Finding Similar Stack Traces

Irving Muller Rodrigues
Polytechnique Montreal
irving.muller-rodrigues@polymtl.ca

Polytechnique de Montréal
Laboratoire DORSAL

Software crash

*https://www.freepik.com/vectors/computer

Software crash

Crash Report

```
 1  Date: 2016-01-20T22:11:48.034Z
 2  Product: XXXXXXXXXXXX
 3  Version: 144.3143
 4  Action: null
 5  OS: Mac OS X
 6  Java: Oracle Corporation 1.8.0_40-release
 7  Message: new child is an ancestor
 8
 9  java.lang.IllegalArgumentException: new child is an ancestor
10    at javax.swing.tree.DefaultMutableTreeNode.insert(DefaultMutableTreeNode.java:179)
11    at javax.swing.tree.DefaultMutableTreeNode.add(DefaultMutableTreeNode.java:411)
12    at com.openapi.application.impl.ApplicationImpl$8.run(ApplicationImpl.java:374)
    .....
41    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
42    at java.lang.Thread.run(Thread.java:745)
43    at org.ide.PooledThreadExecutor$2$1.run ....
```

Crash Report

```
 1  Date: 2016-01-20T22:11:48.834Z
 2  Product: XXXXXXXXXXXX
 3  Version: 144.3143
 4  Action: null
 5  OS: Mac OS X
 6  Java: Oracle Corporation 1.8.0_40-release
 7  Message: new child is an ancestor
 8
 9  java.lang.IllegalArgumentException: new child is an ancestor
10    at javax.swing.tree.DefaultMutableTreeNode.insert(DefaultMutableTreeNode.java:179)
11    at javax.swing.tree.DefaultMutableTreeNode.add(DefaultMutableTreeNode.java:411)
12    at com.openapi.application.impl.ApplicationImpl$8.run(ApplicationImpl.java:374)
    .....
41    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
42    at java.lang.Thread.run(Thread.java:745)
43    at org.ide.PooledThreadExecutor$2$1.run ....
```

Stack Trace

- Compare the similarity between stack traces (crash dumps)
  - Useful for call stacks
- Group stack traces
  - Prioritization bug fixing: focus on the most frequent and critical issues
  - Problem diagnosis: complementary information about an error

Stack Trace

- Compare the similarity between stack traces (crash dumps)
  - Useful for call stacks
- Group stack traces
  - Prioritization bug fixing: focus on the most frequent and critical issues
  - Problem diagnosis: complementary information about an error

Stack Trace

- Compare the similarity between stack traces (crash dumps)
  - Useful for call stacks
- Group stack traces
  - Prioritization bug fixing: focus on the most frequent and critical issues
  - Problem diagnosis: complementary information about an error

Stack Trace

- Compare the similarity between stack traces (crash dumps)
  - Useful for call stacks
- Group stack traces
  - Prioritization bug fixing: focus on the most frequent and critical issues
  - Problem diagnosis: complementary information about an error

TF-IDF

Stack trace

```
java.lang.ArrayIndexOutOfBoundsException: 4
  at sat4j.pb.PseudoOptDecorator.model(Unknown Source)
  at sat4j.pb.OptToPBSATAdapter.model(Unknown Source)
  at eclipse.Projector.backToIU(Projector.java:67)
  at eclipse.Projector.invokeSolver(Projector.java:64)
  at eclipse.SimplePlanner.getPlan(SimplePlanner.java:24)
  at eclipse.PatchTest3.testCompleteScenario(PatchTest3.java:90)
  at reflect.NativeMethodImpl.invoke0(Native Method)
  at reflect.NativeMethodImpl.invoke(NativeMethodImpl.java:39)
```

Bag of words - Method names

sat4j.pb.PseudoOptDecorator.model,
sat4j.pb.OptToPBSATAdapter.model,
eclipse.Projector.backToIU,
eclipse.Projector.invokeSolver,
eclipse.SimplePlanner.getPlan,
eclipse.PatchTest3.testCompleteScenario,
reflect.NativeMethodImpl.invoke0,
reflect.NativeMethodImpl.invoke

