Detecting Near-duplicate States in Web App Model Inference: a Tree Kernel-based Approach

Luigi Libero Lucio Starace

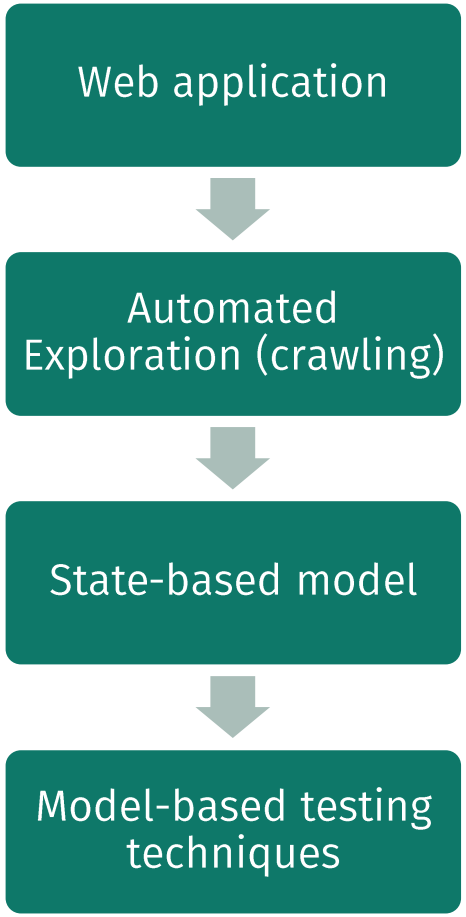
Università degli Studi di Napoli Federico II, Naples, Italy

luigiliberolucio.starace@unina.it

July 13, 2021

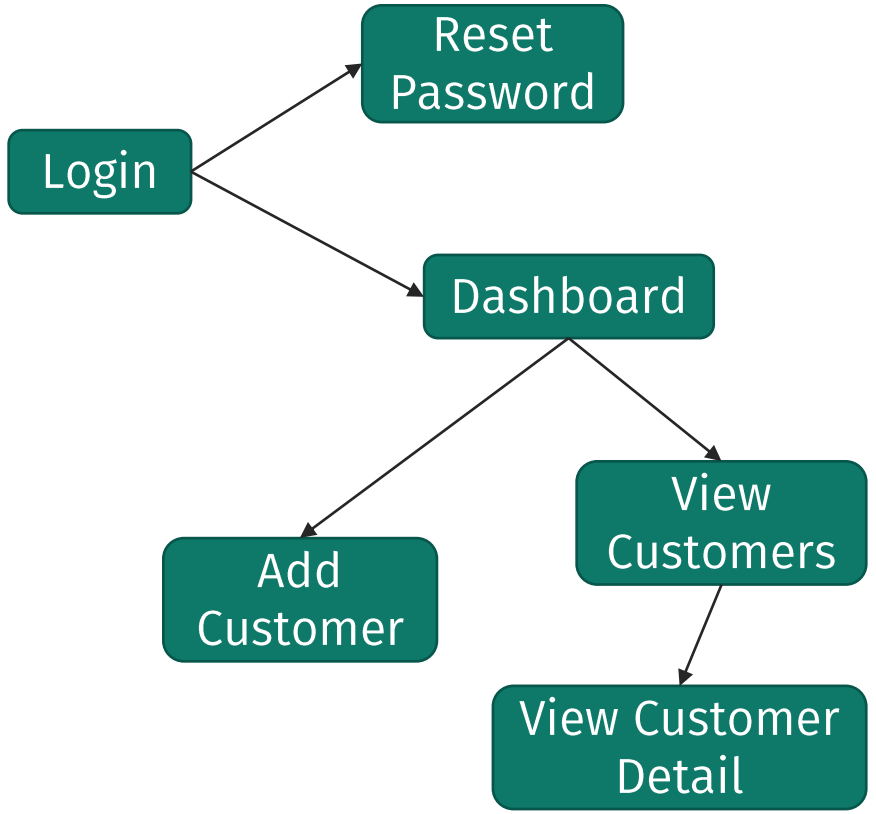
E2E testing of web applications

- Test the web application from the perspective of end-users.
- **Model-based** techniques can be used to support E2E testing
  - test case generation
  - test artifact generation
- **Crawling** is widely-used to **infer** these state-based models



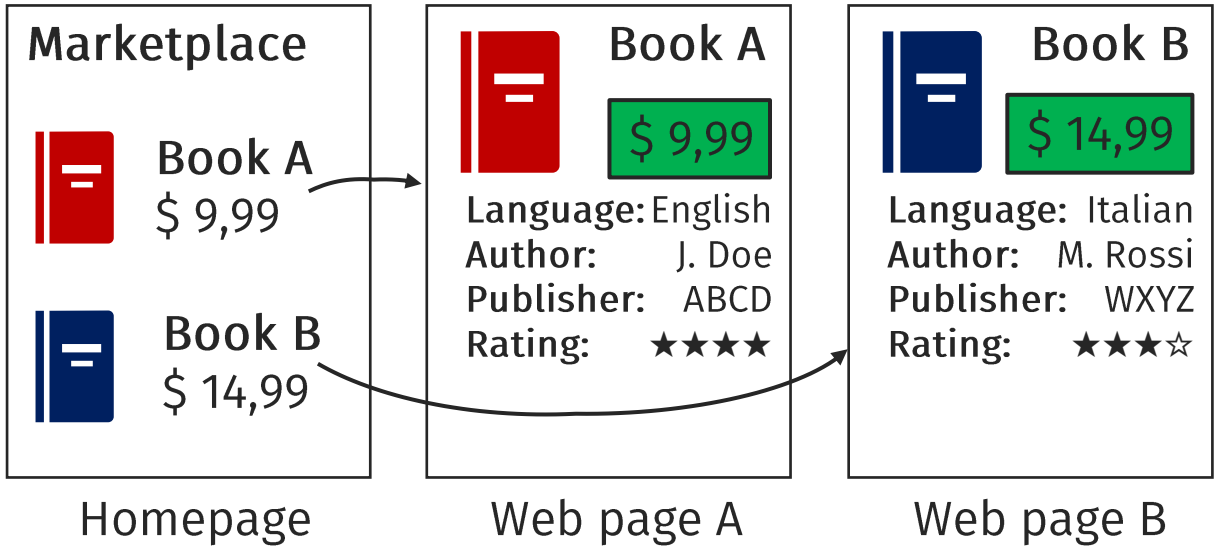
State-based web application models

- Each **state** represents a **feature**
- **Transitions** represent **navigation relations** between states



