:
```

bitbucket.org/ilps/lerot

```python
import sys, random
import environment, retrieval_system, evaluation, query

user_model = environment.CascadeUserModel([...])
learner = retrieval_system.ListwiseLearningSystem(comparison="comparison.TeamDraft")
evaluation = evaluation.NdcgEval([...])
train = query.load_queries(sys.argv[1], [...])
test = query.load_queries(sys.argv[2], [...])

while True:
    q = train[random.choice(train.keys())]
```
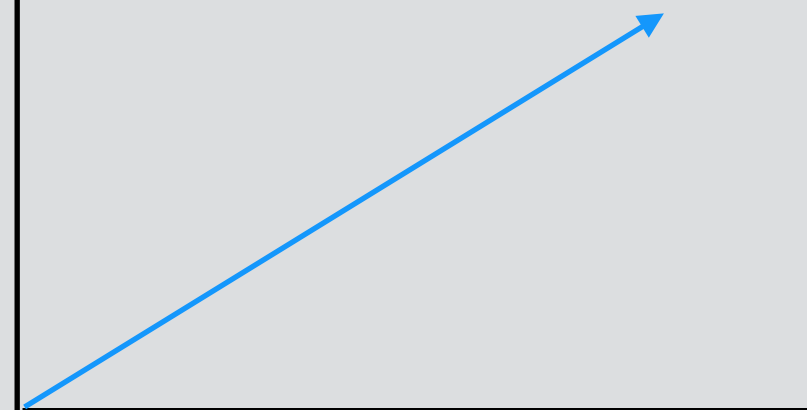
bitbucket.org/ilps/lerot

```python
import sys, random
import environment, retrieval_system, evaluation, query

user_model = environment.CascadeUserModel([…])
learner = retrieval_system.ListwiseLearningSystem(comparison="comparison.TeamDraft")
evaluation = evaluation.NdcgEval([…])
train = query.load_queries(sys.argv[1], […])
test = query.load_queries(sys.argv[2], […])

while True:
    q = train[random.choice(train.keys())]
    l = learner.get_ranked_list(q)
```

bitbucket.org/ilps/lerot

```python
import sys, random
import environment, retrieval_system, evaluation, query

user_model = environment.CascadeUserModel([…])
learner = retrieval_system.ListwiseLearningSystem(comparison="comparison.TeamDraft")
evaluation = evaluation.NdcgEval([…])
train = query.load_queries(sys.argv[1], […])
test = query.load_queries(sys.argv[2], […])

while True:
    q = train[random.choice(train.keys())]
    l = learner.get_ranked_list(q)
    c = user_model.get_clicks(l, q.get_labels())
```

bitbucket.org/ilps/lerot

```python
import sys, random
import environment, retrieval_system, evaluation, query

user_model = environment.CascadeUserModel([...])
learner = retrieval_system.ListwiseLearningSystem(comparison="comparison.TeamDraft")
evaluation = evaluation.NdcgEval([...])
train = query.load_queries(sys.argv[1], [...])
test = query.load_queries(sys.argv[2], [...])

while True:
    q = train[random.choice(train.keys())]
    l = learner.get_ranked_list(q)
    c = user_model.get_clicks(l, q.get_labels())
    s = learner.update_solution(c)
```

```python
import sys, random
import environment, retrieval_system, evaluation, query

user_model = environment.CascadeUserModel([...])
learner = retrieval_system.ListwiseLearningSystem(comparison="comparison.TeamDraft")
evaluation = evaluation.NdcgEval([...])
train = query.load_queries(sys.argv[1], [...])
test = query.load_queries(sys.argv[2], [...])

while True:
    q = train[random.choice(train.keys())]
    l = learner.get_ranked_list(q)
    c = user_model.get_clicks(l, q.get_labels())
    s = learner.update_solution(c)
    print evaluation.evaluate_all(s, test)
```

bitbucket.org/ilps/lerot

```python
import sys, random
import environment, retrieval_system, evaluation, query

user_model = environment.CascadeUserModel([...])
learner = retrieval_system.ListwiseLearn...              Draft")
evaluation = evaluation.NdcgEval([...])
train = query.load_queries(sys.argv[1], [
test = query.load_queries(sys.argv[2], [...

while True:
    q = train[random.choice(train.keys()
    l = learner.get_ranked_list(q)
    c = user_model.get_clicks(l, q.get_la...
    s = learner.update_solution(c)
    print evaluation.evaluate_all(s, test)
```

W_BM25

W_Pagerank

```python
import sys, random
import environment, retrieval_system, evaluation, query

user_model = environment.CascadeUserModel([...])
learner = retrieval_system.ListwiseLearn...            ...raft")
evaluation = evaluation.NdcgEval([…])
train = query.load_queries(sys.argv[1], [
test = query.load_queries(sys.argv[2], [.

while True:
    q = train[random.choice(train.keys()
    l = learner.get_ranked_list(q)
    c = user_model.get_clicks(l, q.get_la
    s = learner.update_solution(c)
    print evaluation.evaluate_all(s, test)
```

W$_{BM25}$

W$_{Pagerank}$

```python
import sys, random
import environment, retrieval_system, evaluation, query

user_model = environment.CascadeUserModel([...])
learner = retrieval_system.ListwiseLearn          Draft")
evaluation = evaluation.NdcgEval([...])
train = query.load_queries(sys.argv[1], [
test = query.load_queries(sys.argv[2], [.

