

# Identifying Features of Android Apps from Execution Traces

Qi Xin, Farnaz Behrang, Mattia Fazzini, and Alessandro Orso



# Understanding a Program & its Features



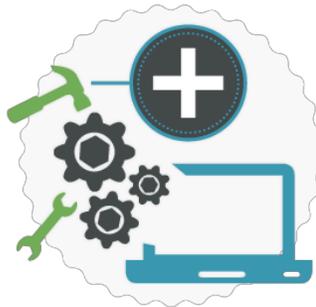
**Refactoring**



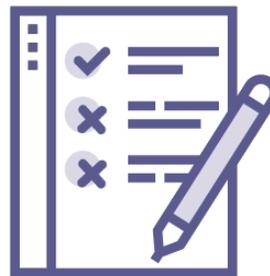
**Debloating**



**Debugging**



**Functionality  
Modification**



**Testing**



**Documentation**

# Understanding a Program & its Features

**Identifying Features of  
a program by Analyzing  
its Executions**





# Program Understanding is HARD

```
...a.fn.scrollspy=d,this),a(window).on("load",function(){...  
...y),+function(a){"use strict";function b(b){return this.each(function(){var d=a...  
...e[b]({}))var c=function(b){this.element=a(b)};c.VERSION="3.3.7",c.TRANSITION_DURATION=150,c.pro...  
...topdown-menu"),d=b.data("target");if(d||(d=b.attr("href"),d=d&&d.replace(/.*(?=#[^\s]*$)/,"")),!...  
...st a"),f=a.Event("hide.bs.tab",{relatedTarget:b[0]}),g=a.Event("show.bs.tab",{relatedTarget:e[0]}...  
...faultPrevented()){var h=a(d);this.activate(b.closest("li"),c),this.activate(h,h.parent(...  
...rigger({type:"shown.bs.tab",relatedTarget:e[0]}))}}},c.prototype.activate=function(a,b,c){...  
...u > .active").removeClass("active").end().find("[data-toggle="tab"]')...  
...ia-expanded",!0),h?(b[0].offsetWidth,b.addClass("in")):b.remove...  
...().find("[data-toggle="tab"]').attr("aria-expanded"...  
...e")||!d.find("> .fade").length);g.length&&g...  
...;var d=a.fn.tab;a.fn.tab=b,a.fn.tab.show=function(a){...  
...show");a(document).on("click.bs.tab",function(e){...  
...se strict";function b(b){return this.each(function(){var d=a...  
...typeof b&&e[b]({}))var c=function(b){this.element=a(b)};c...  
...a.proxy(this.checkPosition,this),this.pinnedOffset=null,...  
...null,this.pinnedOffset=null,this.cheat=function(a,b,c,d){var e=this.$...  
..."bottom"==this.affixed)return null!=c&&e["top"]<e["bottom"]...  
...!<c&&e["top"]<e["bottom"]&&e["bottom"]<e["top"]&&e["top"]<e["bottom"]...  
...RESET).addClass("affix");var a=this.$element;this.affix...  
...withEventLoop=function(){setTimeout(a.proxy(this.affix,...  
...ent.height(),d=this.options.offset,e=d.top,d=this.options...  
...peof e&&(e=d.top(this.$element))&&e["top"]<e["bottom"]&&e["...  
...ent.css("top","auto");return this.affix("top");}});c(...
```

**~50% maintenance effort spent on program understanding alone\***



\*Program understanding: Challenge for the 1990s by Corbi

Figure from Understanding Execution Traces Using Massive Sequence and Circular Bundle Views by Cornelissen et al.

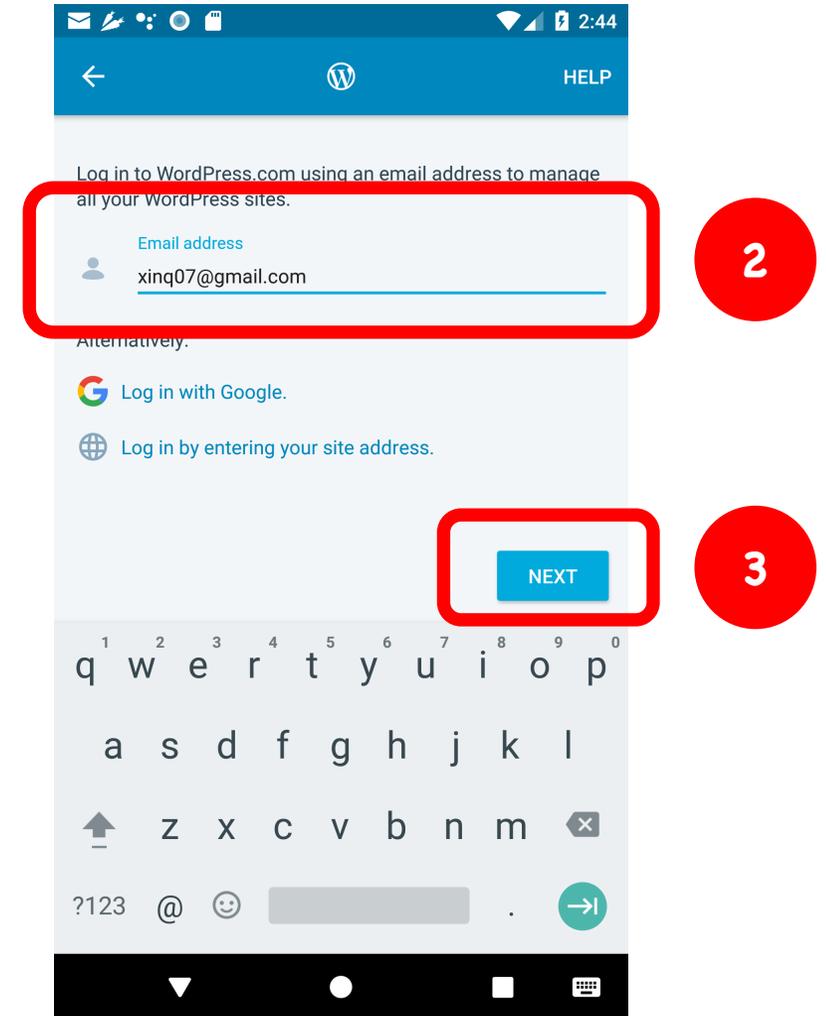
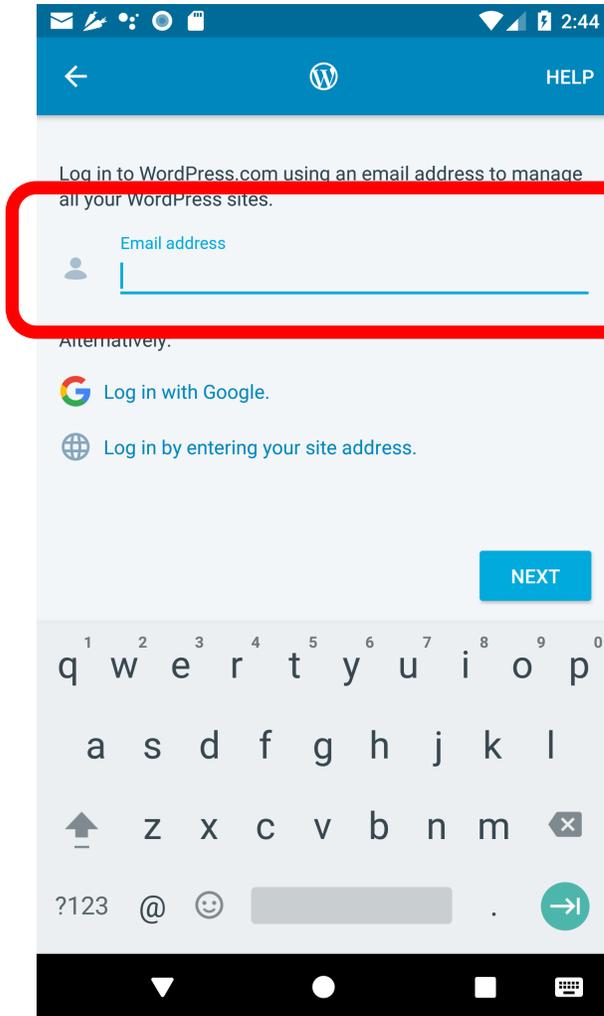
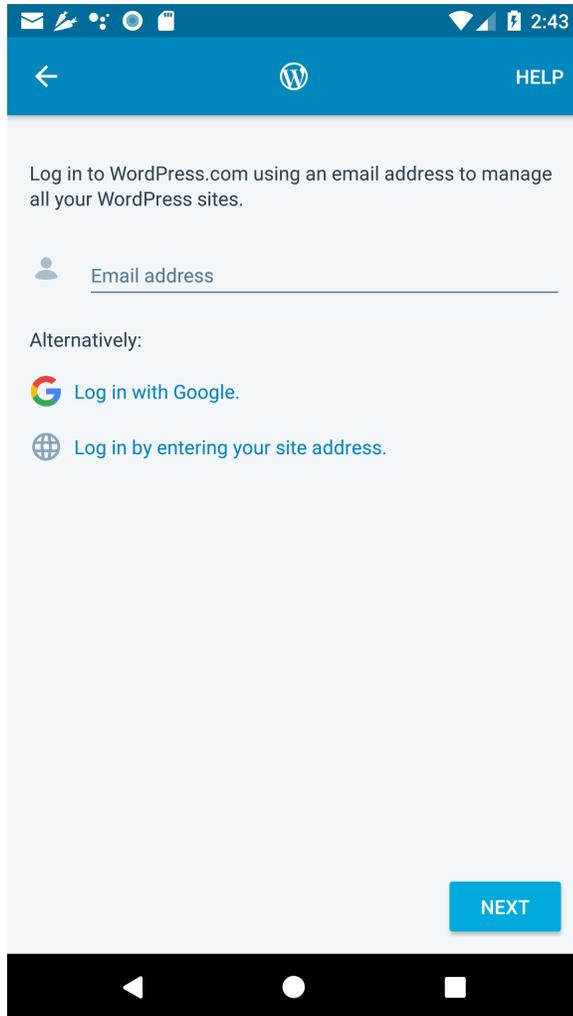
# Our Approach

- Identifies features by analyzing execution trace
- Targets Mobile (Android) apps
- In our context, a feature is a sequence of user events that exercise some functionality of the app