TF-IDF

$$similarity = \sum_{word \in query} term\_freq(word, doc) * IDF(word, doc)$$

TF-IDF

Problem: position information is lost

Optimal Global Alignment

- Equivalent to Edit Distance
- Find the best alignment between sequences
  - Dynamic programming

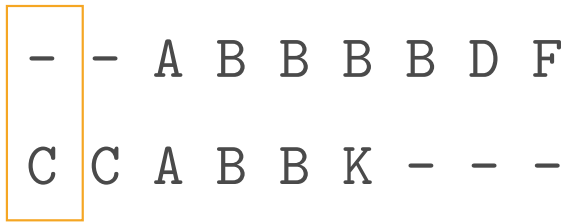
Optimal Global Alignment

- Equivalent to Edit Distance
- Find the best alignment between sequences
  - Dynamic programming

Global Alignment Types

```
- - A B B B B D F
C C A B B K - - -
```

Global Alignment Types



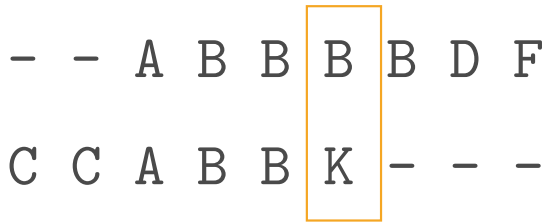
Alignment between an element and a gap

Global alignment

```
- - A B B B B D F
C C A B B K - - -
```

Match

Global Alignment Types



Mismatch

Optimal Global Alignment

- Query:$(q_1, q_2, \ldots, q_i)$
- Candidate:$(c_1, c_2, \ldots, c_j)$
- Matrix M:
  - $M_{i,j}$ = optimal alignment score between q_i and c_j
  - Bottom-up strategy to progressively create M

Optimal Global Alignment

$$M_{i,j} = \max \begin{cases} M_{i-1,j} - gap\_score(q_i) & q_i \text{ is aligned with gap} \\ M_{i,j-1} - gap\_score(c_j) & c_j \text{ is aligned with gap} \\ M_{i-1,j-1} + S(q_i, c_j) & q_i \text{ is aligned with } c_j \end{cases}$$

Optimal Global Alignment

$$M_{i,j} = \max \begin{cases} M_{i-1,j} - gap\_score(q_i) & q_i \text{ is aligned with gap} \\ M_{i,j-1} - gap\_score(c_j) & c_j \text{ is aligned with gap} \\ M_{i-1,j-1} + S(q_i, c_j) & q_i \text{ is aligned with } c_j \end{cases}$$

Optimal Global Alignment

$$M_{i,j} = \max \begin{cases} M_{i-1,j} - gap\_score(q_i) & q_i \text{ is aligned with gap} \\ M_{i,j-1} - gap\_score(c_j) & c_j \text{ is aligned with gap} \\ M_{i-1,j-1} + S(q_i, c_j) & q_i \text{ is aligned with } c_j \end{cases}$$

Optimal Global Alignment

$$M_{i,j} = \max \begin{cases} M_{i-1,j} - gap\_score(q_i) & q_i \text{ is aligned with gap} \\ M_{i,j-1} - gap\_score(c_j) & c_j \text{ is aligned with gap} \\ M_{i-1,j-1} + S(q_i, c_j) & q_i \text{ is aligned with } c_j \end{cases}$$

Optimal Global Alignment

$$S(q_i, c_j) = \begin{cases} match\_score(q_i, c_j), & \text{if } q_i = c_j \\ -\ mismatch\_score(q_i, c_j), & \text{otherwise} \end{cases},$$

Optimal Global Alignment

$$S(q_i, c_j) = \begin{cases} match\_score(q_i, c_j), & \text{if } q_i = c_j \\ -\ mismatch\_score(q_i, c_j), & \text{otherwise} \end{cases},$$

Optimal Global Alignment

- **match_score, mismatch_score and gap_score** are fixed
- Ignore two important pieces of information:
  1. Rarity: the alignment of rare methods should have a higher impact on the similarity
  2. Position: first positions are more relevant than the last ones