The near-duplicate problem

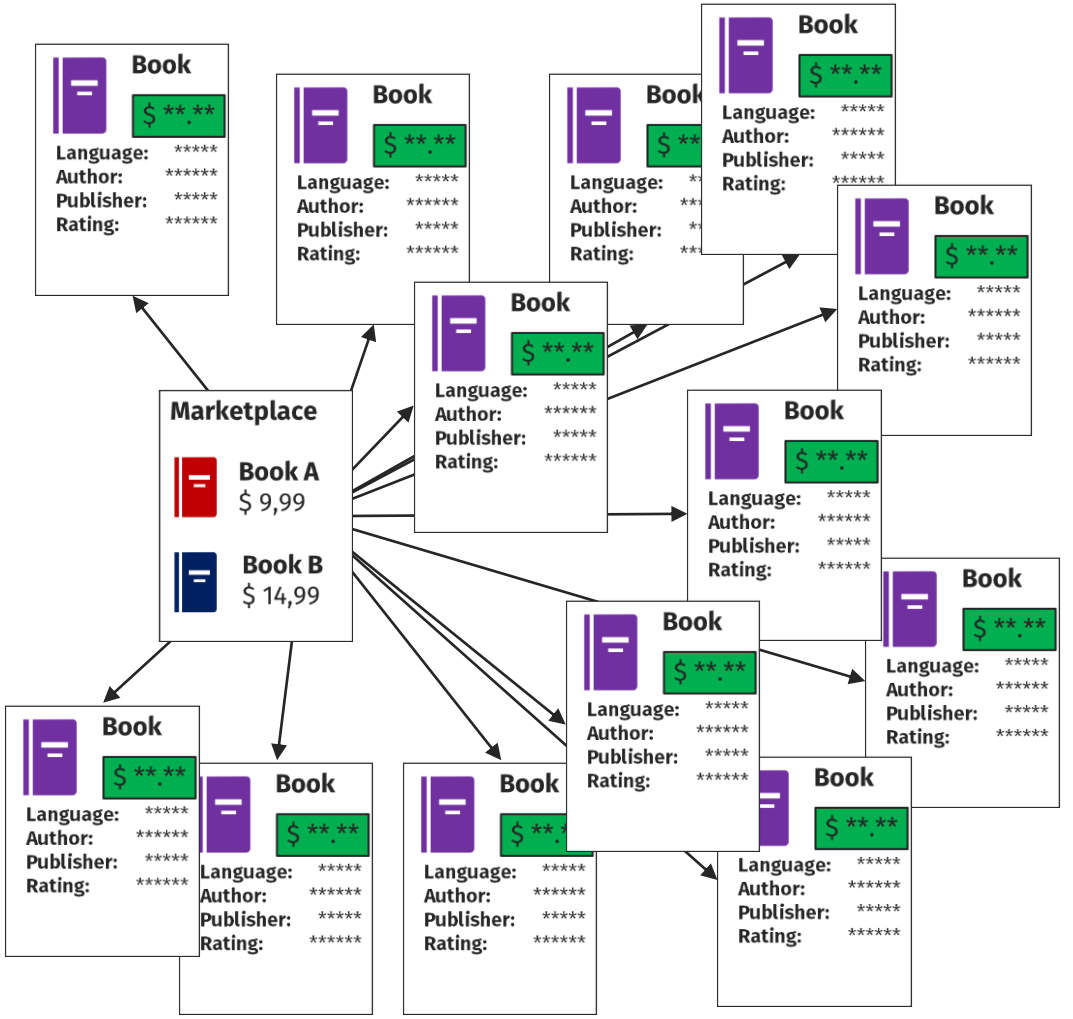
- In practice, models obtained by crawling are affected by **near-duplicates**[1].
- States corresponding to different web pages that represent the **same functionality**.
- It's **hard** for crawlers to **detect** near-duplicate states!



[1] Yandrapally et al. "*Near-duplicate detection in web app model inference.*" *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. 2020.