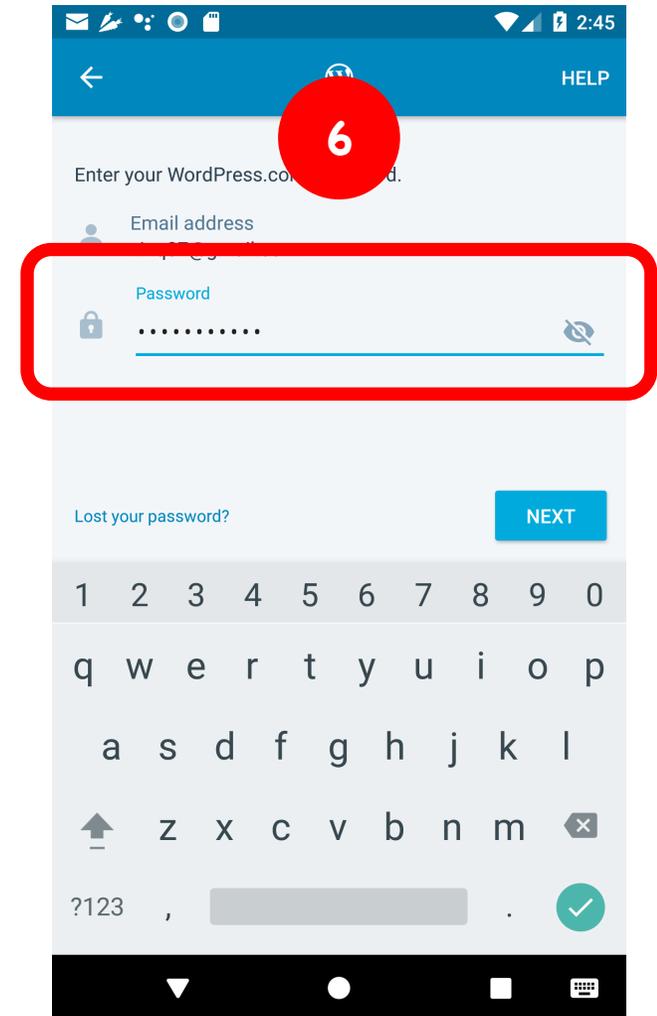
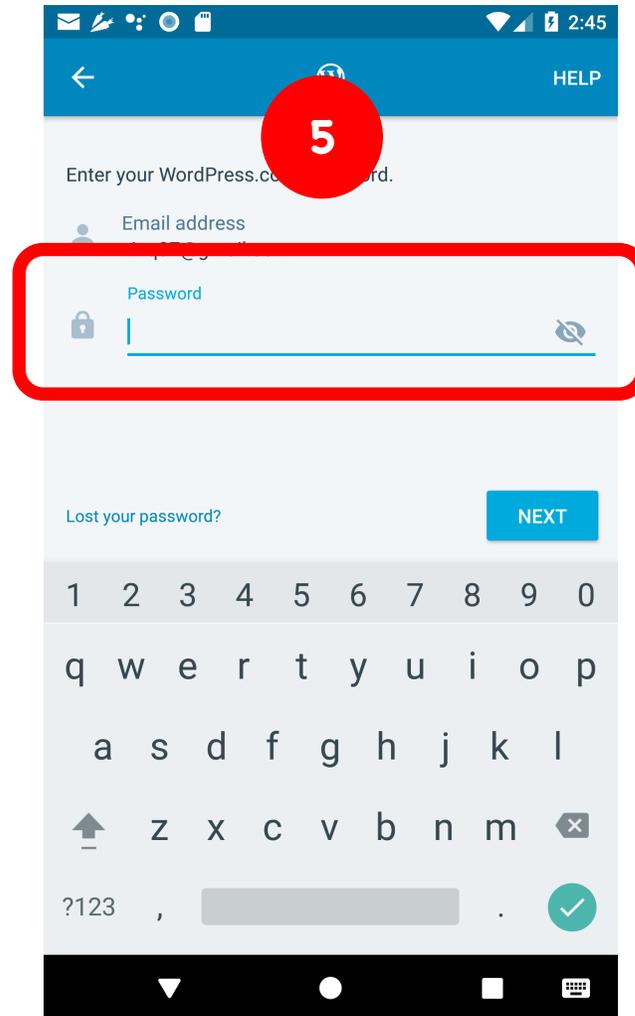
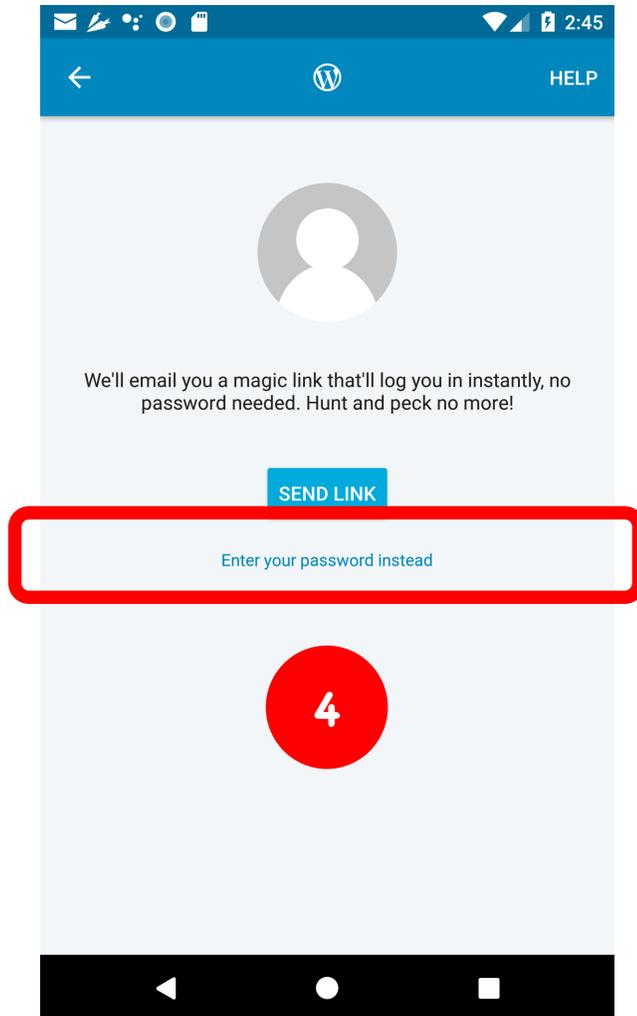
# Our Approach

- Identifies features by analyzing execution trace
- Targets Mobile (Android) apps
- In our context, a feature is *a sequence of user events* that exercise some functionality of the app