Optimal Global Alignment

- **match_score, mismatch_score and gap_score** are fixed
- Ignore two important pieces of information:
  1. Rarity: the alignment of rare methods should have a higher impact on the similarity
  2. Position: first positions are more relevant than the last ones

Optimal Global Alignment

- **match_score, mismatch_score and gap_score** are fixed
- Ignore two important pieces of information:
  1. Rarity: the alignment of rare methods should have a higher impact on the similarity
  2. Position: first positions are more relevant than the last ones

Optimal Global Alignment

- **match_score, mismatch_score and gap_score** are fixed
- Ignore two important pieces of information:
  1. Rarity: the alignment of rare methods should have a higher impact on the similarity
  2. Position: first positions are more relevant than the last ones

Our solution

We modify how match_score, mismatch_score and gap_score are computed

Our solution

- For each method in a stack trace (s_k)
  1. Global weight (gw): importance of s_k in a database
  2. Local weight (lw): importance of s_k in a stack trace

Our solution

- For each method in a stack trace (s_k)
  1. Global weight (gw): importance of s_k in a database
  2. Local weight (lw): importance of s_k in a stack trace

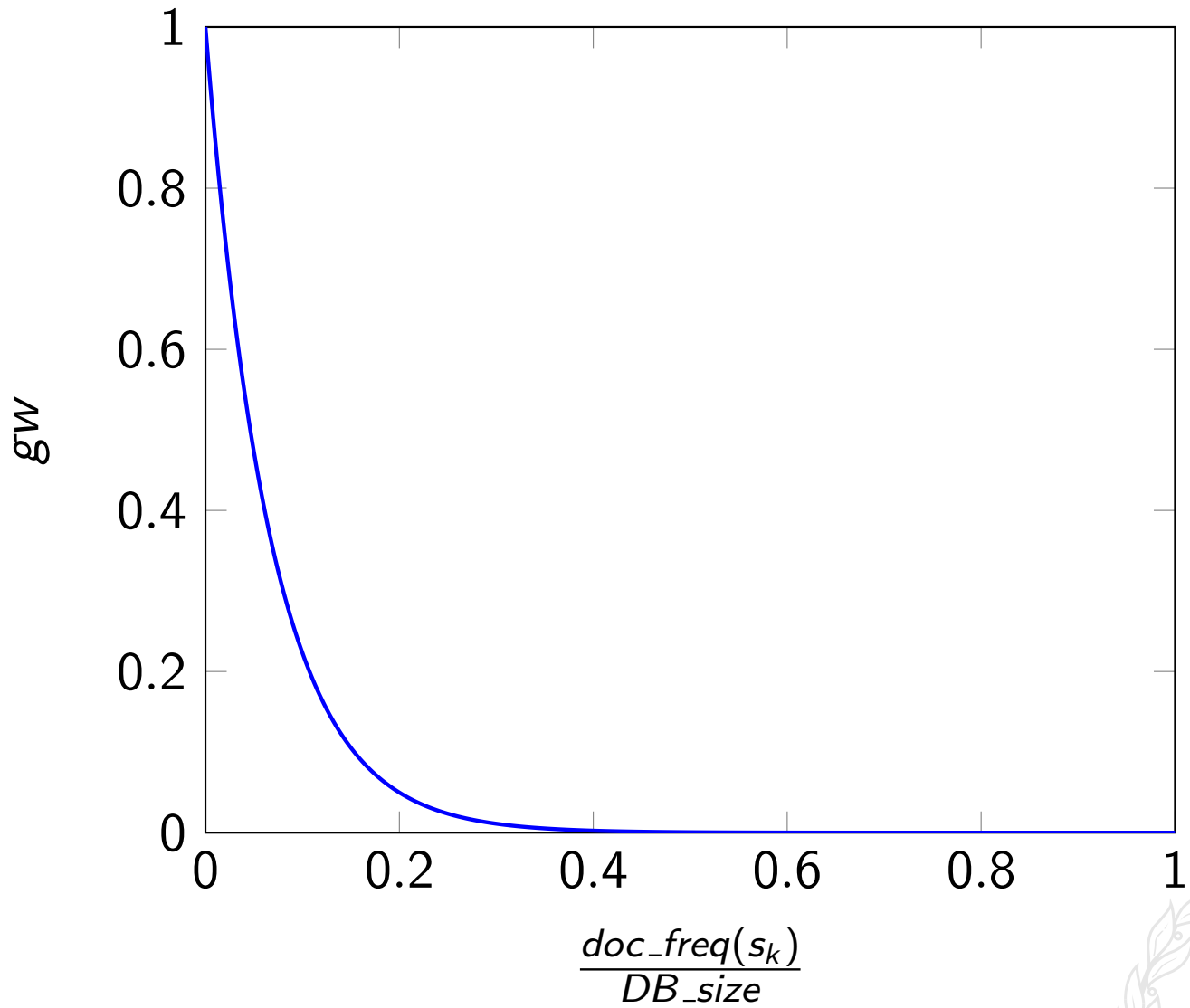
Our solution

- For each method in a stack trace (s_k)
  1. Global weight (gw): importance of s_k in a database
  2. Local weight (lw): importance of s_k in a stack trace