while True:
    q = train[random.choice(train.keys()
    l = learner.get_ranked_list(q)
    c = user_model.get_clicks(l, q.get_la
    s = learner.update_solution(c)
    print evaluation.evaluate_all(s, test)
```

W_{BM25}

$W_{Pagerank}$

```python
import sys, random
import environment, retrieval_system, evaluation, query

user_model = environment.CascadeUserModel([...])
learner = retrieval_system.ListwiseLearn                    Draft")
evaluation = evaluation.NdcgEval([...])
train = query.load_queries(sys.argv[1], [
test = query.load_queries(sys.argv[2], [.

while True:
    q = train[random.choice(train.keys()
    l = learner.get_ranked_list(q)
    c = user_model.get_clicks(l, q.get_la
    s = learner.update_solution(c)
    print evaluation.evaluate_all(s, test)
```

W_BM25

W_Pagerank

```
import sys, random
import environment, retrieval_system, evaluation, query

user_model = environment.CascadeUserModel([...])
learner = retrieval_system.ListwiseLearn...                ...raft")
evaluation = evaluation.NdcgEval([...])
train = query.load_queries(sys.argv[1], [...
test = query.load_queries(sys.argv[2], [...

while True:
    q = train[random.choice(train.keys()...
    l = learner.get_ranked_list(q)
    c = user_model.get_clicks(l, q.get_la...
    s = learner.update_solution(c)
    print evaluation.evaluate_all(s, test)
```

```python
import sys, random
import environment, retrieval_system, evaluation, query

user_model = environment.CascadeUserModel([...])
learner = retrieval_system.ListwiseLearn                  )raft")
evaluation = evaluation.NdcgEval([...])
train = query.load_queries(sys.argv[1],
test = query.load_queries(sys.argv[2], [.

while True:
    q = train[random.choice(train.keys(
    l = learner.get_ranked_list(q)
    c = user_model.get_clicks(l, q.get_la
    s = learner.update_solution(c)
    print evaluation.evaluate_all(s, test)
```

```
import sys, random
import environment, retrieval_system, evaluation, query

user_model = environment.CascadeUserModel([...])
learner = retrieval_system.ListwiseLearn                        Draft")
evaluation = evaluation.NdcgEval([...])
train = query.load_queries(sys.argv[1], [
test = query.load_queries(sys.argv[2], [..

while True:
    q = train[random.choice(train.keys()
    l = learner.get_ranked_list(q)
    c = user_model.get_clicks(l, q.get_la
    s = learner.update_solution(c)
    print evaluation.evaluate_all(s, test)
```

B

A

```python
import sys, random
import environment, retrieval_system, evaluation, query

user_model = environment.CascadeUserModel([...])
learner = retrieval_system.ListwiseLearn...            Draft")
evaluation = evaluation.NdcgEval([...])
train = query.load_queries(sys.argv[1], [...
test = query.load_queries(sys.argv[2], [...

while True:
    q = train[random.choice(train.keys()...
    l = learner.get_ranked_list(q)
    c = user_model.get_clicks(l, q.get_la...
    s = learner.update_solution(c)
    print evaluation.evaluate_all(s, test)
```

```python
import sys, random
import environment, retrieval_system, evaluation, query

user_model = environment.CascadeUserModel([...])
learner = retrieval_system.ListwiseLearn              Draft")
evaluation = evaluation.NdcgEval([...])
train = query.load_queries(sys.argv[1], [
test = query.load_queries(sys.argv[2], [.

while True:
    q = train[random.choice(train.keys()
    l = learner.get_ranked_list(q)
    c = user_model.get_clicks(l, q.get_la
    s = learner.update_solution(c)
    print evaluation.evaluate_all(s, test)
```

bitbucket.org/ilps/lerot

```python
import sys, random
import environment, retrieval_system, evaluation, query

user_model = environment.CascadeUserModel([...])
learner = retrieval_system.ListwiseLearningSystem(comparison="comparison.TeamDraft")
evaluation = evaluation.NdcgEval([...])
train = query.load_queries(sys.argv[1], [...])
test = query.load_queries(sys.argv[2], [...])

while True:
    q = train[random.choice(train.keys())]
    l = learner.get_ranked_list(q)
    c = user_model.get_clicks(l, q.get_labels())
    s = learner.update_solution(c)
    print evaluation.evaluate_all(s, test)
```
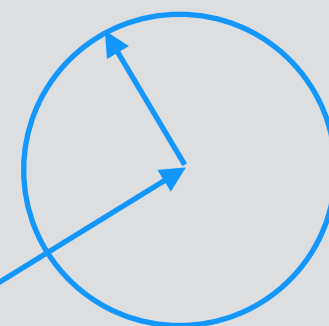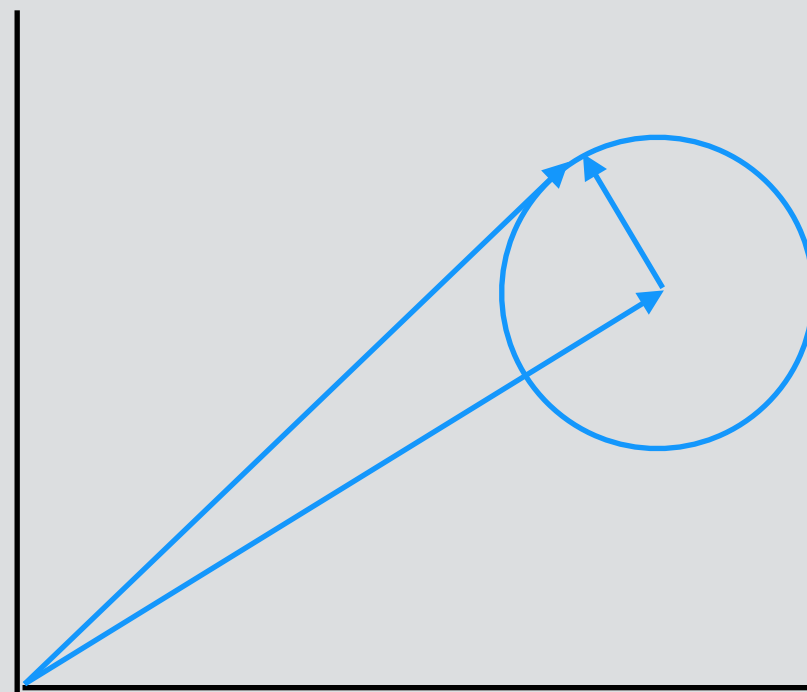
```python
import sys, random
import environment, retrieval_system, evaluation, query