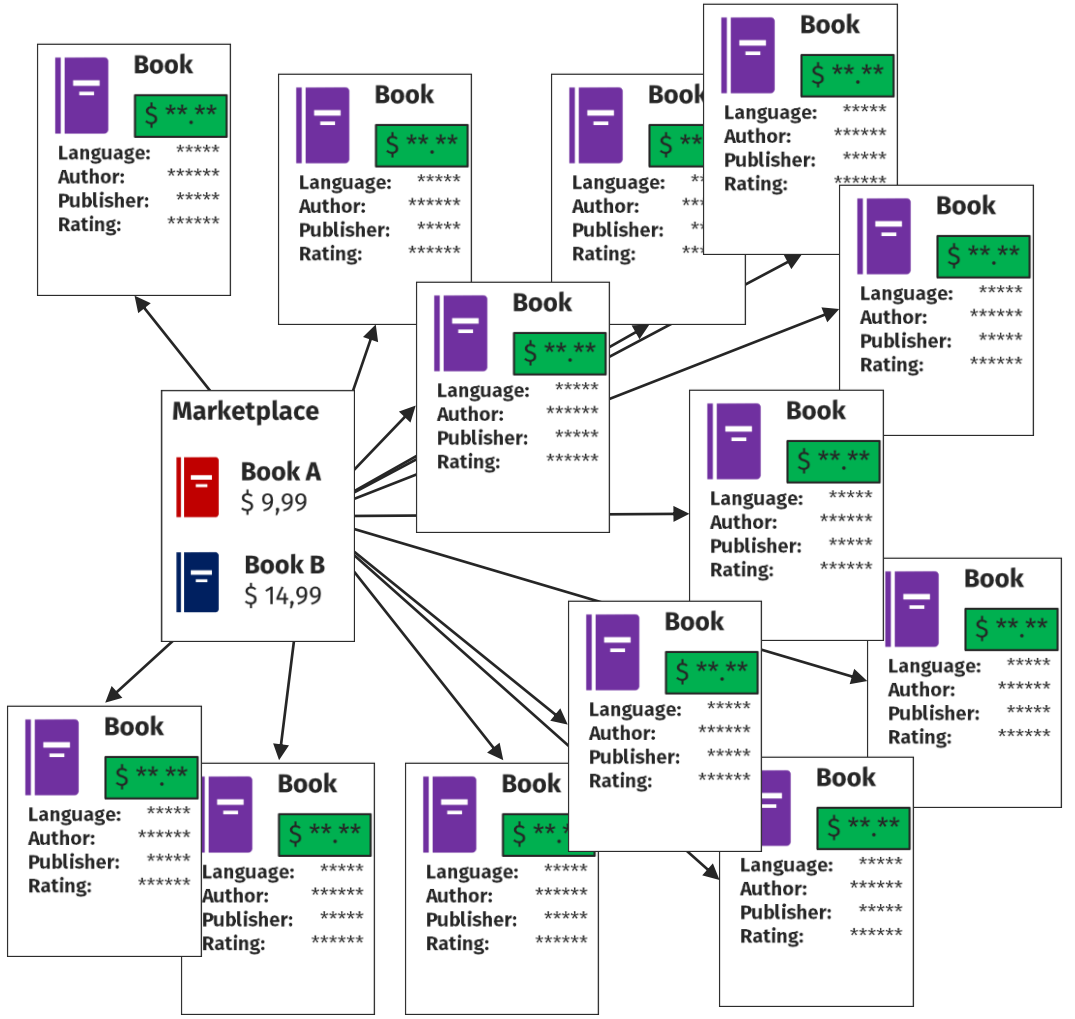
The near-duplicate problem

- Near-duplicates negatively affect the **precision** and **completeness** of the model
- This significantly **hinders** the application of model-based testing techniques



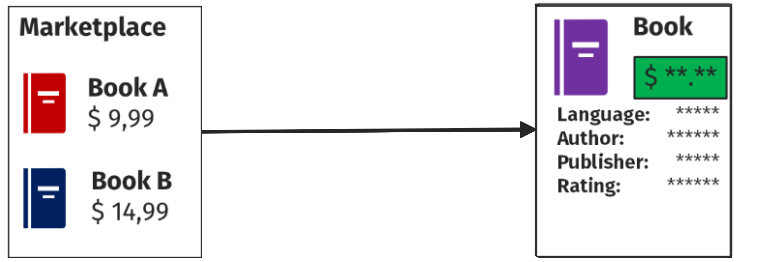
The near-duplicate problem

- Detecting near-duplicates can help crawlers infer **better** models
- Near-duplicates web pages should be mapped to the same «logical page»



The near-duplicate problem

- Detecting near-duplicates can help crawlers infer **better** models
- Near-duplicates web pages should be mapped to the same «logical page»



Catalog

Book detail page

Related Works

The problem of near-duplicate/clone web page detection arises in different domains:

- **Information retrieval/indexing**
  - Typically focus on content rather than structure or looks
  - Mostly use content hashing techniques (e.g., simhash)
- **Phishing detection**
  - Typically focus on the looks of web pages
  - Often use visual-based approaches (e.g., perceptual hash, color histograms, …)

No approaches specifically geared towards model inference for testing!

State of the art

A first study on near-duplicate detection for model inference was presented by Yandrapally et Al. at ICSE20[1]. In that work:

- 10 state-of-the-art near-duplicate detection techniques from different domains are evaluated on the model inference task
- A **massive dataset** of approx. 100k manually annotated web page pairs is made available

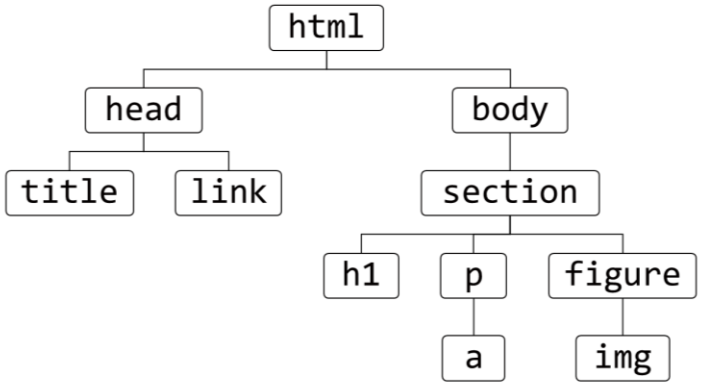
The **goal** of my Ph.D. is to improve test effectiveness for web applications by designing **novel/specific similarity measures** to improve model inference.

[1] Yandrapally et al. "*Near-duplicate detection in web app model inference.*" *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. 2020.

A first approach for near-duplicate detection

Web pages and their DOM representation


```
<html>
  <head>
    <title></title>
    <link/>
  </head>
  <body>
    <section>
      <h1></h1>
      <p><a></a></p>
      <figure>
        <img/>
      </figure>
    </section>
  </body>
</html>
```