Global weight

$$gw(s_k) = e^{-\beta \frac{doc\_freq(s_k)}{DB\_size}}$$

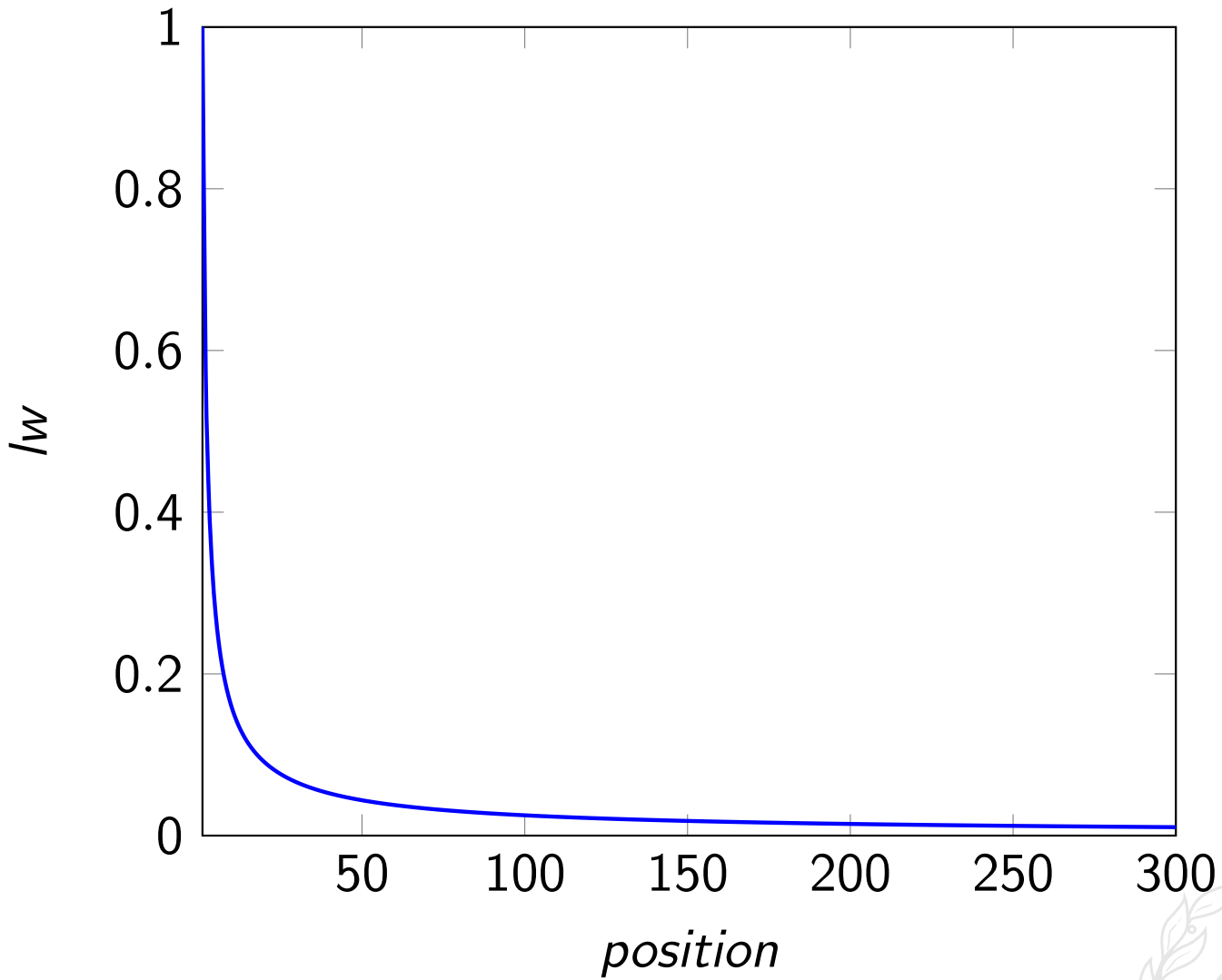
Global weight: $\beta = 15$



Local weight

$$lw(s_k) = k^{-\alpha}$$

Local weight: $\alpha = 0.8$



Frame weight

Frame weight: overall importance of a method

$$w(s_k) = lw(s_k) * gw(s_k)$$

Alignment Costs

- $gap\_cost(s_k) = w(s_k)$
- $mismatch\_cost(q_i, c_j) = w(q_i) + w(c_j)$
- $match\_cost(q_i, c_j) = \max(w(q_i) + w(c_j)) * diff\_pos(i, j)$

Alignment Costs

- $gap\_cost(s_k) = w(s_k)$
- $mismatch\_cost(q_i, c_j) = w(q_i) + w(c_j)$
- $match\_cost(q_i, c_j) = \max(w(q_i) + w(c_j)) * diff\_pos(i, j)$

Alignment Costs

- $gap\_cost(s_k) = w(s_k)$
- $mismatch\_cost(q_i, c_j) = w(q_i) + w(c_j)$
- $match\_cost(q_i, c_j) = \max(w(q_i) + w(c_j)) * diff\_pos(i, j)$

Position Difference

$$diff\_pos(i, j) = e^{-\gamma|i-j|}$$

Compare the matched frames positions

Experiments

- Task: Duplicate crash reports detection
- Datasets: Netbeans, Eclipse, OpenOffice, and Gnome
- 50 different runs
- Metrics
  1. Area Under the ROC Curve (AUC)
  2. Recall Rate@1 (RR@1): percentage of queries whose the first report in the recommend list is the correct duplicate.

Experiments

- Task: Duplicate crash reports detection
- Datasets: Netbeans, Eclipse, OpenOffice, and Gnome
- 50 different runs
- Metrics