user_model = environment.CascadeUserModel([...])
learner = retrieval_system.ListwiseLearningSystem(comparison="comparison.TeamDraft")
evaluation = evaluation.NdcgEval([...])
train = query.load_queries(sys.argv[1], [...])
test = query.load_queries(sys.argv[2], [...])

while True:
    q = train[random.choice(train.keys())]
    l = learner.get_ranked_list(q)
    c = user_model.get_clicks(l, q.get_labels())
    s = learner.update_solution(c)
    print evaluation.evaluate_all(s, test)
```
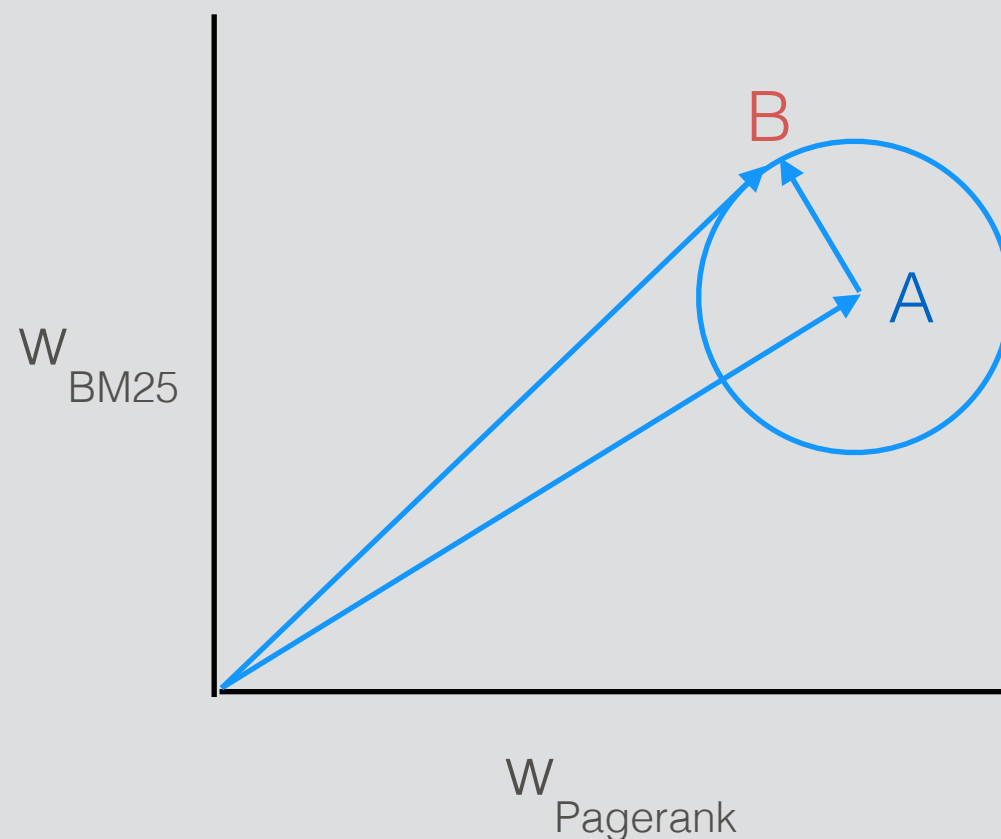
```python
import sys, random
import environment, retrieval_system, evaluation, query

user_model = environment.CascadeUserModel([...])
learner = retrieval_system.ListwiseLearningSystem(comparison="comparison.TeamDraft")
evaluation = evaluation.NdcgEval([...])
train = query.load_queries(sys.argv[1], [...])
test = query.load_queries(sys.argv[2], [...])

while True:
    q = train[random.choice(train.keys())]
    l = learner.get_ranked_list(q)
    c = user_model.get_clicks(l, q.get_labels())
    s = learner.update_solution(c)
    print evaluation.evaluate_all(s, test)
```

```python
import sys, random
import environment, retrieval_system, evaluation, query

user_model = environment.CascadeUserModel([...])
learner = retrieval_system.ListwiseLearningSystem(comparison="comparison.TeamDraft")
evaluation = evaluation.NdcgEval([...])
train = query.load_queries(sys.argv[1], [...])
test = query.load_queries(sys.argv[2], [...])