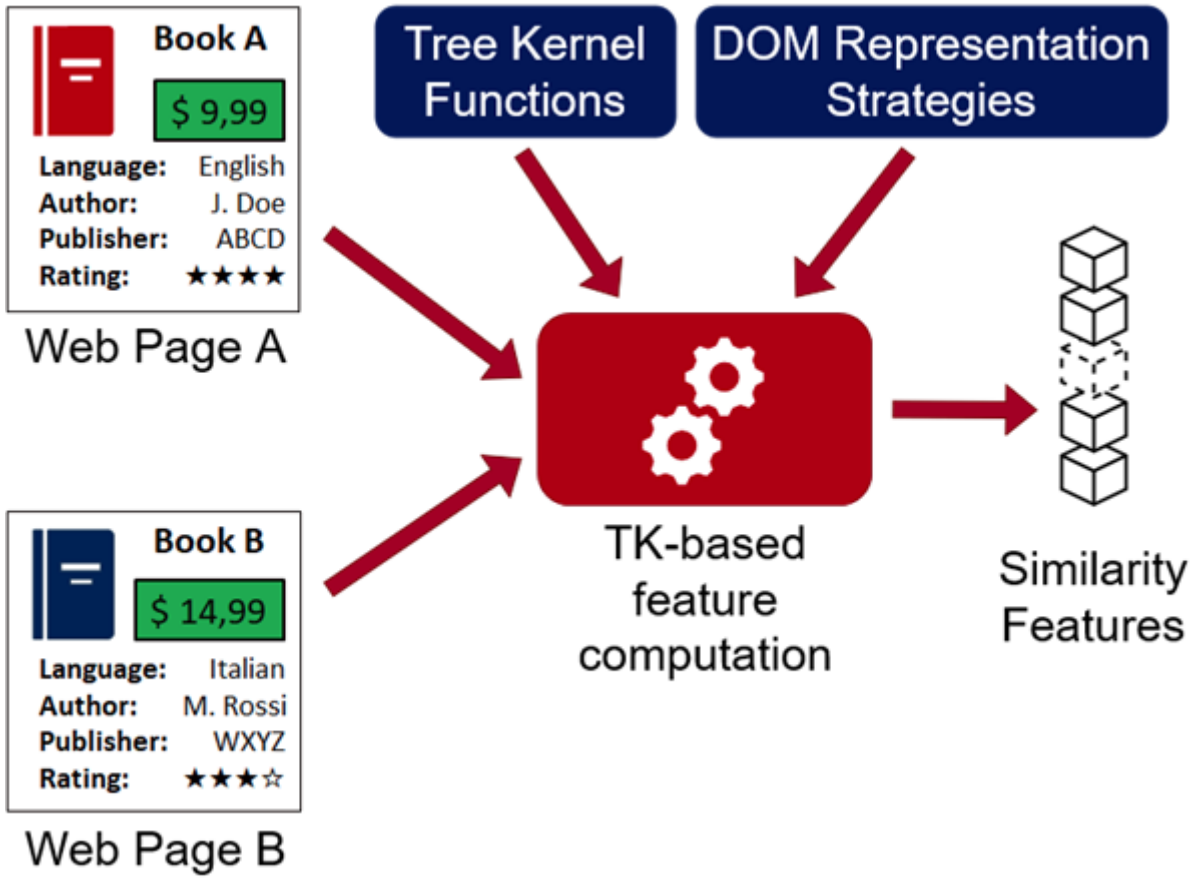


Web Page

HTML code

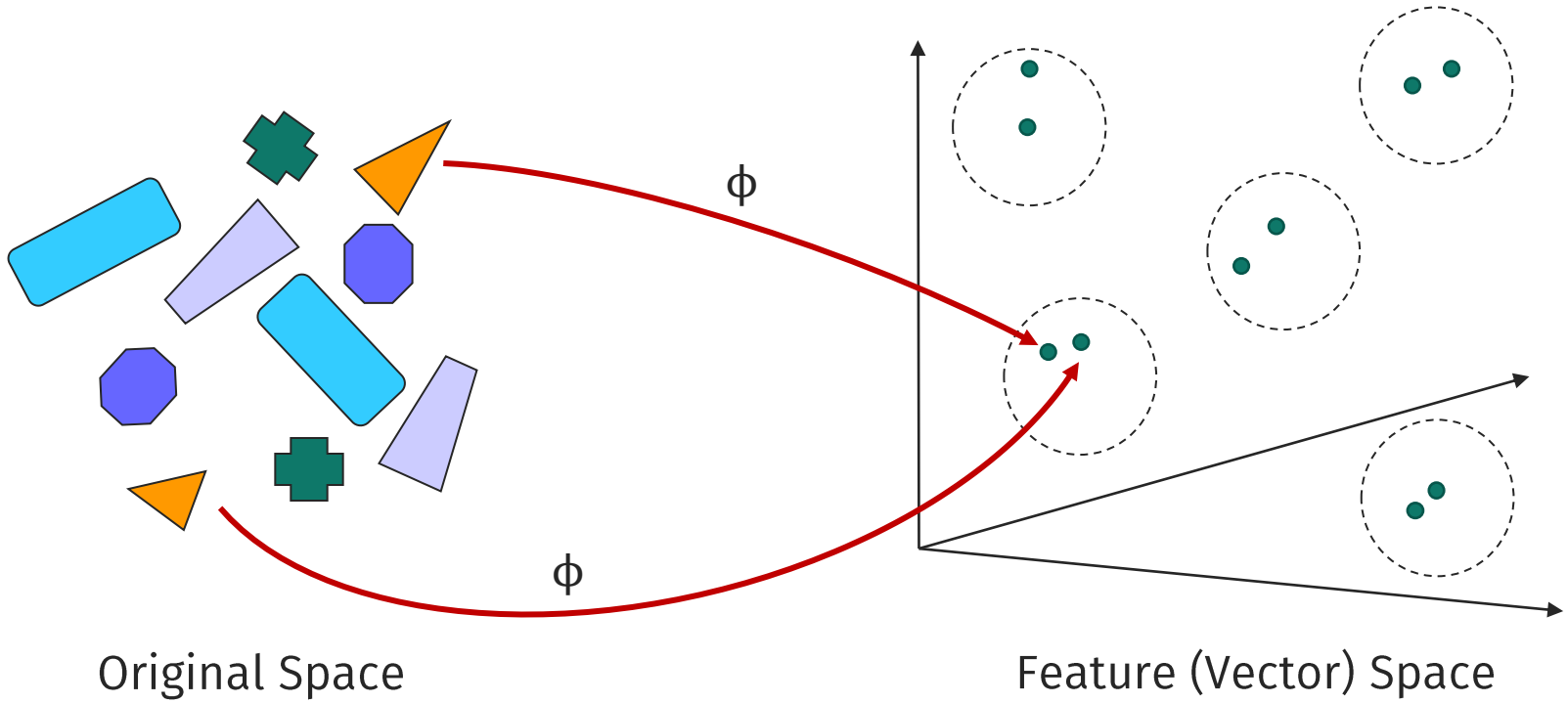
DOM Representation

Tree Kernel-based similarity features



Kernel Methods: intuition

- Find a mapping ϕ such that, in the new space, problem solving is easier (e.g. linear)