  1. Area Under the ROC Curve (AUC)
  2. Recall Rate@1 (RR@1): percentage of queries whose the first report in the recommend list is the correct duplicate.

Experiments

- Task: Duplicate crash reports detection
- Datasets: Netbeans, Eclipse, OpenOffice, and Gnome
- 50 different runs
- Metrics
  1. Area Under the ROC Curve (AUC)
  2. Recall Rate@1 (RR@1): percentage of queries whose the first report in the recommend list is the correct duplicate.

Experiments

- Task: Duplicate crash reports detection
- Datasets: Netbeans, Eclipse, OpenOffice, and Gnome
- 50 different runs
- Metrics
  1. Area Under the ROC Curve (AUC)
  2. Recall Rate@1 (RR@1): percentage of queries whose the first report in the recommend list is the correct duplicate.

Experiments

- Task: Duplicate crash reports detection
- Datasets: Netbeans, Eclipse, OpenOffice, and Gnome
- 50 different runs
- Metrics
  1. Area Under the ROC Curve (AUC)
  2. Recall Rate@1 (RR@1): percentage of queries whose the first report in the recommend list is the correct duplicate.

Experiments

- Task: Duplicate crash reports detection
- Datasets: Netbeans, Eclipse, OpenOffice, and Gnome
- 50 different runs
- Metrics
  1. Area Under the ROC Curve (AUC)
  2. Recall Rate@1 (RR@1): percentage of queries whose the first report in the recommend list is the correct duplicate.