while True:
    q = train[random.choice(train.keys())]
    l = learner.get_ranked_list(q)
    c = user_model.get_clicks(l, q.get_labels())
    s = learner.update_solution(c)
    print evaluation.evaluate_all(s, test)
```

```python
import sys, random
import environment, retrieval_system, evaluation, query

user_model = environment.CascadeUserModel([...])
learner = retrieval_system.ListwiseLearningSystem(comparison="comparison.TeamDraft")
evaluation = evaluation.NdcgEval([...])
train = query.load_queries(sys.argv[1], [...])
test = query.load_queries(sys.argv[2], [...])

while True:
    q = train[random.choice(train.keys())]
    l = learner.get_ranked_list(q)
    c = user_model.get_clicks(l, q.get_labels())
    s = learner.update_solution(c)
    print evaluation.evaluate_all(s, test)
```

bitbucket.org/ilps/lerot

```python
import sys, random
import environment, retrieval_system, evaluation, query

user_model = environment.CascadeUserModel([...])
learner = retrieval_system.ListwiseLearningSystem(comparison="comparison.TeamDraft")
evaluation = evaluation.NdcgEval([...])
train = query.load_queries(sys.argv[1], [...])
test = query.load_queries(sys.argv[2], [...])

while True:
    q = train[random.choice(train.keys())]
    l = learner.get_ranked_list(q)
    c = user_model.get_clicks(l, q.get_labels())
    s = learner.update_solution(c)
    print evaluation.evaluate_all(s, test)
```
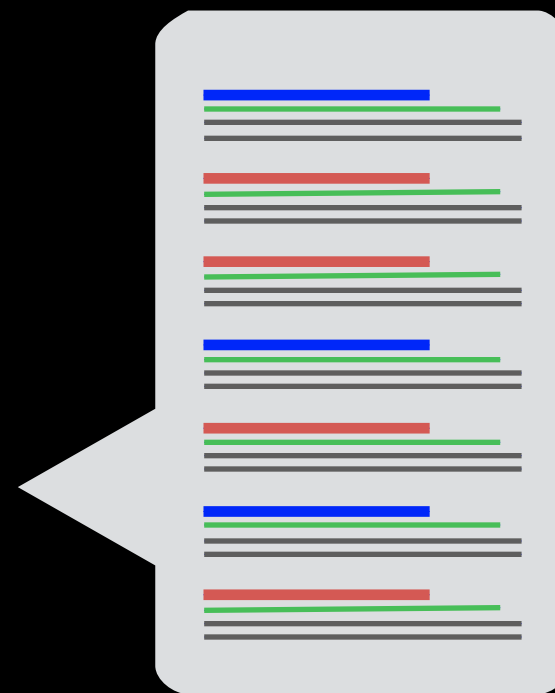
```python
import sys, random
import environment, retrieval_system, evaluation, query

user_model = environment.CascadeUserModel([...])
learner = retrieval_system.ListwiseLearningSystem(comparison="comparison.TeamDraft")
evaluation = evaluation.NdcgEval([...])
train = query.load_queries(sys.argv[1], [...])
test = query.load_queries(sys.argv[2], [...])

while True:
    q = train[random.choice(train.keys())]
    l = learner.get_ranked_list(q)
    c = user_model.get_clicks(l, q.get_labels())
    s = learner.update_solution(c)
    print evaluation.evaluate_all(s, test)
```
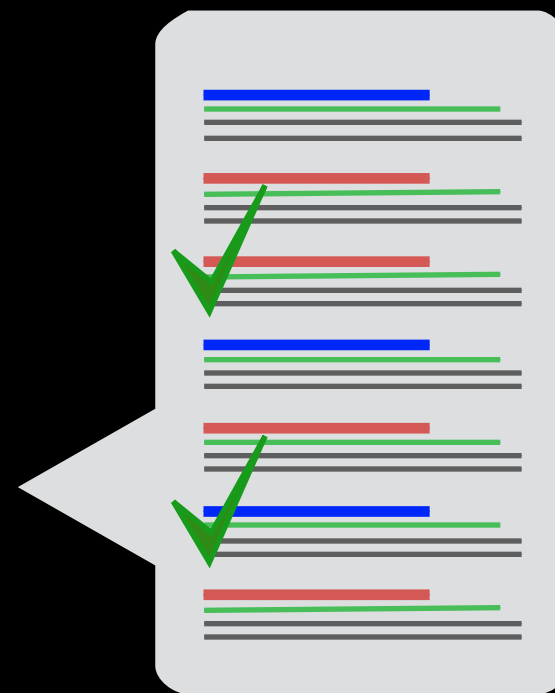
```python
import sys, random
import environment, retrieval_system, evaluation, query

user_model = environment.CascadeUserModel([…])
learner = retrieval_system.ListwiseLearningSystem(comparison="comparison.TeamDraft")
evaluation = evaluation.NdcgEval([…])
train = query.load_queries(sys.argv[1], […])
test = query.load_queries(sys.argv[2], […])