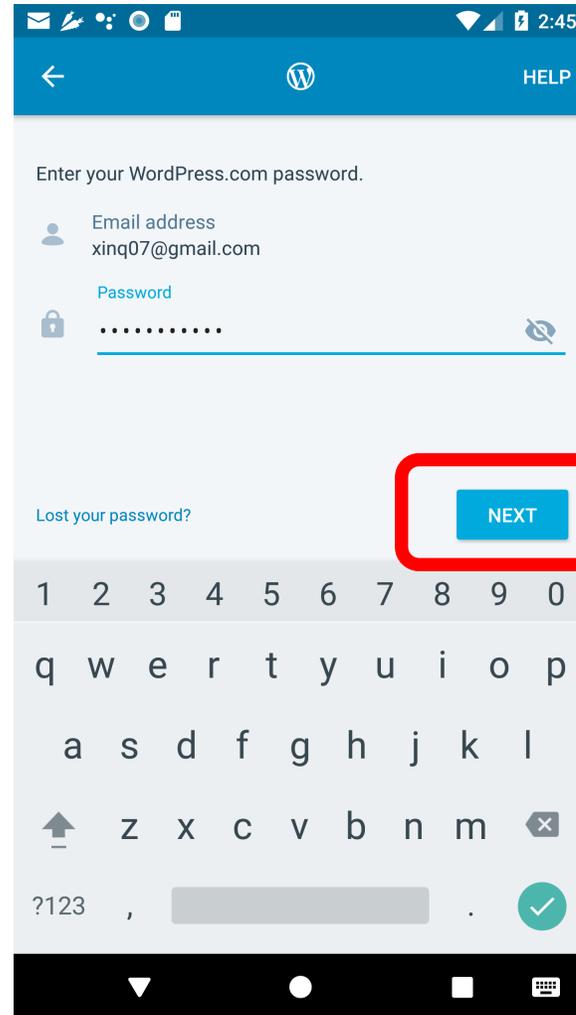
# Login Feature of WordPress



# Login Feature of WordPress



# Login Feature of WordPress

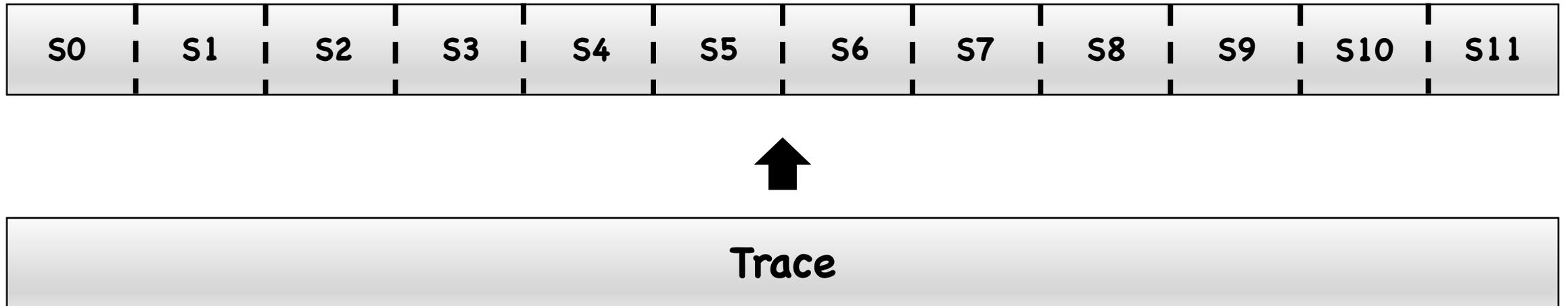


# High-level Approach

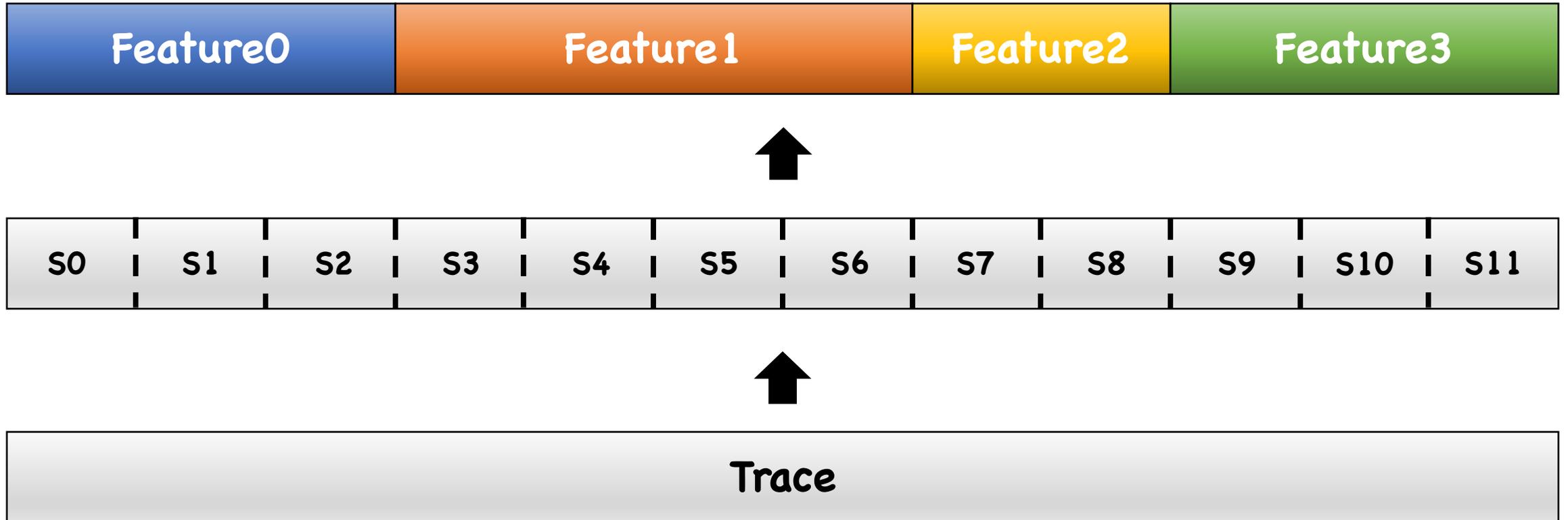


**Trace**

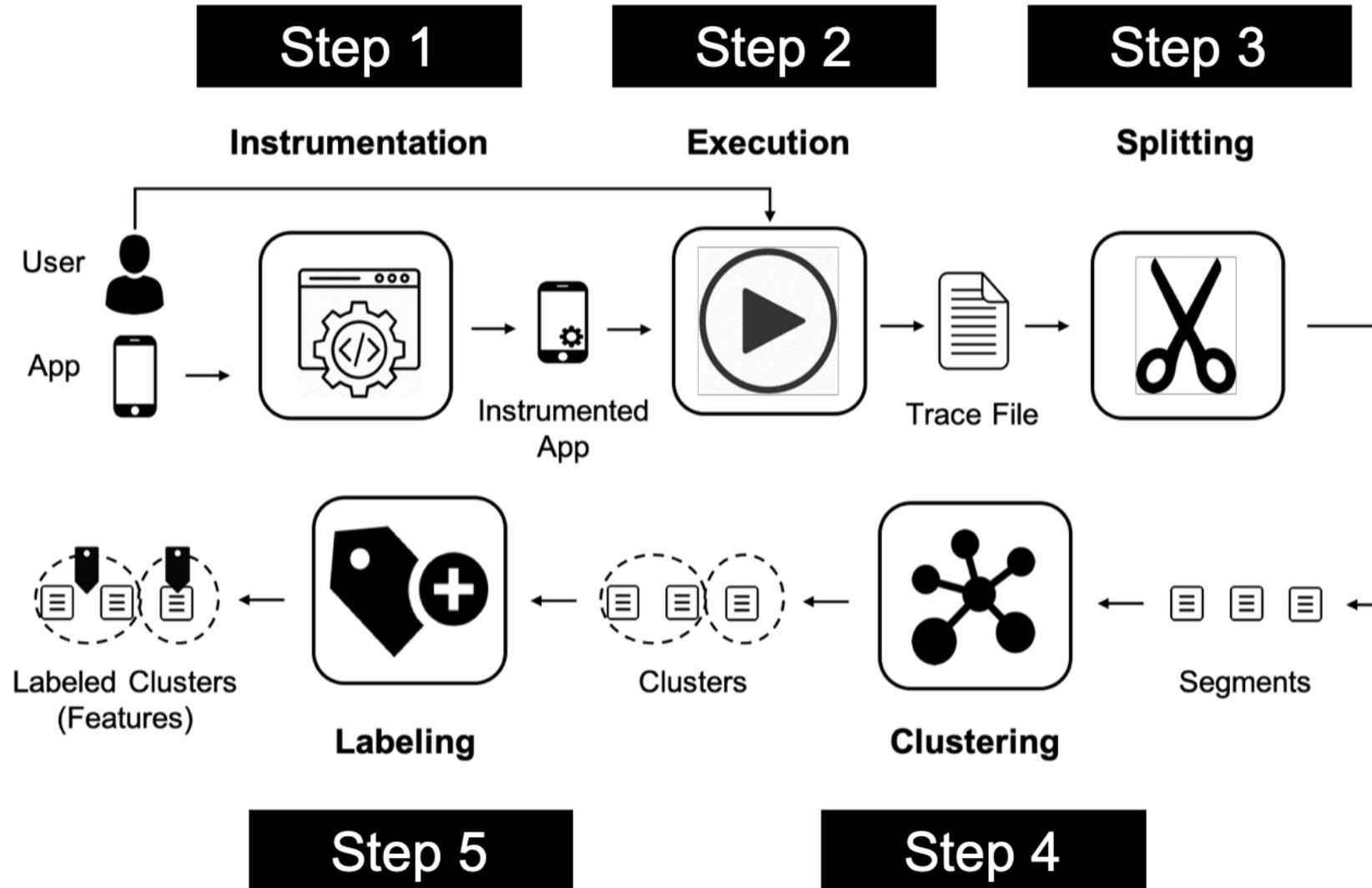
# High-level Approach



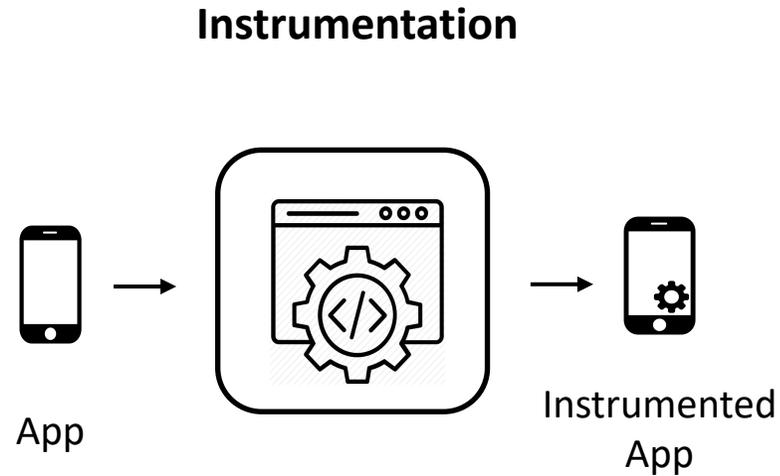
# High-level Approach



# FeatureFinder



# FeatureFinder

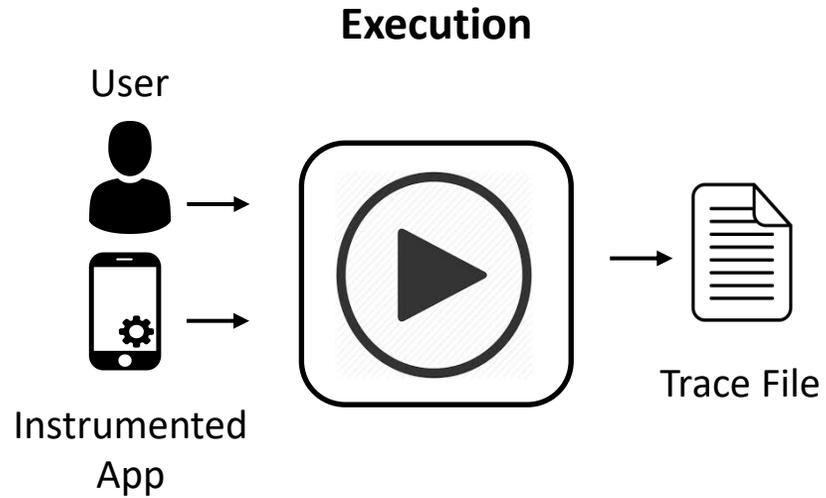


Step 1

## Capture execution information

- Stacks of Method Calls
- Activities & Fragments
- User Events
  - Touch event & widgets
  - Keyboard event & labels

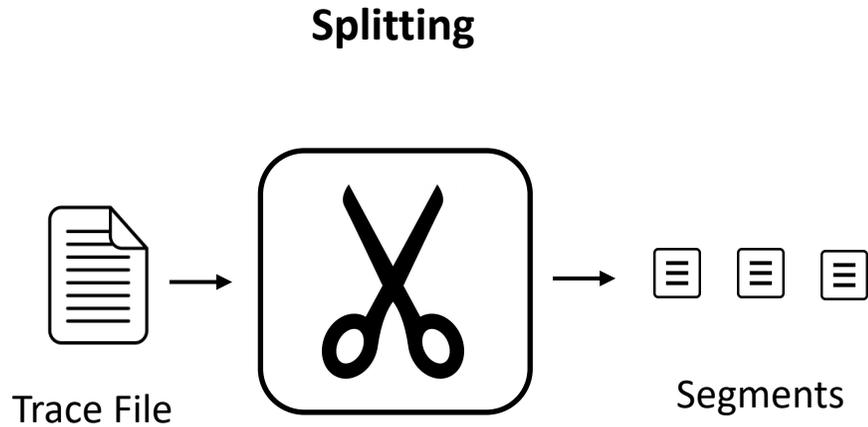
# FeatureFinder



User executes the app  
to exercise its features

Step 2

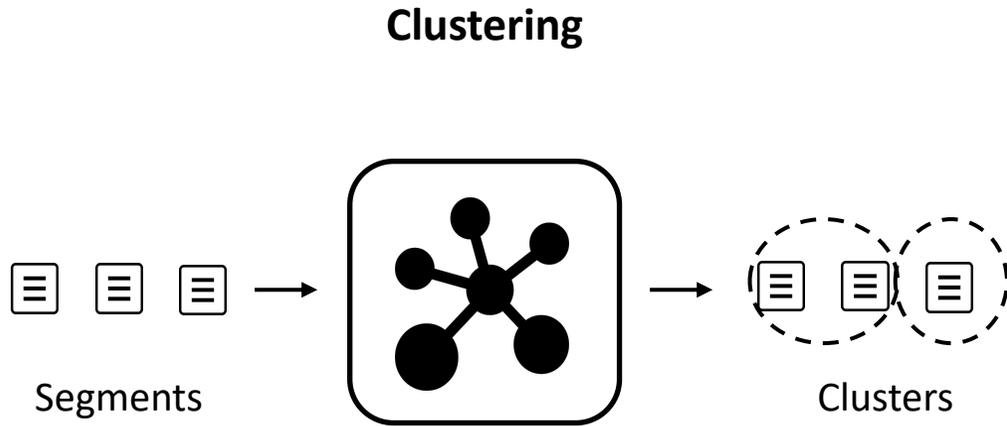
# FeatureFinder



Split the trace into segments by user events

Step 3

# FeatureFinder



Step 4

Group "related" segments

- Compare execution info
- Use a classifier to decide "relatedness"