Kernel Functions

- A **kernel function** k represents the **similarity** between two (possibly structured) objects, defined as the dot-product in this new vector space
- The mapping might remain **implicit**

$$k(x, y) = \phi(x) \cdot \phi(y)$$

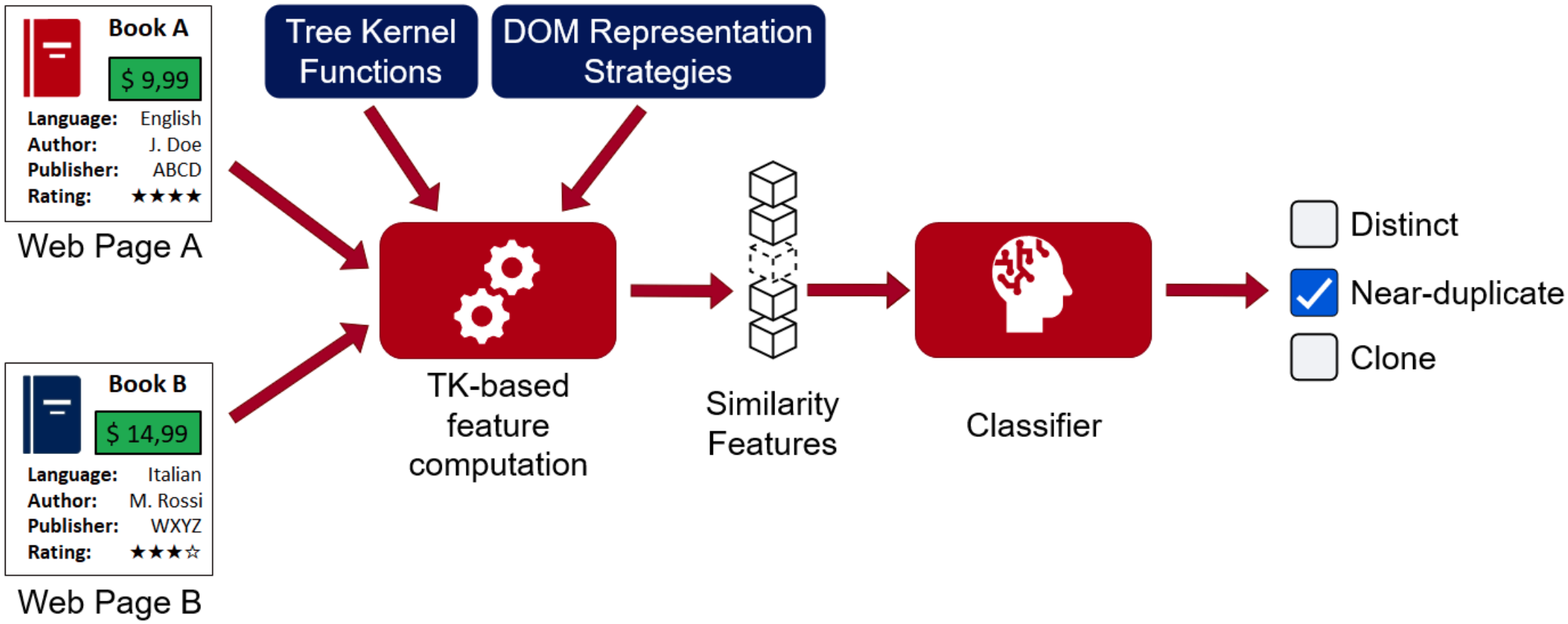
Tree Kernels (TKs)

- Kernel functions specifically designed to work on trees
- Largely and effectively used in **NLP**
- Idea: similarity between two trees depends on the number of **fragments** *(subsets of nodes and edges)* they share
- Different definitions of «fragments» lead to different TKs
- We are currently considering 3 tree kernel functions:
  - Subtree Kernels
  - Subset-Tree Kernels
  - Partial Tree Kernels

DOM Representation Strategies

- Ways to **pre-process** the DOM of the web pages (e.g. to remove useless parts)
- Three DOM representation strategies at the moment:
  - As-is: leaves DOM unchanged
  - Removes `<script>` elements
  - Keeps only the `<body>` element

Tree Kernel-based near-dupes detection



Experimental Procedure

We're currently replicating the procedure in Yandrapally et Al.'s ICSE20 paper:

- Near-duplicate detection framed as a **classification task**

Leveraging their dataset to **train** and **test** our classifier.

- ~100k pairs of annotated **same-website** page pairs
- Both from **real-world** websites and from **open source applications** in a controlled environment

Preliminary Results

- Outperform the best technique from [ICSE20] (PDiff) by **3%**
- PDiff is a computationally expensive visual-based technique
- Compared with similar, DOM-based techniques, improvement ranges from approx. **10%** to **30%**.

Technique	Average
PDiff	0.60
BlockHash	0.58
SSIM	0.57
Levenshtein	0.54
RTED	0.54
SIFT	0.54
pHASH	0.52
TLSH	0.50
Color-histogram	0.44
simhash	0.33
TK-based SVM	0.63

Macro-averaged F1

Future research road map

The goal of my Ph.D. is to improve test effectiveness for web applications by designing novel/specific similarity measures to improve model inference.

Future research I plan to carry on includes:

- Extending this first approach with more refined similarity features (e.g.: subpath kernels, or specifically tailored Tree Kernel functions)
- Investigating whether content hashing techniques such as simhash can be more effective when pre-processing the DOMs
- Leveraging deep learning/embeddings

Future research road map (assessment)

The **key goal** is to assess the effectiveness of the techniques on the **real** task at hand, i.e. **Model Inference to support E2E testing.**

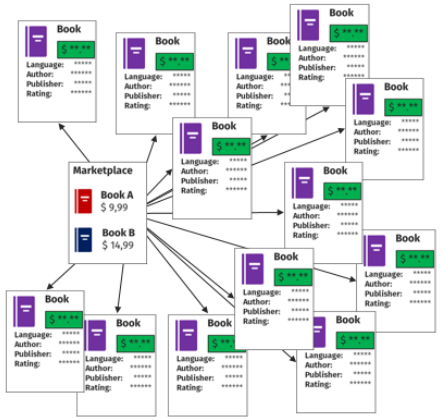
- Implement the novel similarity measure in a Crawler
- Use the hand-made **gold standard** models for different open source web applications made available by the ICSE20 paper
- The original dataset could also be extended by manually annotating more web page pairs from new websites

Detecting Near-duplicates in Web App Model Inference: a Tree Kernel-based Approach