while True:
    q = train[random.choice(train.keys())]
    l = learner.get_ranked_list(q)
    c = user_model.get_clicks(l, q.get_labels())
    s = learner.update_solution(c)
    print evaluation.evaluate_all(s, test)
```
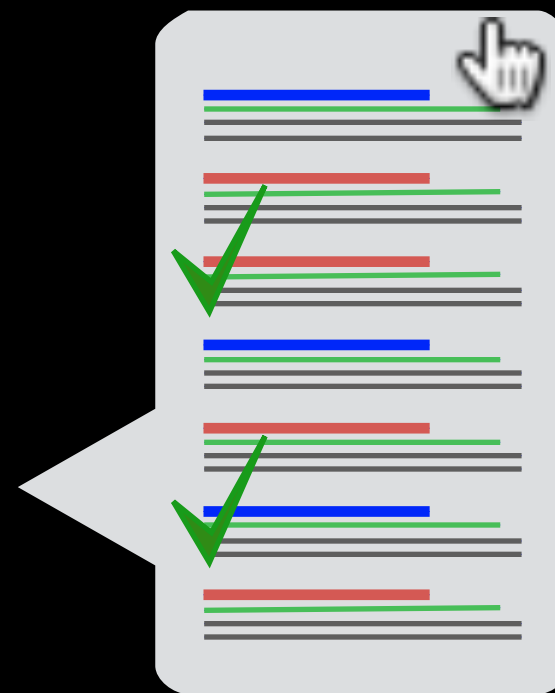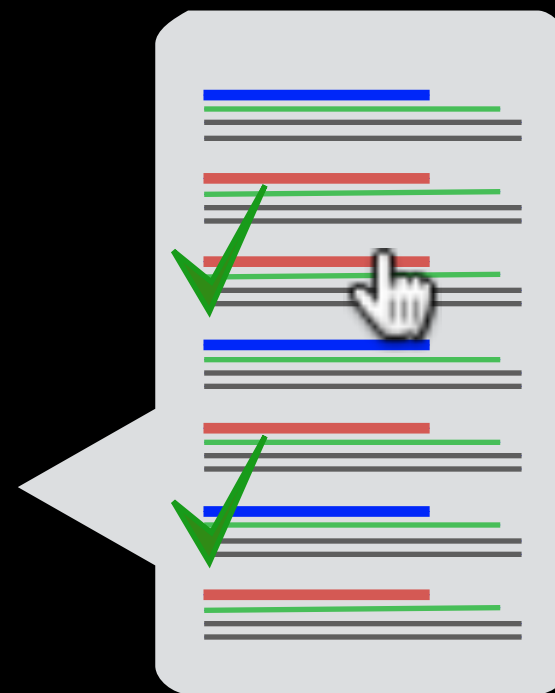
![Lerot logo] bitbucket.org/ilps/lerot

```python
import sys, random
import environment, retrieval_system, evaluation, query

user_model = environment.CascadeUserModel([...])
learner = retrieval_system.ListwiseLearnin                nDraft")
evaluation = evaluation.NdcgEval([...])
train = query.load_queries(sys.argv[1], [...
test = query.load_queries(sys.argv[2], [...]

while True:
    q = train[random.choice(train.keys())]
    l = learner.get_ranked_list(q)
    c = user_model.get_clicks(l, q.get_lab
    s = learner.update_solution(c)
    print evaluation.evaluate_all(s, test)
```
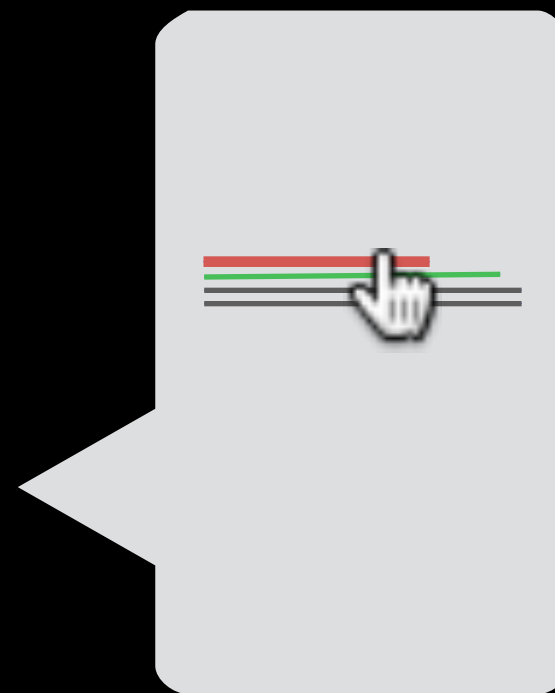
W_{BM25}

W_{Pagerank}

B

A

```python
import sys, random
import environment, retrieval_system, evaluation, query

user_model = environment.CascadeUserModel([...])
learner = retrieval_system.ListwiseLearnin...            nDraft")
evaluation = evaluation.NdcgEval([...])
train = query.load_queries(sys.argv[1], [...
test = query.load_queries(sys.argv[2], [...]