# Clustering algorithm

S0



S1



S2



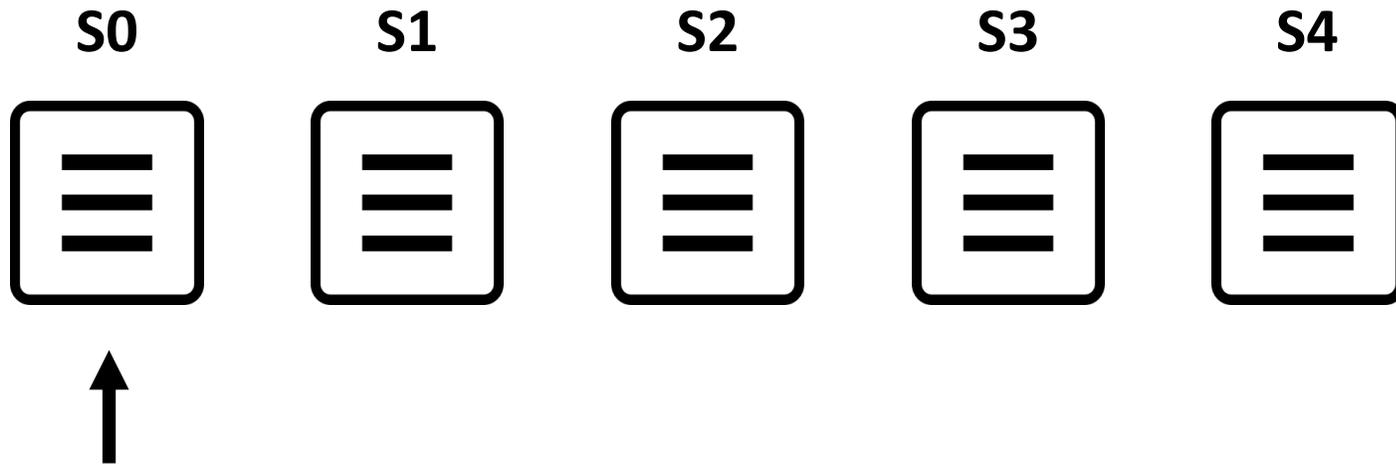
S3



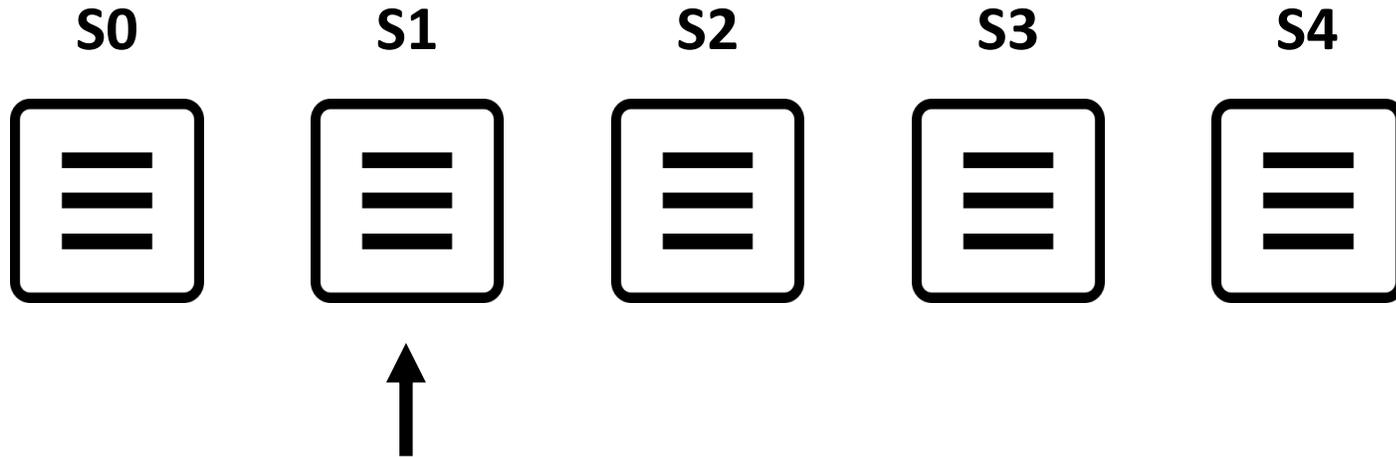
S4



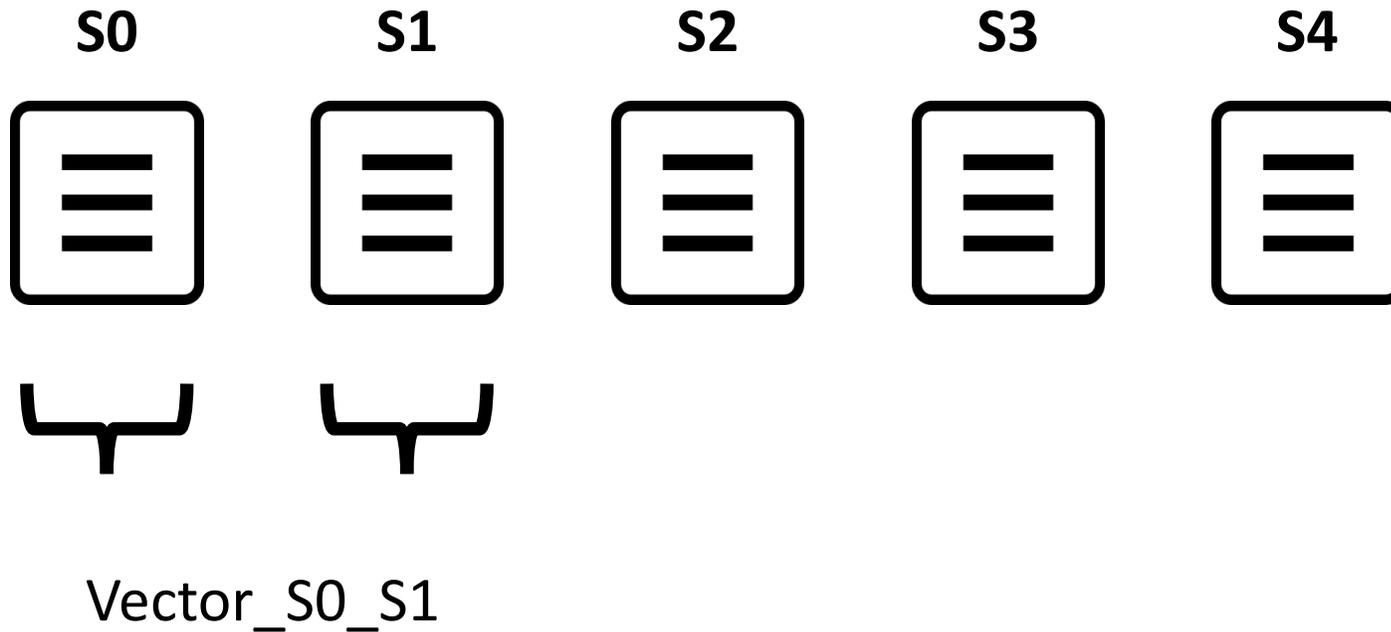
# Clustering algorithm



# Clustering algorithm

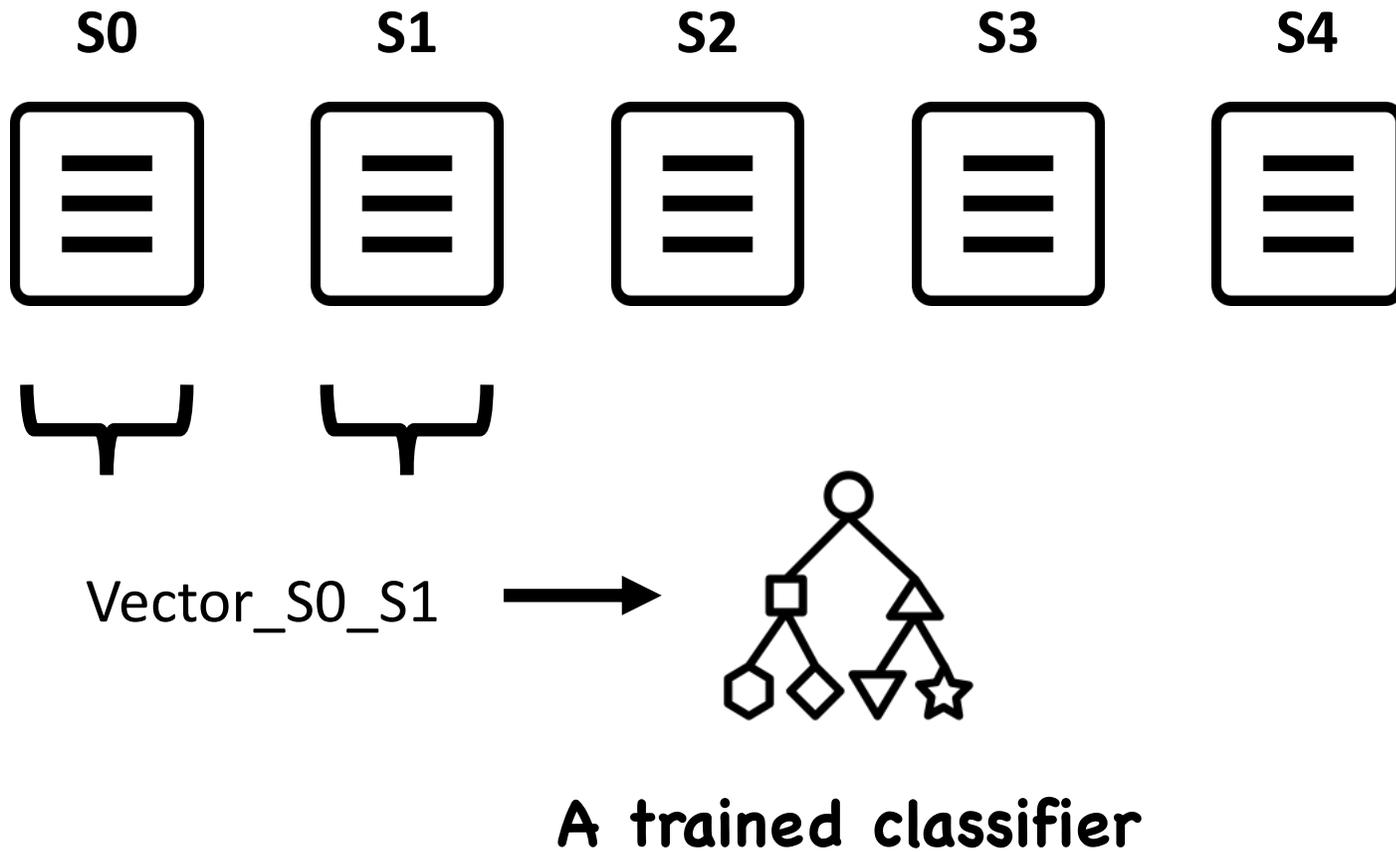


# Clustering algorithm

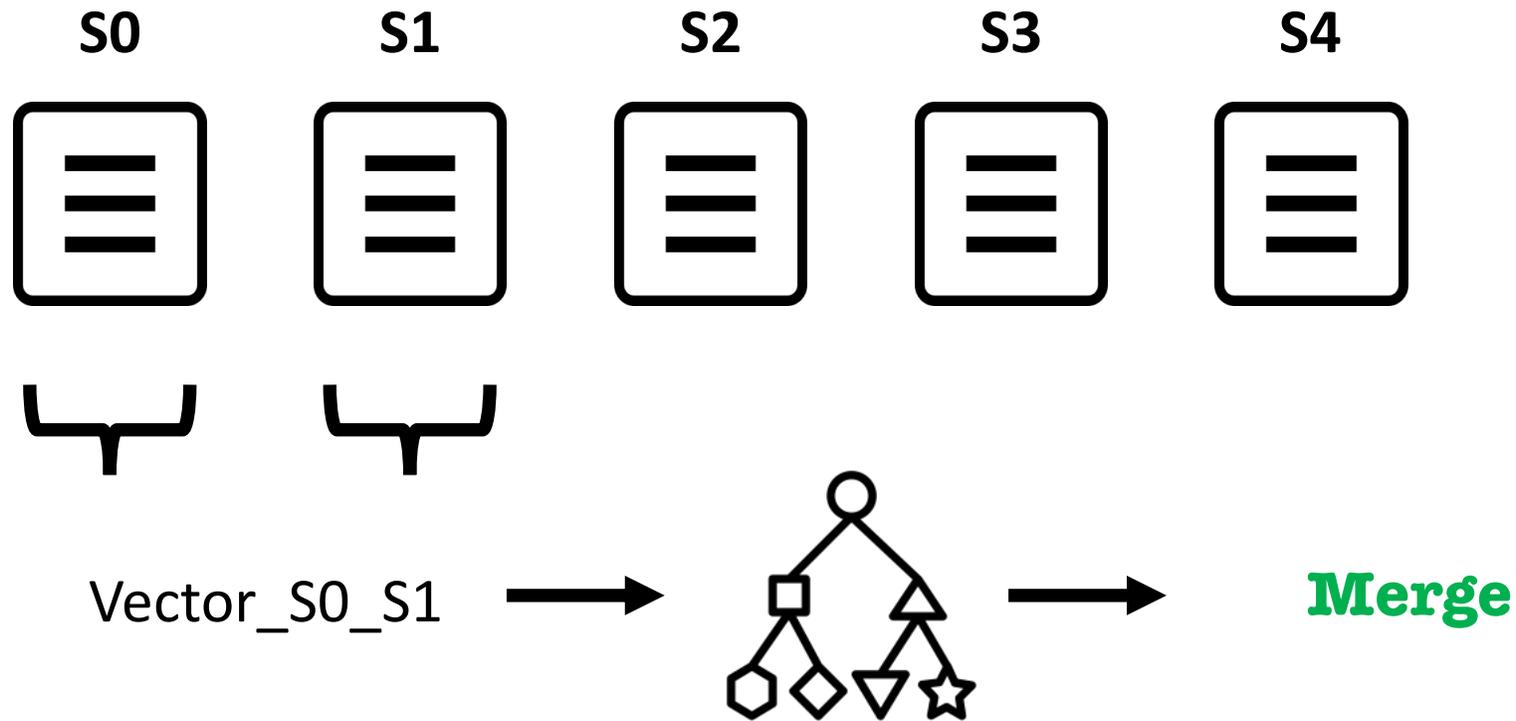