Experiments

- Our method is compared to TF-IDF and Vanilla Optimal Global Alignment
  - Difference: ΔAUC and ΔRR@1
  - Wilcoxon signed-rank test: statistical significance is indicated by $\star$

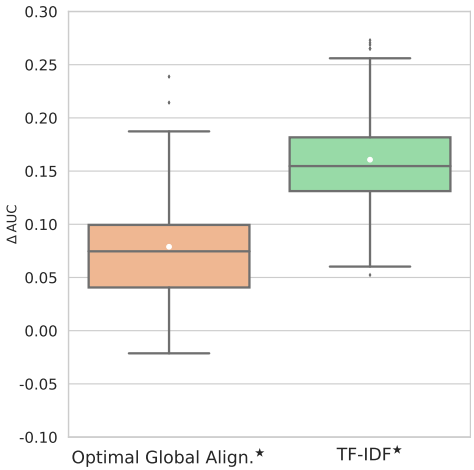
Experiments

- Our method is compared to TF-IDF and Vanilla Optimal Global Alignment
  - Difference: ΔAUC and ΔRR@1
  - Wilcoxon signed-rank test: statistical significance is indicated by $\star$

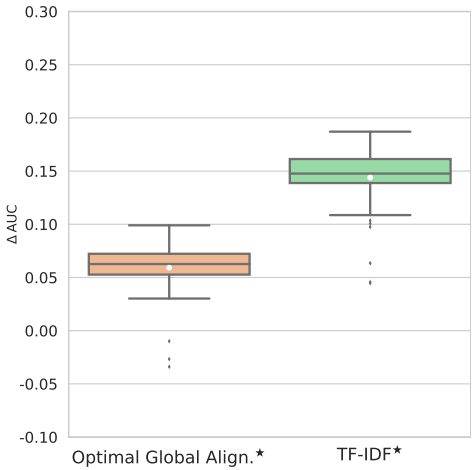
Experiments

- Our method is compared to TF-IDF and Vanilla Optimal Global Alignment
  - Difference: ΔAUC and ΔRR@1
  - Wilcoxon signed-rank test: statistical significance is indicated by $\bigstar$