while True:
    q = train[random.choice(train.keys())]
    l = learner.get_ranked_list(q)
    c = user_model.get_clicks(l, q.get_lab
    s = learner.update_solution(c)
    print evaluation.evaluate_all(s, test)
```
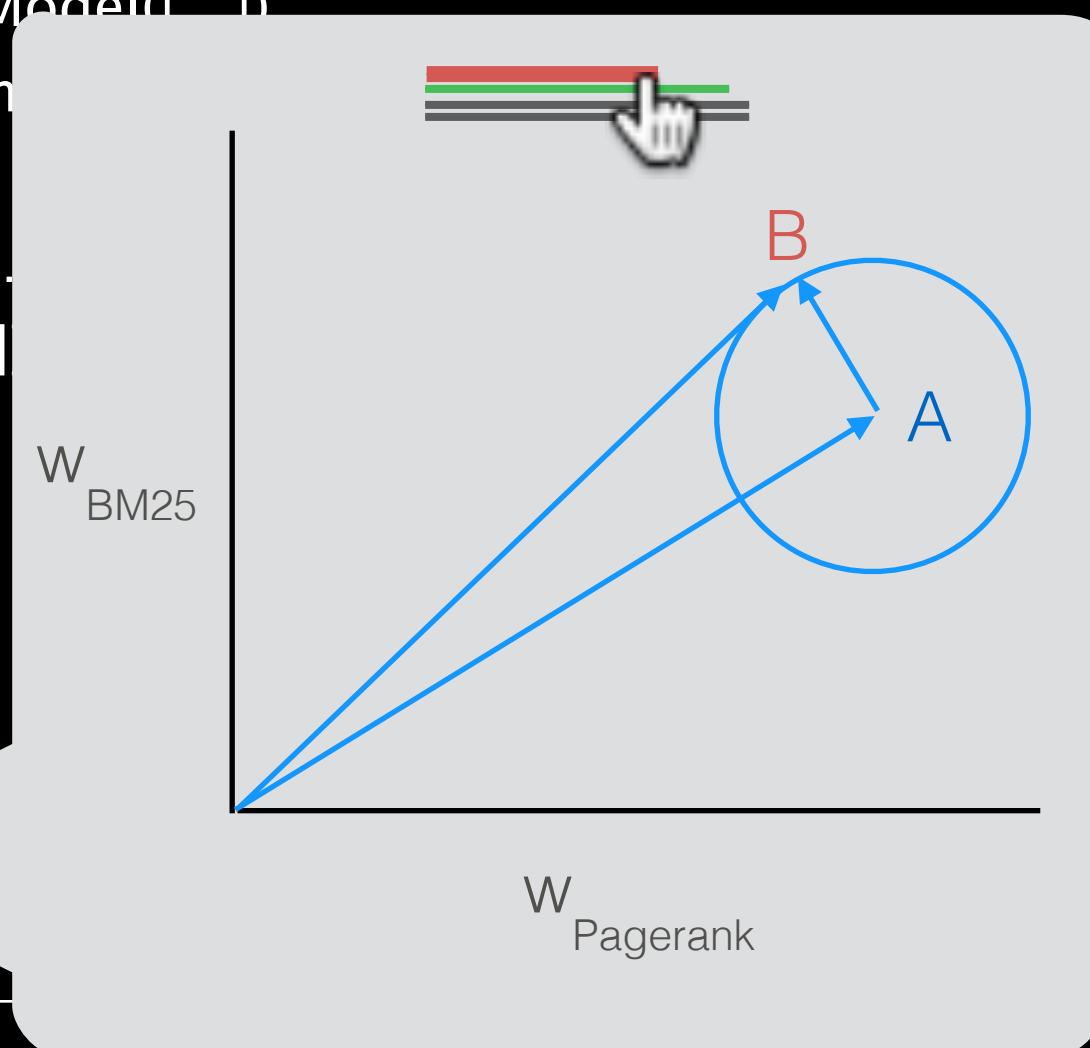
```python
import sys, random
import environment, retrieval_system, evaluation, query

user_model = environment.CascadeUserModel([...])
learner = retrieval_system.ListwiseLearnin...                    nDraft")
evaluation = evaluation.NdcgEval([...])
train = query.load_queries(sys.argv[1], [...
test = query.load_queries(sys.argv[2], [...]

while True:
    q = train[random.choice(train.keys())]
    l = learner.get_ranked_list(q)
    c = user_model.get_clicks(l, q.get_lab
    s = learner.update_solution(c)
    print evaluation.evaluate_all(s, test)
```
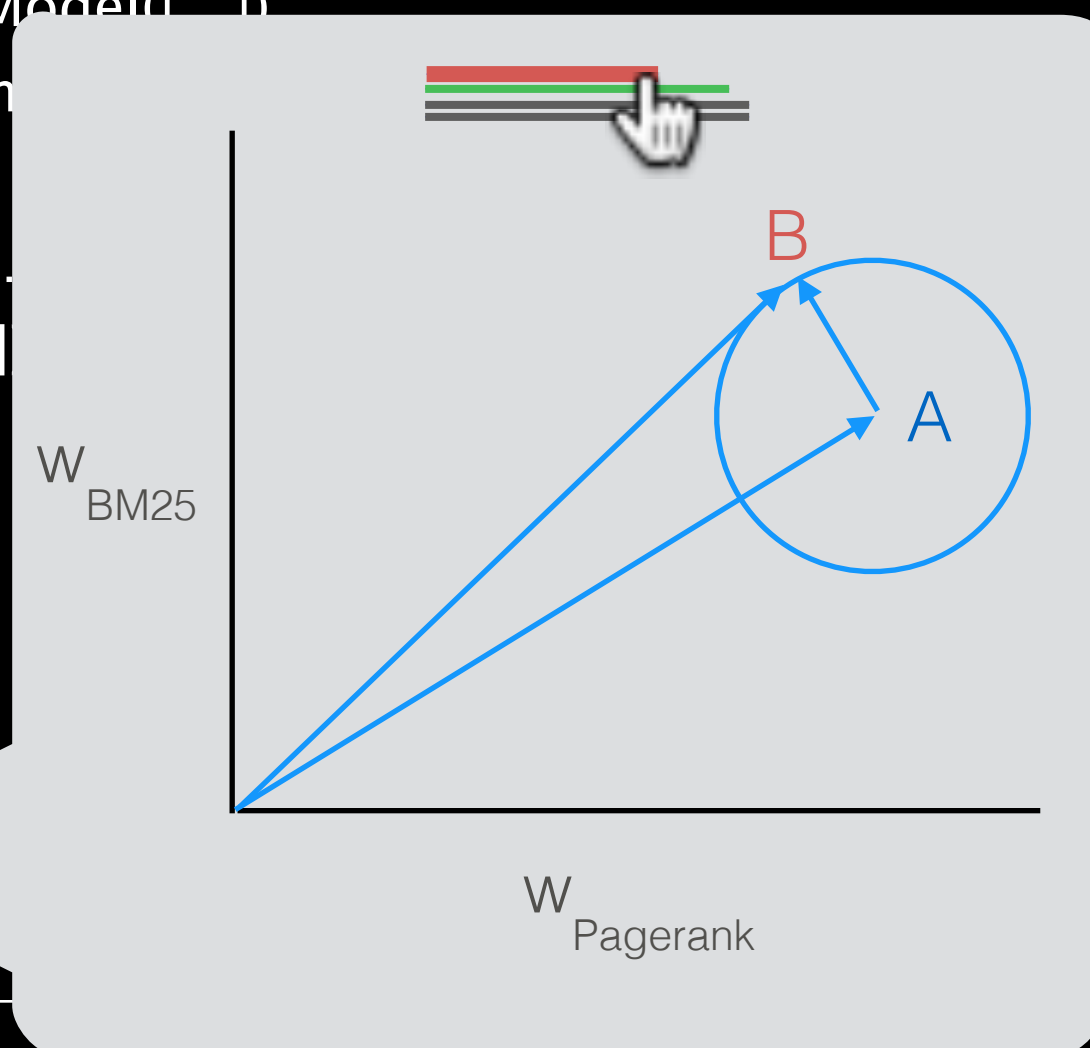
```python
import sys, random
import environment, retrieval_system, evaluation, query