**A numeric vector encoded as the comparison b/w S0 and S1**

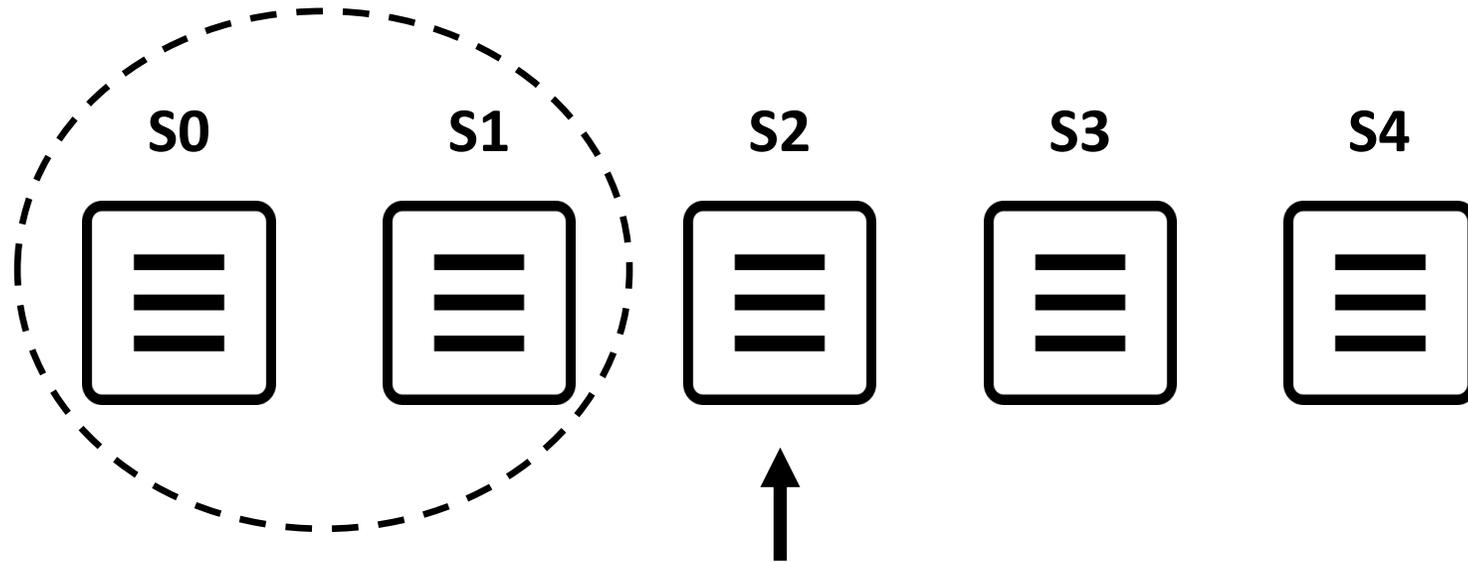
# Clustering algorithm



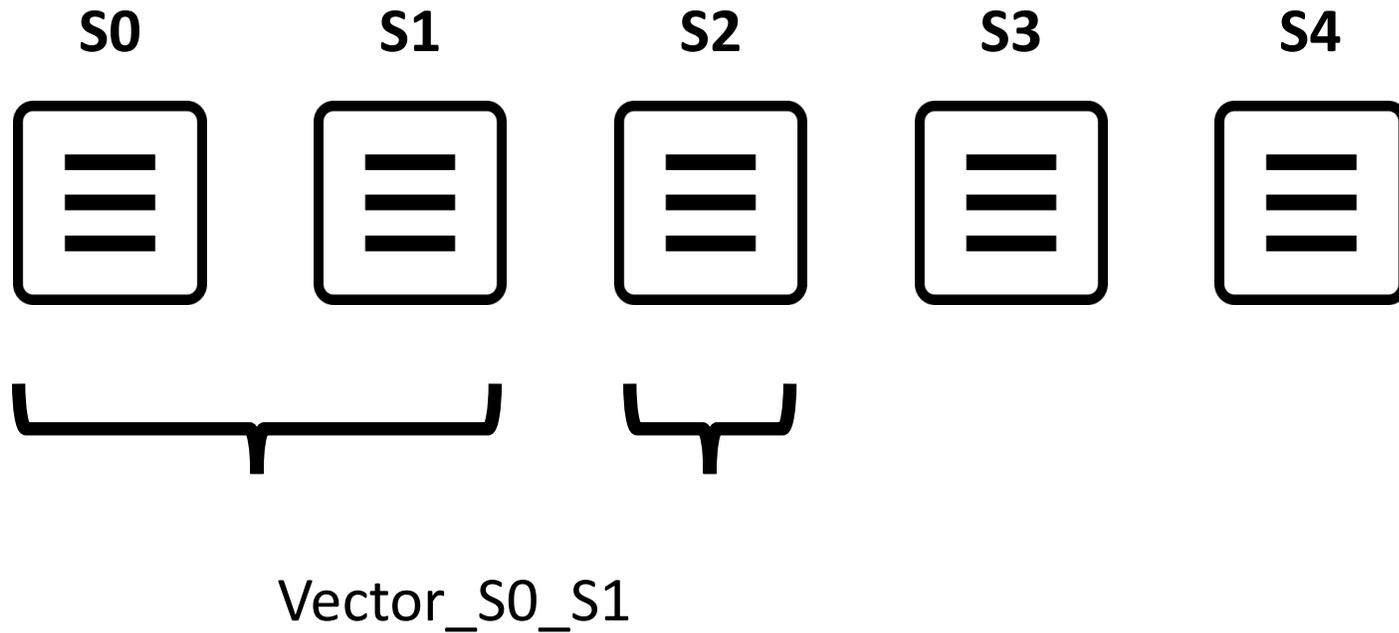
# Clustering algorithm



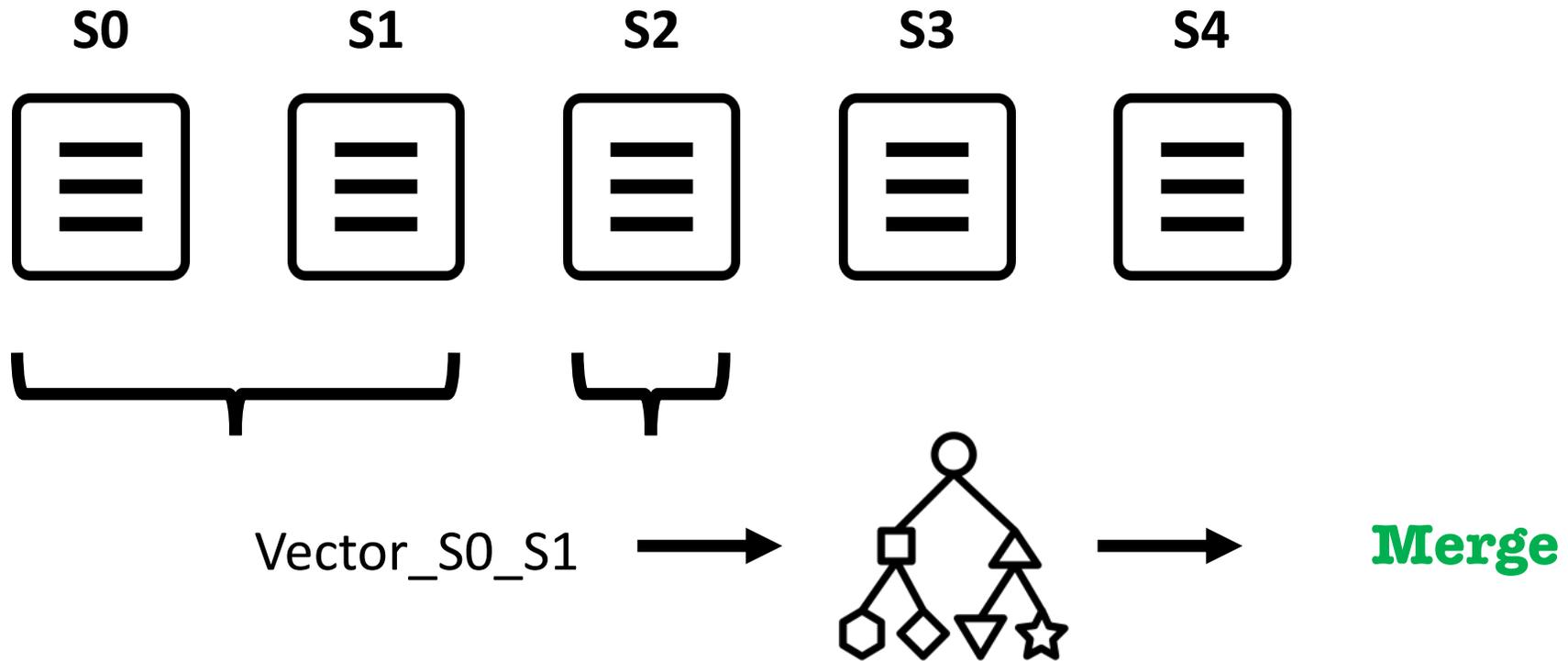
# Clustering algorithm



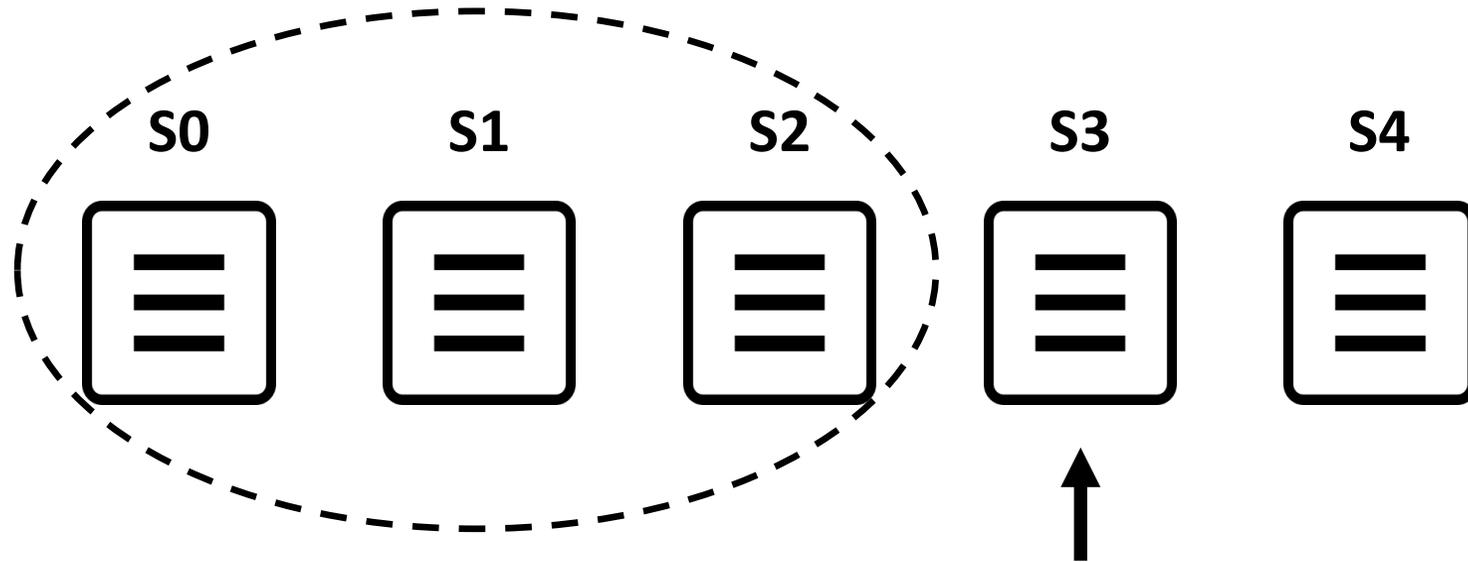
# Clustering algorithm



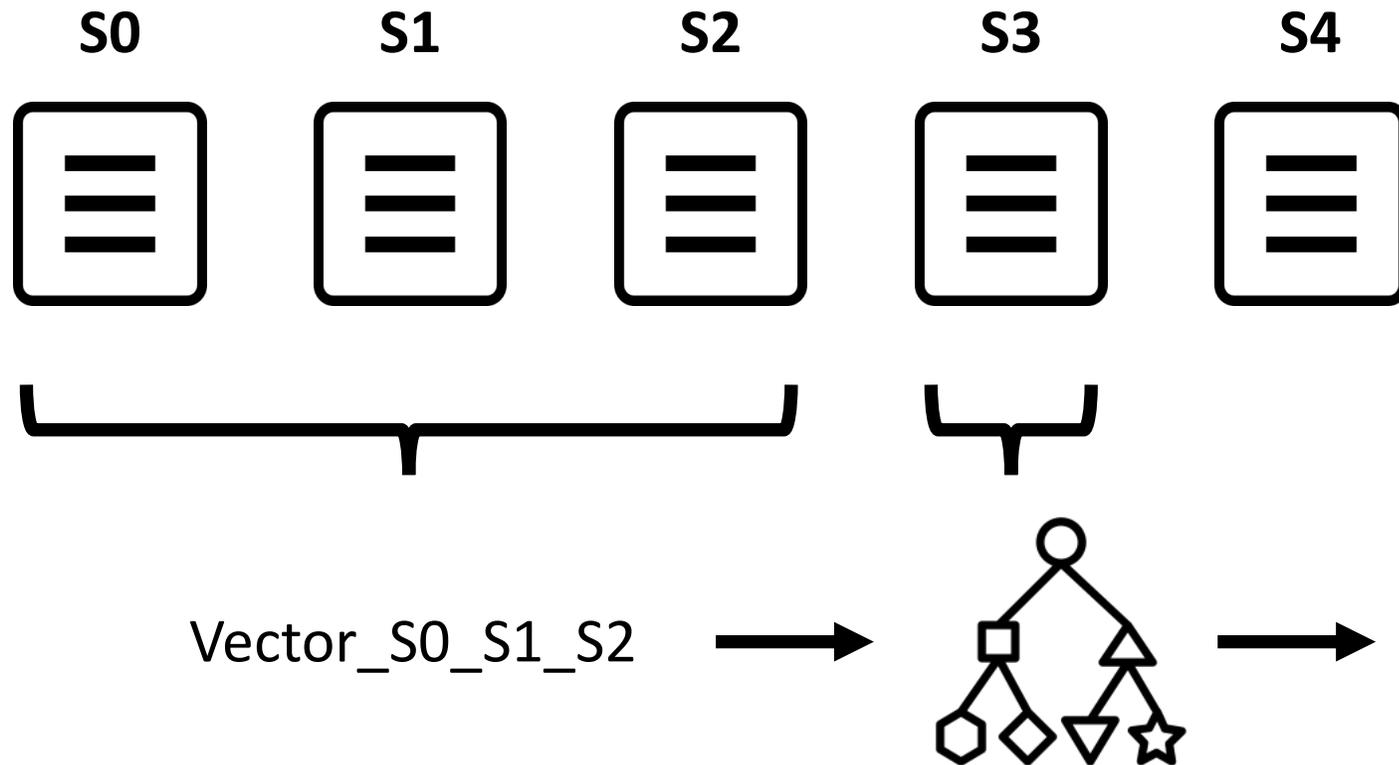