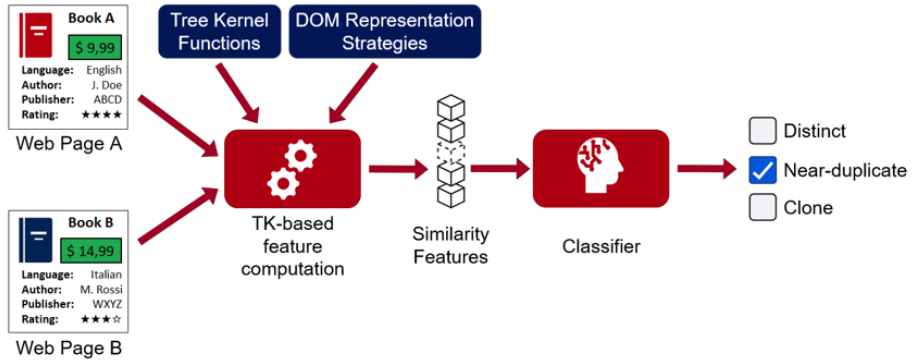
Luigi Libero Lucio Starace
luigiliberolucio.starace@unina.it

The near-duplicate problem

- Near-duplicates negatively affect the **precision** and **completeness** of the model
- This significantly **hinders** the application of model-based techniques



Tree Kernel-based near-dupes detection



Preliminary Results

- Outperform the best technique from [ICSE20] (PDiff) by **3%**
- PDiff is a computationally expensive visual-based technique
- Compared with similar, DOM-based techniques, improvement ranges from approx. **10%** to **30%**.

Technique	Average
PDiff	0.60
BlockHash	0.58
SSIM	0.57
Levenshtein	0.54
RTED	0.54
SIFT	0.54
pHASH	0.52
TLSH	0.50
Color-histogram	0.44
simhash	0.33
TK-based SVM	0.63

Macro-averaged F1

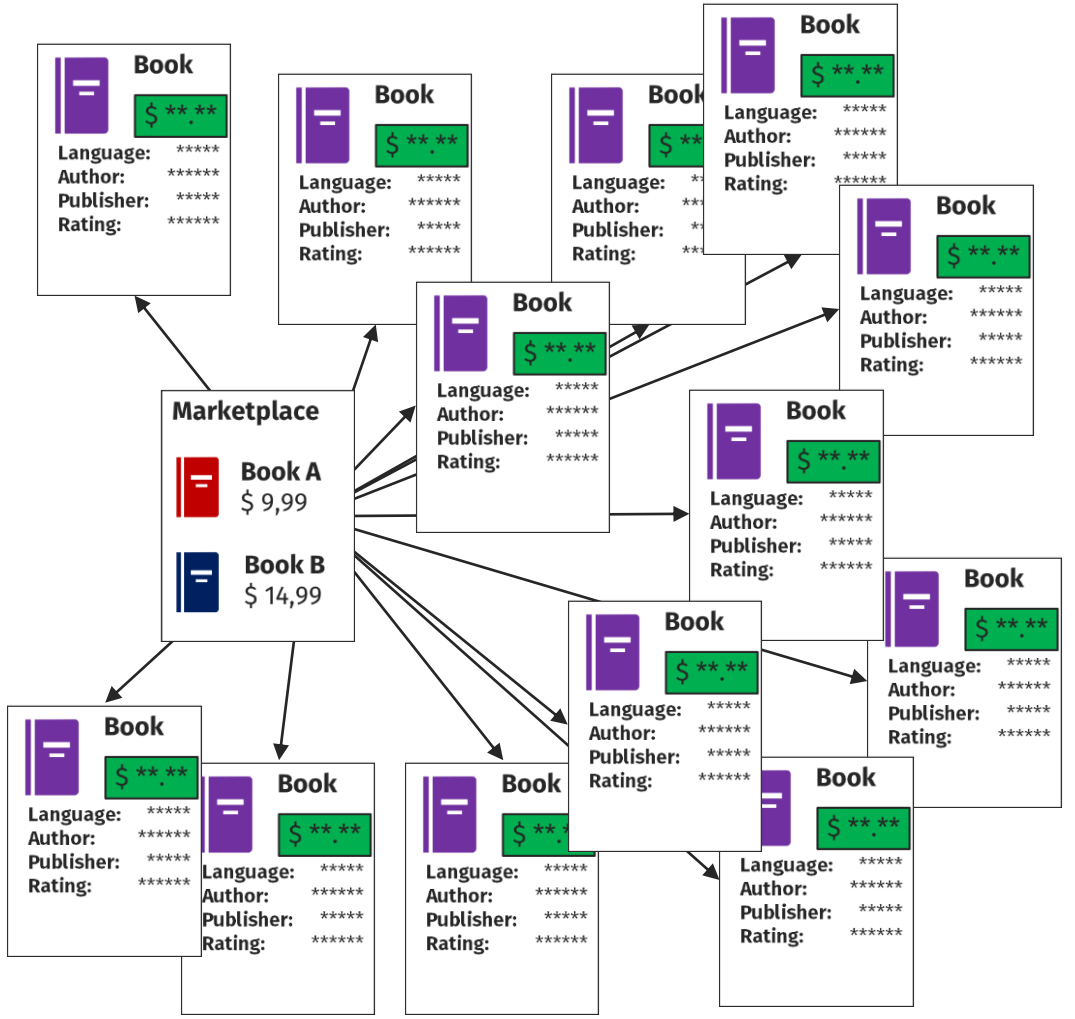
Future research road map

The **goal** of my Ph.D. is to study novel **similarity** measures for web pages. Future research I plan to carry on includes:

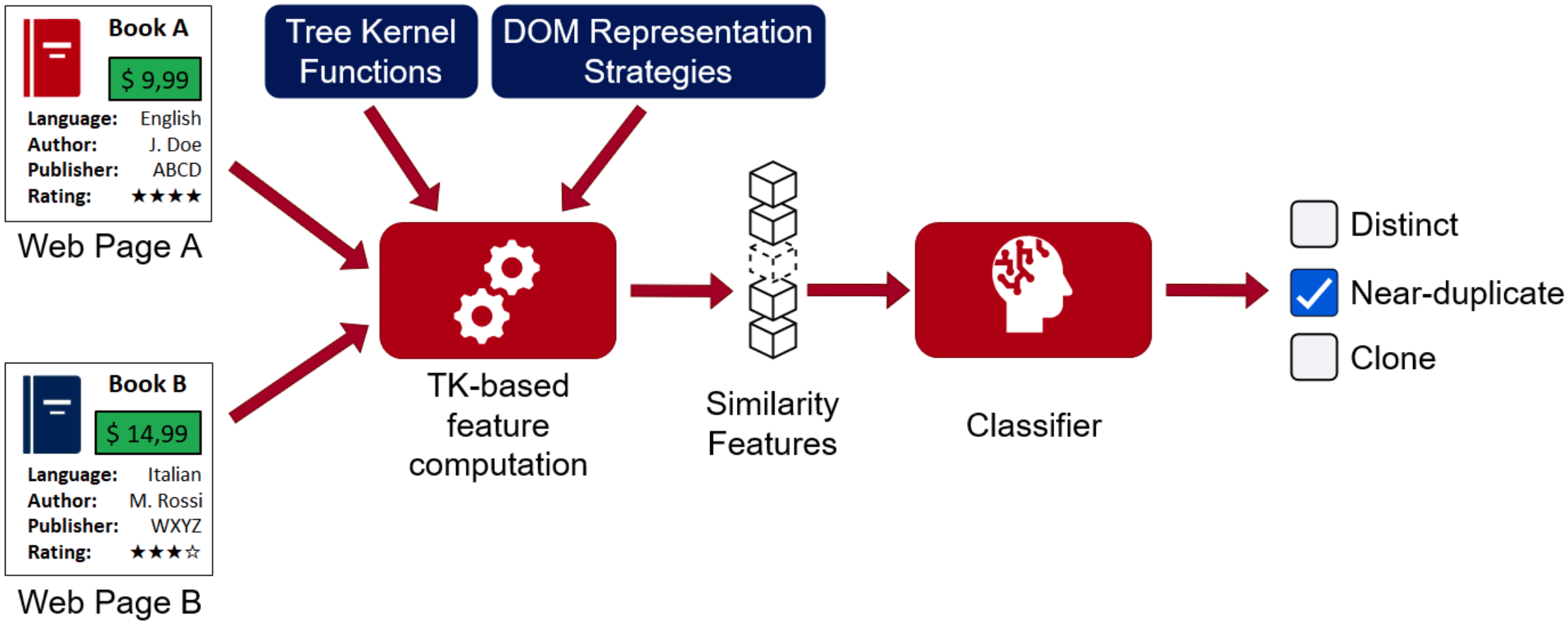
- **Extending** this first approach with more refined similarity features (e.g.: subpath kernels)
- Designing a **novel Tree Kernel function** for the task at hand
- Investigating whether content hashing techniques such as simhash can be more effective when pre-processing the DOMs
- Leveraging **deep learning/embeddings**

The near-duplicate problem

- Near-duplicates negatively affect the **precision** and **completeness** of the model
- This significantly **hinders** the application of model-based techniques



Tree Kernel-based near-dupes detection



Preliminary Results

- Outperform the best technique from [ICSE20] (PDiff) by **3%**
- PDiff is a computationally expensive visual-based technique
- Compared with similar, DOM-based techniques, improvement ranges from approx. **10%** to **30%**.

Technique	Average
PDiff	0.60
BlockHash	0.58
SSIM	0.57
Levenshtein	0.54
RTED	0.54
SIFT	0.54
pHASH	0.52
TLSH	0.50
Color-histogram	0.44
simhash	0.33
TK-based SVM	0.63

Macro-averaged F1

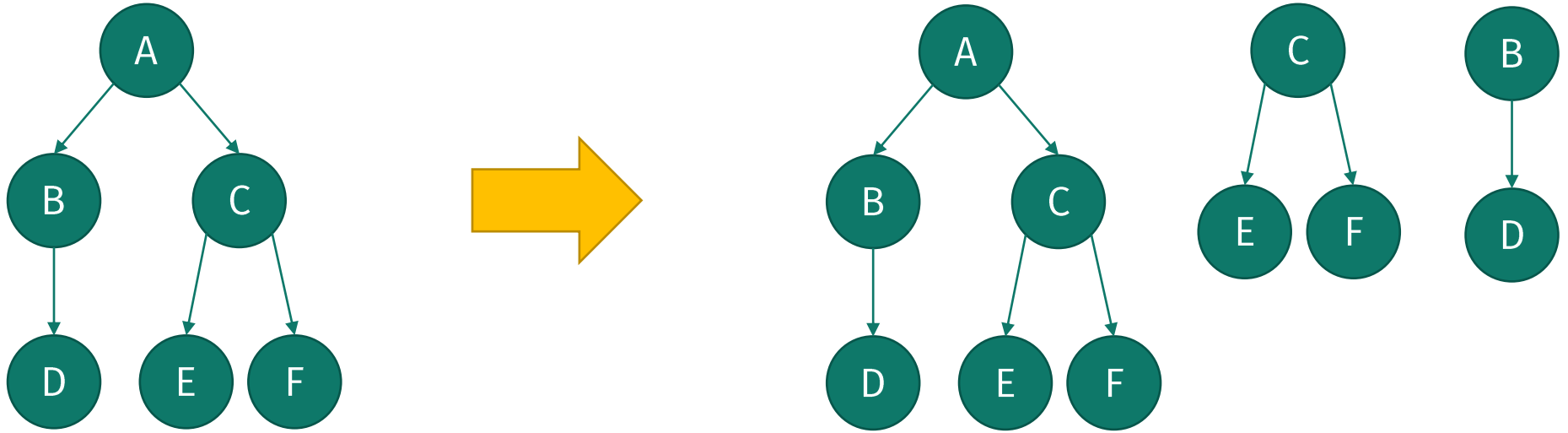
Future research road map

The goal of my Ph.D. is to study novel similarity measures for web pages. Future research I plan to carry on includes:

- Extending this first approach with more refined similarity features (e.g.: subpath kernels)
- Designing a novel Tree Kernel function for the task at hand
- Investigating whether content hashing techniques such as simhash can be more effective when pre-processing the DOMs
- Leveraging deep learning/embeddings