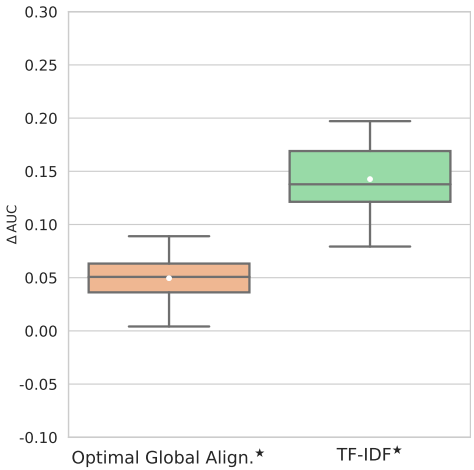
ΔAUC



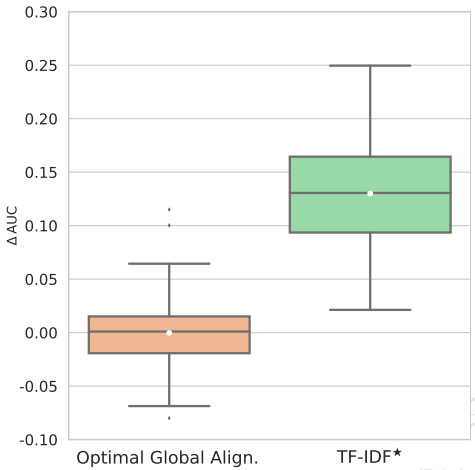
Ubuntu



Eclipse

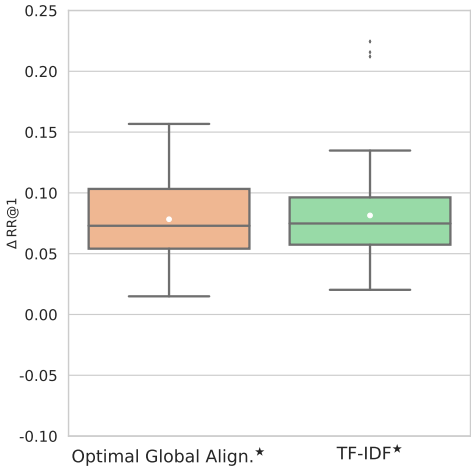


Netbeans

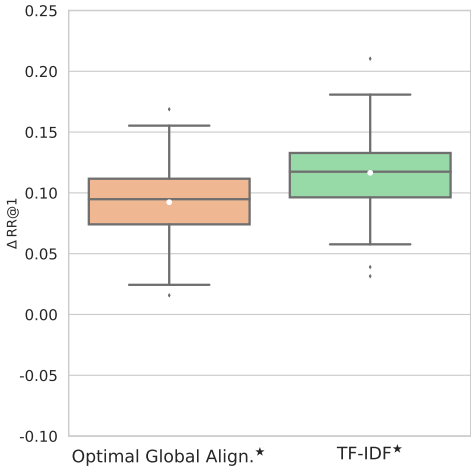


Gnome

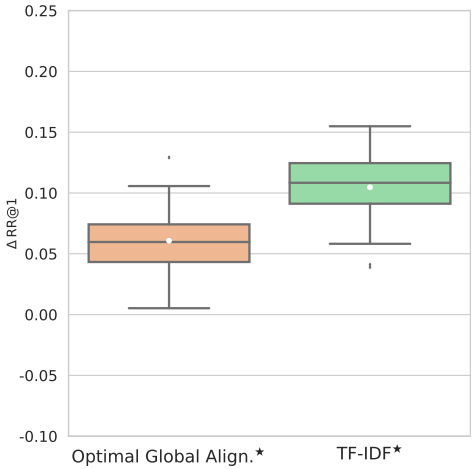
ΔRR@1



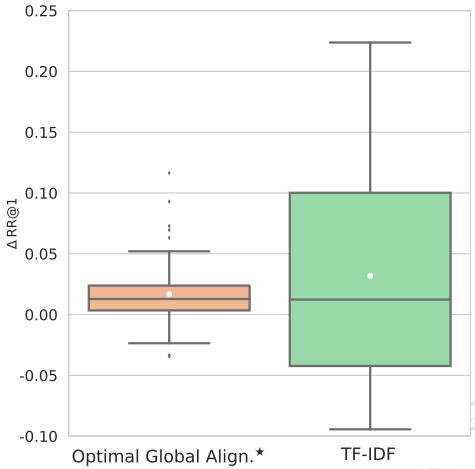
Ubuntu



Eclipse



Netbeans



Gnome

Concluding Remarks

- Our method significantly surpasses Vanilla Optimal Global Alignment and TF-IDF
- We developed a simple scheme to compute the alignment costs
  - 3 parameters
  - Computational cost is not signifincatly increased
- Unsupervised: parameter values might be manually chosen

Concluding Remarks

- Our method significantly surpasses Vanilla Optimal Global Alignment and TF-IDF
- We developed a simple scheme to compute the alignment costs
  - 3 parameters
  - Computational cost is not signifincatly increased
- Unsupervised: parameter values might be manually chosen

Concluding Remarks

- Our method significantly surpasses Vanilla Optimal Global Alignment and TF-IDF
- We developed a simple scheme to compute the alignment costs
  - 3 parameters
  - Computational cost is not signifincatly increased
- Unsupervised: parameter values might be manually chosen

Concluding Remarks

- Our method significantly surpasses Vanilla Optimal Global Alignment and TF-IDF
- We developed a simple scheme to compute the alignment costs
  - 3 parameters
  - Computational cost is not signifincatly increased
- Unsupervised: parameter values might be manually chosen

Concluding Remarks

- Our method significantly surpasses Vanilla Optimal Global Alignment and TF-IDF
- We developed a simple scheme to compute the alignment costs
  - 3 parameters
  - Computational cost is not signifincatly increased
- Unsupervised: parameter values might be manually chosen

Future works

- Make our method available for our partners
  - Should we integrate it to a software?
  - Inputs and outputs?
- Please let us know how our method could be useful for your project!

Thank you for your attention!

Questions?