# Clustering algorithm



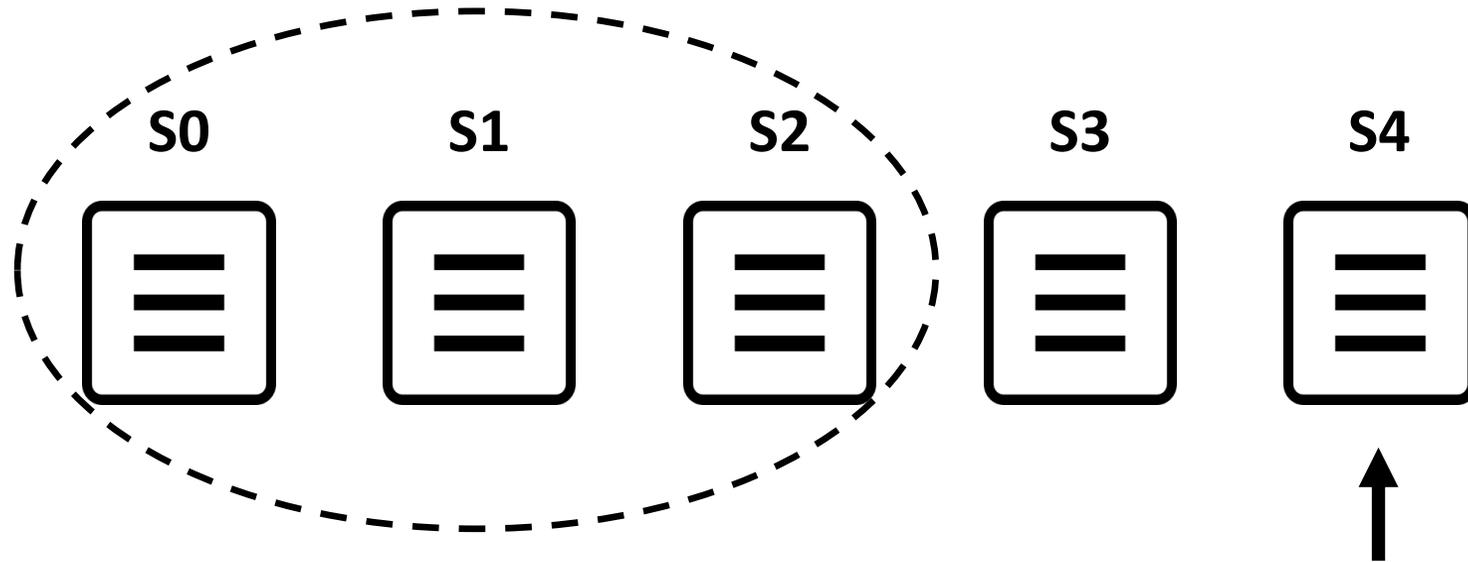
# Clustering algorithm



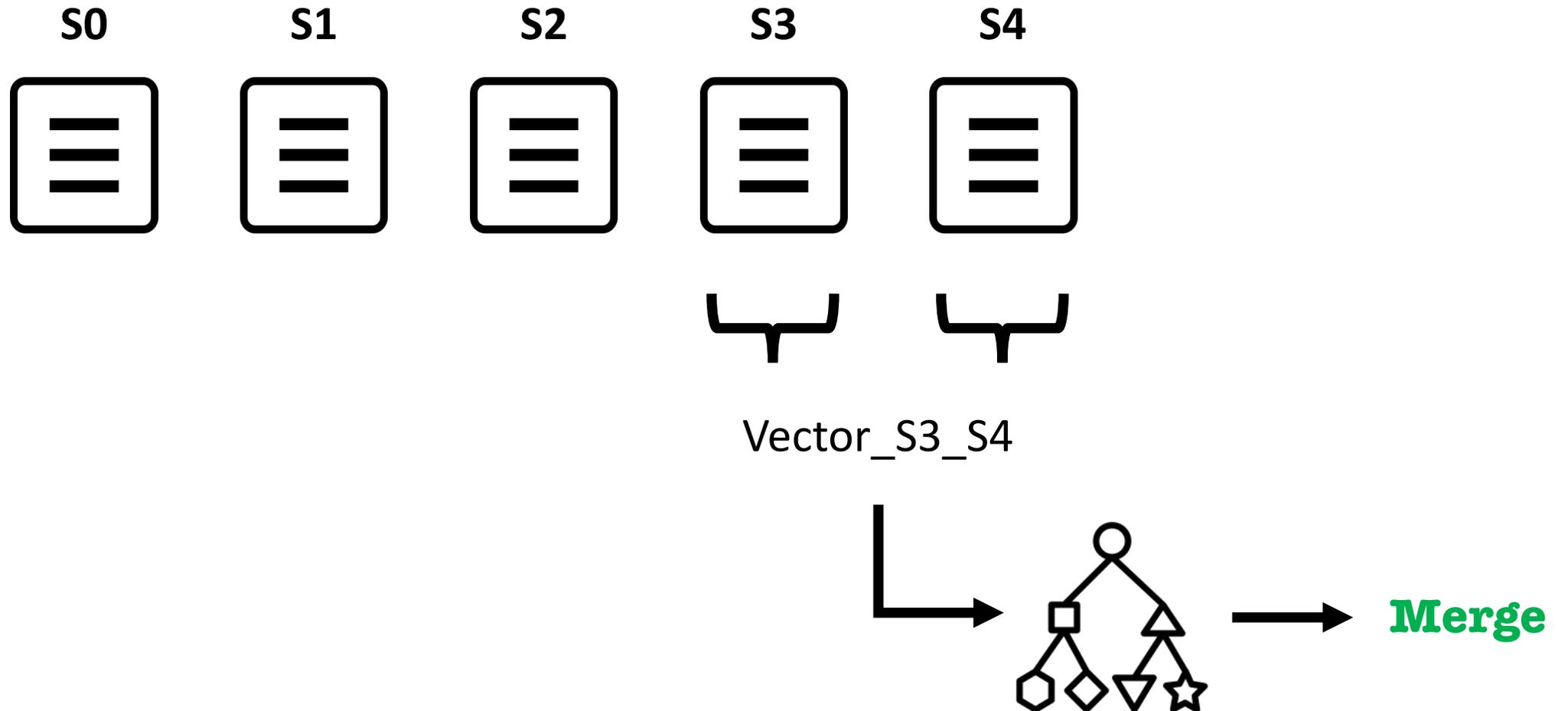
# Clustering algorithm



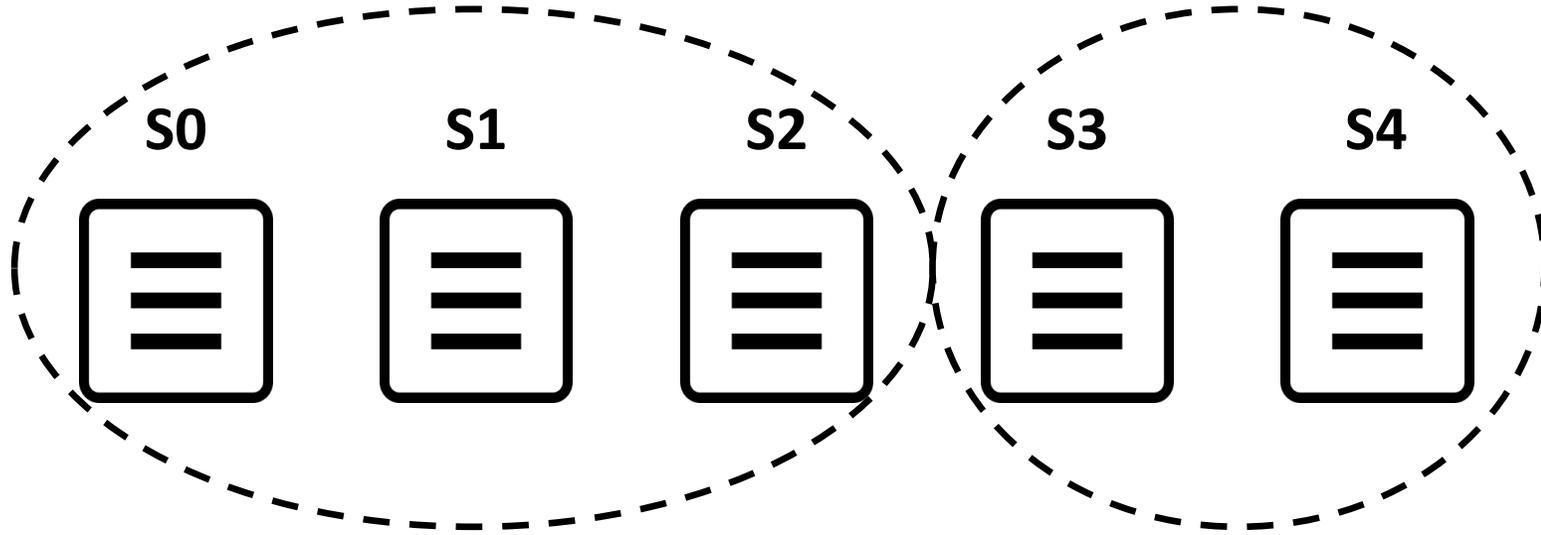
# Clustering algorithm



# Clustering algorithm

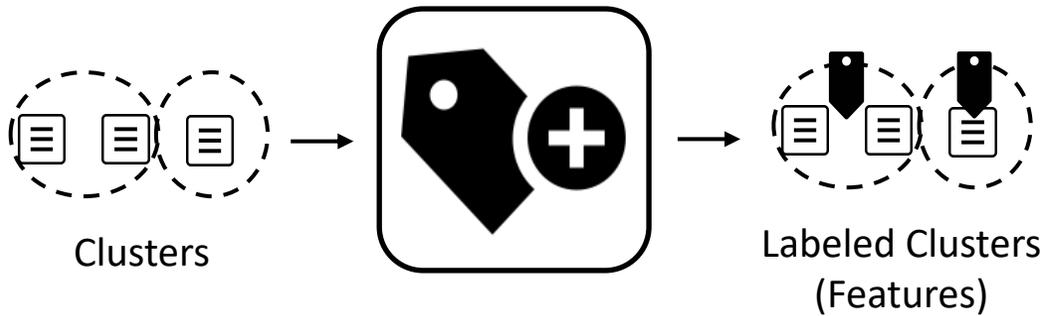


# Clustering algorithm



# FeatureFinder

Labeling



**Label each cluster**

- **Use activity & fragment names**
- **Rank names by TF-IDF values**
- **Select the top-10**

**Step 5**

# Case Study

- Conducted a study using 5 apps
- Exercised different app features and generated traces
- Used *4 apps* for classifier training
- Evaluated FeatureFinder on the other app *K-9 Mail*