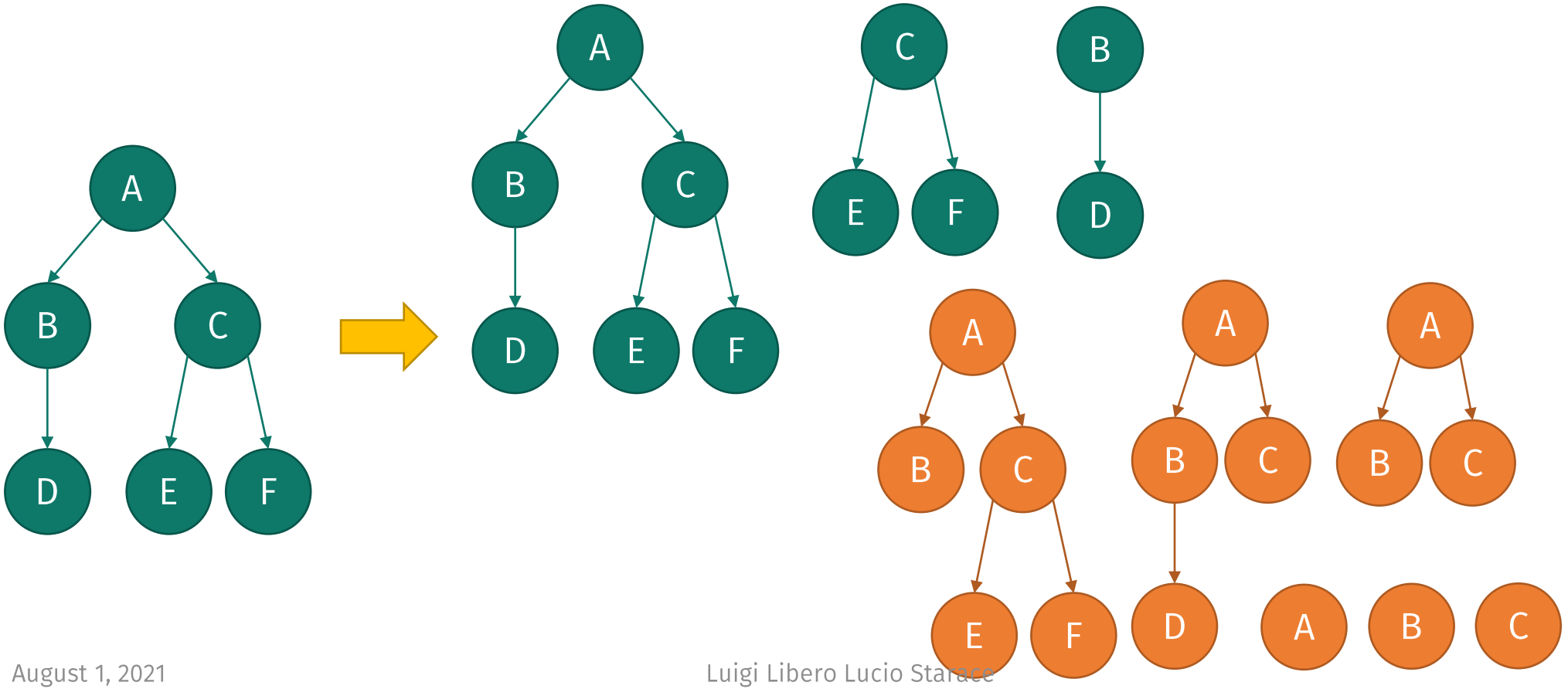
Subtree Kernels

Consider as fragments only proper subtrees (i.e., if a node belongs to a fragment, then so do all of its descendants)



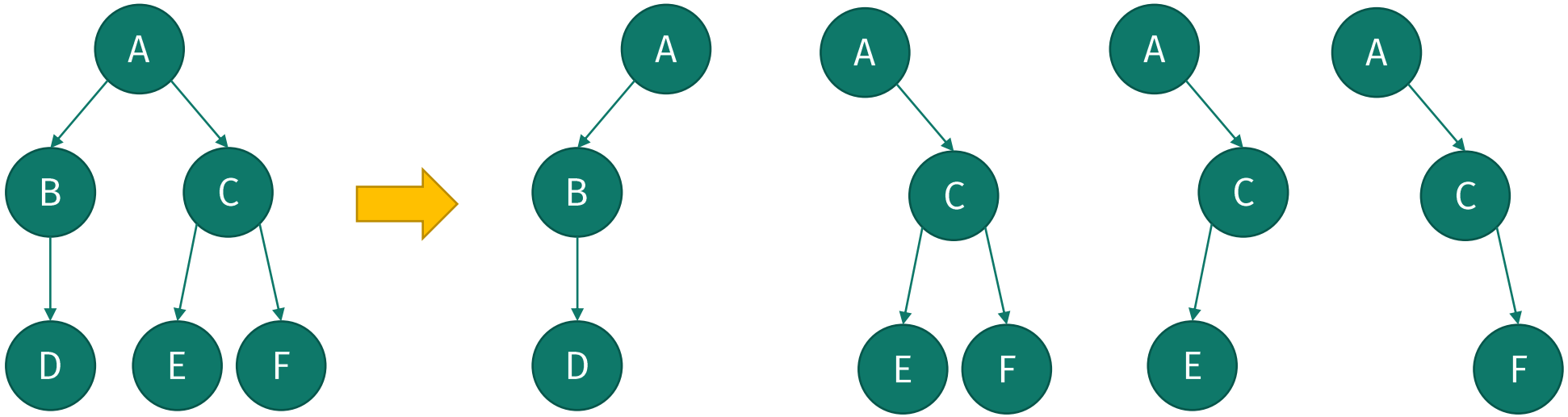
Subset-Tree Kernels

Relax the constraints of taking all the descendants of a node



Partial Tree Kernels

Relax the constraint of taking all the children of a node



Preliminary Results

- Outperform PDiff by **5%** and **1%**
- Pdiff is a computationally expensive visual-based technique
- Compared with similar, DOM-based techniques, improvement ranges from approx. **10%** to **40%**.

Technique	$\mathcal{SS}$	$\mathcal{TS}$	Average
PDiff	0.53	0.67	0.60
BlockHash	0.54	0.62	0.58
SSIM	0.53	0.62	0.57
Levenshtein	0.48	0.59	0.54
RTED	0.50	0.57	0.54
SIFT	0.47	0.61	0.54
pHASH	0.40	0.63	0.52
TLSH	0.44	0.56	0.50
Color-histogram	0.37	0.52	0.44
simhash	0.17	0.48	0.33
TK-based SVM	0.58	0.68	0.63

Macro-averaged F1