user_model = environment.CascadeUserModel([...])
learner = retrieval_system.ListwiseLearnin...                    nDraft")
evaluation = evaluation.NdcgEval([...])
train = query.load_queries(sys.argv[1], [...
test = query.load_queries(sys.argv[2], [...]

while True:
    q = train[random.choice(train.keys())]
    l = learner.get_ranked_list(q)
    c = user_model.get_clicks(l, q.get_lab
    s = learner.update_solution(c)
    print evaluation.evaluate_all(s, test)
```
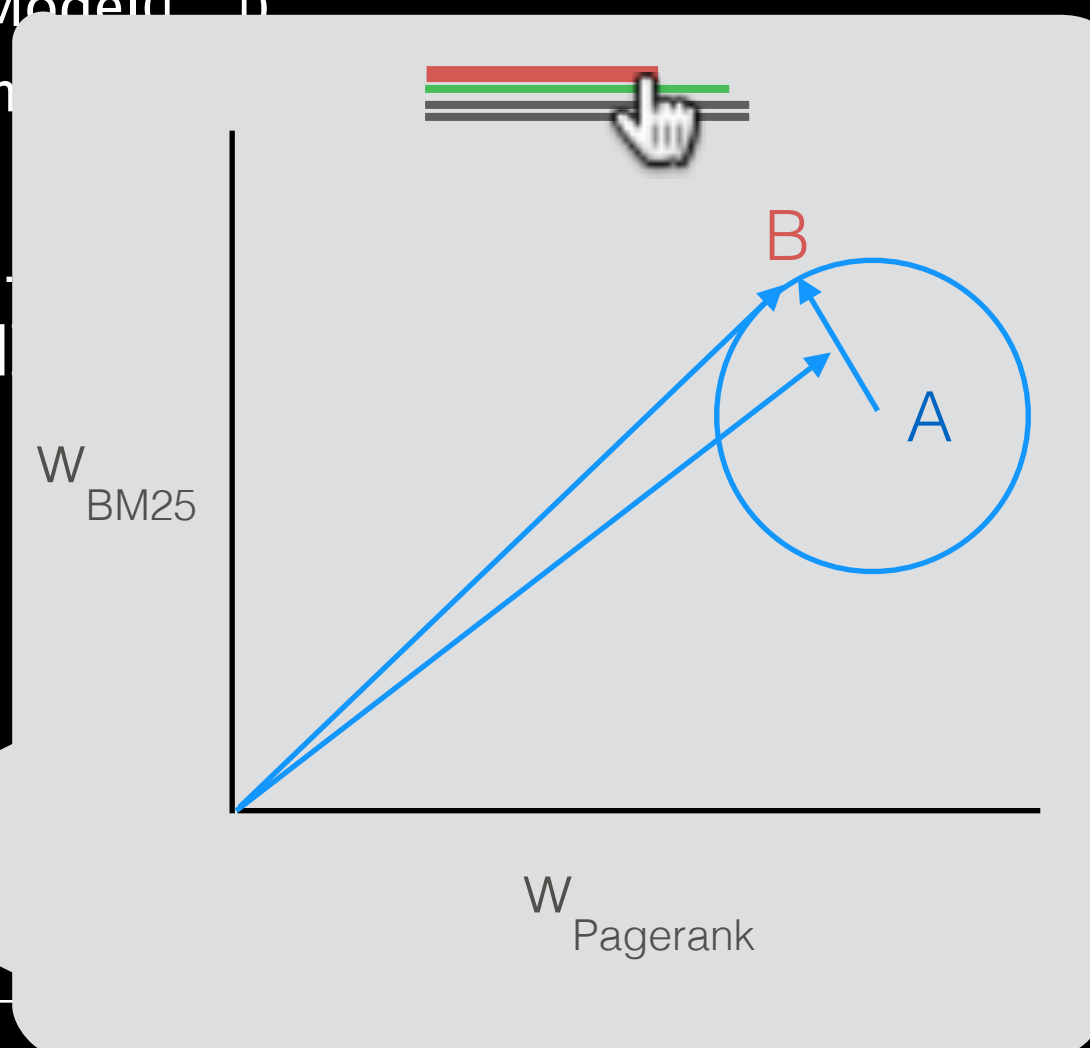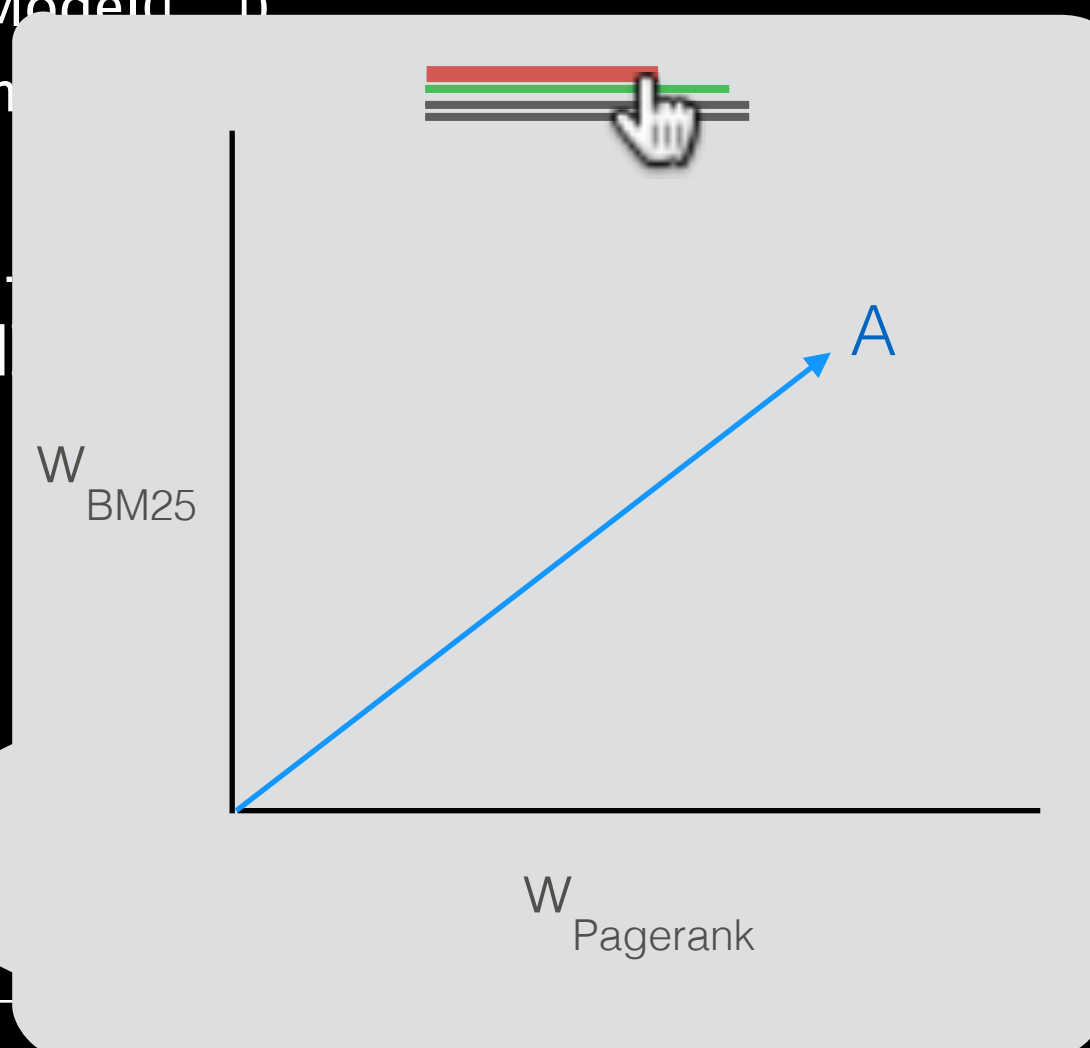
bitbucket.org/ilps/lerot

- Learning approaches

  - pairwise learning
    (Joachims 2002, Zhang 2004, Sculley 2009)

  - listwise learning

    - dueling bandit gradient descent
      (Yue and Joachims 2009)

    - candidate preselection
      (Hofmann et al 2013)

bitbucket.org/ilps/lerot

- Interleaving methods

    - balanced interleave (Joachims et al 2006, Radlinski et al 2008)

    - team draft interleave (Radlinski et al 2008)

    - document constraints interleave (He et al 2009)

    - probabilistic interleave (Hofmann et al 2011)

    - optimized interleave (Radlinski and Craswell 2013)

    - vertical aware team draft interleave(Chuklin et al 2013)

bitbucket.org/ilps/lerot

- User models

  - dependent click model
    (Guo et al 2009)

  - cascade click model
    (Craswell et al 2008)

  - random click model

  - federated click model
    (Chen et al 2012)

bitbucket.org/ilps/lerot

thank you

download Lerot from
https://bitbucket.org/ilps/lerot

LiMo SINe