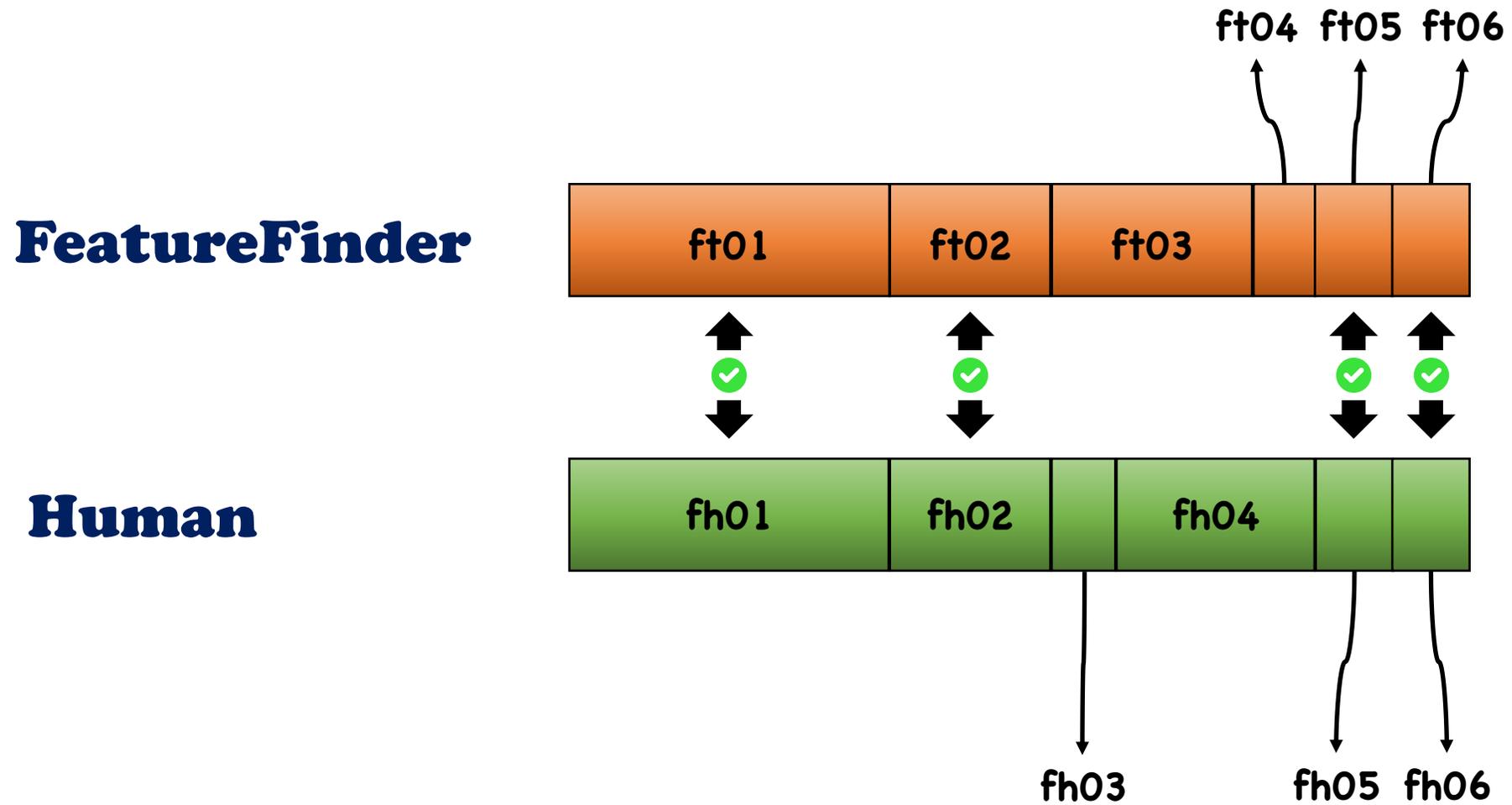
WordPress  
DailyMoney  
PasswordMaker  
Music Player  
K-9 Mail

Two Trace for  
each App

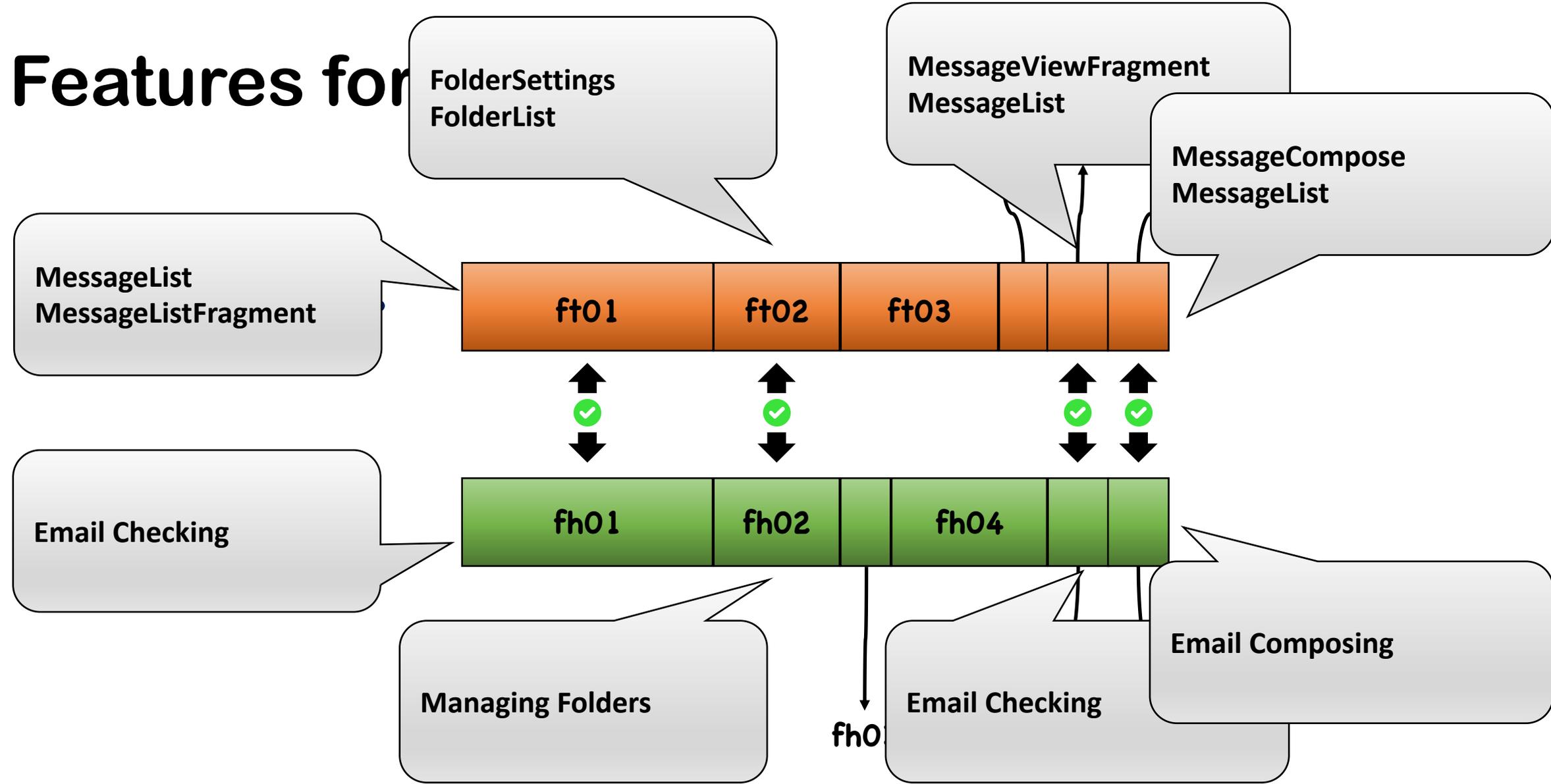
# Classifier Training

- Used FeatureFinder to split trace into segments
- Manually Identified clusters
- Generated *490 segments pairs* labeled as “Merge” & “Don’t merge”
- Trained *10 classifiers*
- Used the best: *k-NN (k=10)*

# Features for Trace 0



# Features for



# Evaluation Results

- Manually identified 11 feature clusters (**ground truth**)
- FeatureFinder generated 9 clusters
- Identified 6 of the 11 (**55%**) features
- Labels generated are ***in close meaning*** to the human labels

# Conclusion & Future work

- FeatureFinder identifies *features* from app's *execution traces*
- Case study results, albeit preliminary, are promising
- As future work
  - Perform a user study
  - Extend FeatureFinder to identify features hierarchically
  - Define a visualization for showing the features

# Understanding a program & its features



Refactoring



Debloating



Debugging



Functionality Modification



Testing



Documentation

**Step 1**

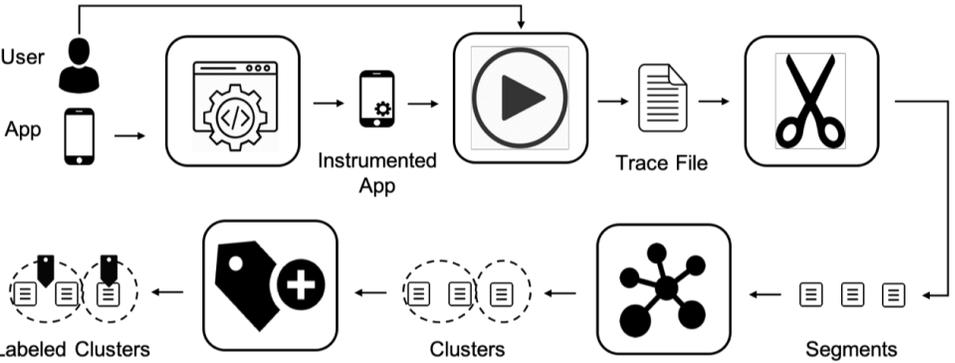
Instrumentation

**Step 2**

Execution

**Step 3**

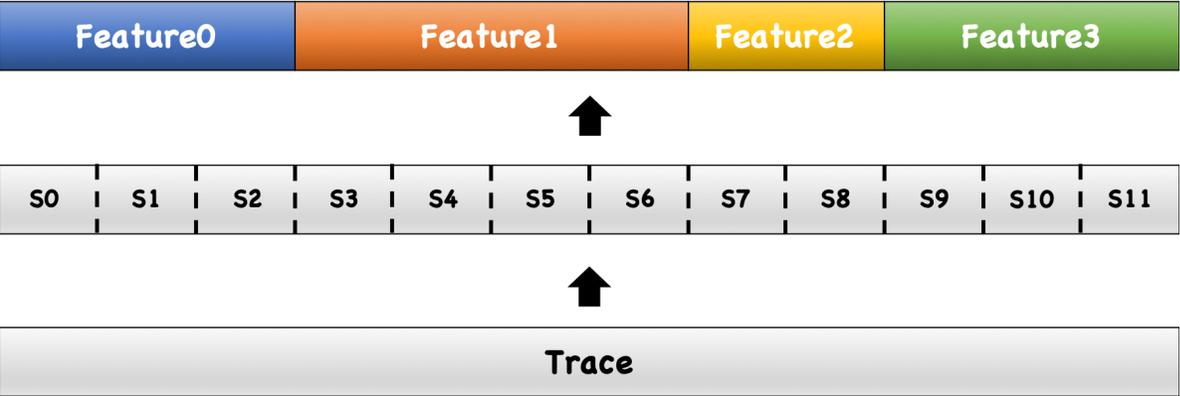
Splitting



**Step 5**

**Step 4**

# High-level Approach



# Evaluation Results

- Manually identified 11 feature clusters (**ground truth**)
- FeatureFinder generated 9 clusters
- Identified 6 of the 11 (**55%**) features
- Labels generated are **in close meaning** to the human labels