On automatically detecting similar Android apps (CLANdroid)

Authors: M. Linares-Vásquez, A. Holtzhauer, and D. Poshyvanyk
Presented By: Abhinav Kumar

Building the hype!

Imagine!

What would you do if….

- You are an aspiring **app developer** with an amazing idea, but not sure if it's already done
- You have a **developer** and you want to learn how other apps have implemented an idea
- You are **Google** and you want to know if the newly submitted app has security vulnerabilities
- You are a **user**, who wants to look at free/less buggy apps, similar to a paid/buggy app. Play store recommendation did not help you
- You are **Google** and you want to detect plagiarism.

Few solutions

- Look at Play Store recommendations
- Type a long sentence and let Google do its magic

Limitation(s)

- Heavily dependent on textual description of the app
  - **Opposing Argument 1**: Can you explain the entire functionality of an app in few sentences?
  - **Opposing Argument 2**: The code should have some say in the decision
- Code obfuscation makes it hard to understand code
- Third party libraries

Now we know there is a problem/need.
Solution?

Introducing CLANDroid

- An approach for automatically detecting **C**losely re**L**ated applications in **AN**droid
- Using advanced IR techniques (**Latent Semantic Indexing**)
- And 5 semantic anchors
  - Identifiers, Android APIs, Intents, Permissions, and Sensors

Contrasting with CLAN

- CLAN only used API calls to detect similarity

It’s time for a deep dive into CLANdroid

Let’s start with some background knowledge

Android Concepts

Intents: you express your wishes as intents to Android.

Ex.: open this url in browser, open this image file, make a phone call, etc.

Permissions: you allow apps to use Android features.

Ex.: allow app to read stored files, allow app to access contacts, etc.

Android Concepts

Sensors

Ex.: accelerometer, ambient temperature, magnetic field sensor, etc.

API

Ex.: Google sign-on api, FCM api, etc

Latent Semantic Indexing (LSI)

A technique in **natural language processing**, of analyzing **relationships** between a set of **documents** and the **terms they contain** by producing a set of concepts related to the documents and terms.

LSI example

	D1	D2	D3	D4
W1	1	40	6	90
W2	100	4	80	10
W3	2	70	3	20
W4	30	8	50	1

Term Document Matrix (TDM)

Statistical Significance

A result has statistical significance when it is very unlikely to have occurred.

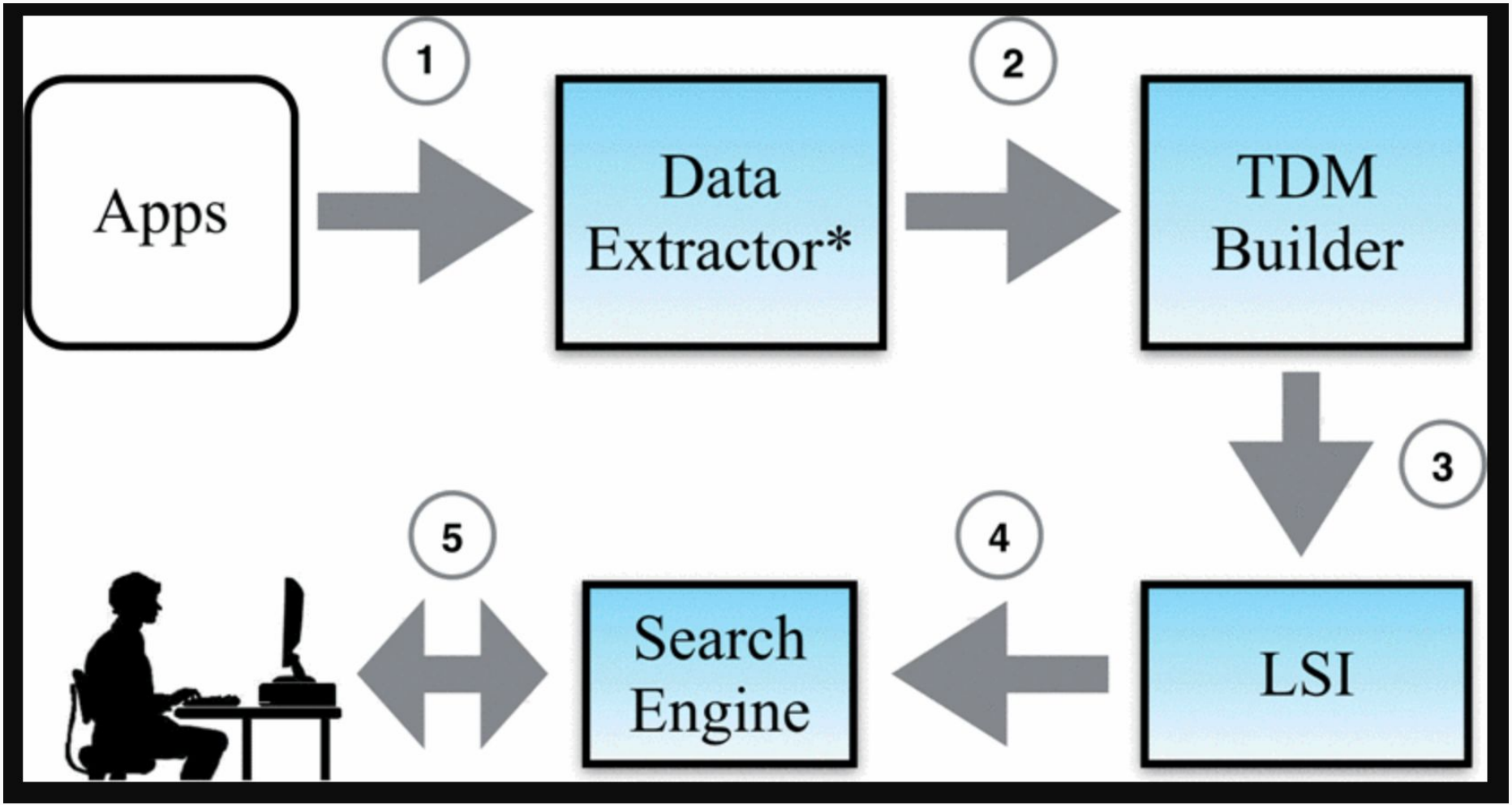
Significance Level (α): the probability of the study rejecting the null hypothesis

P-value: the probability of obtaining a result

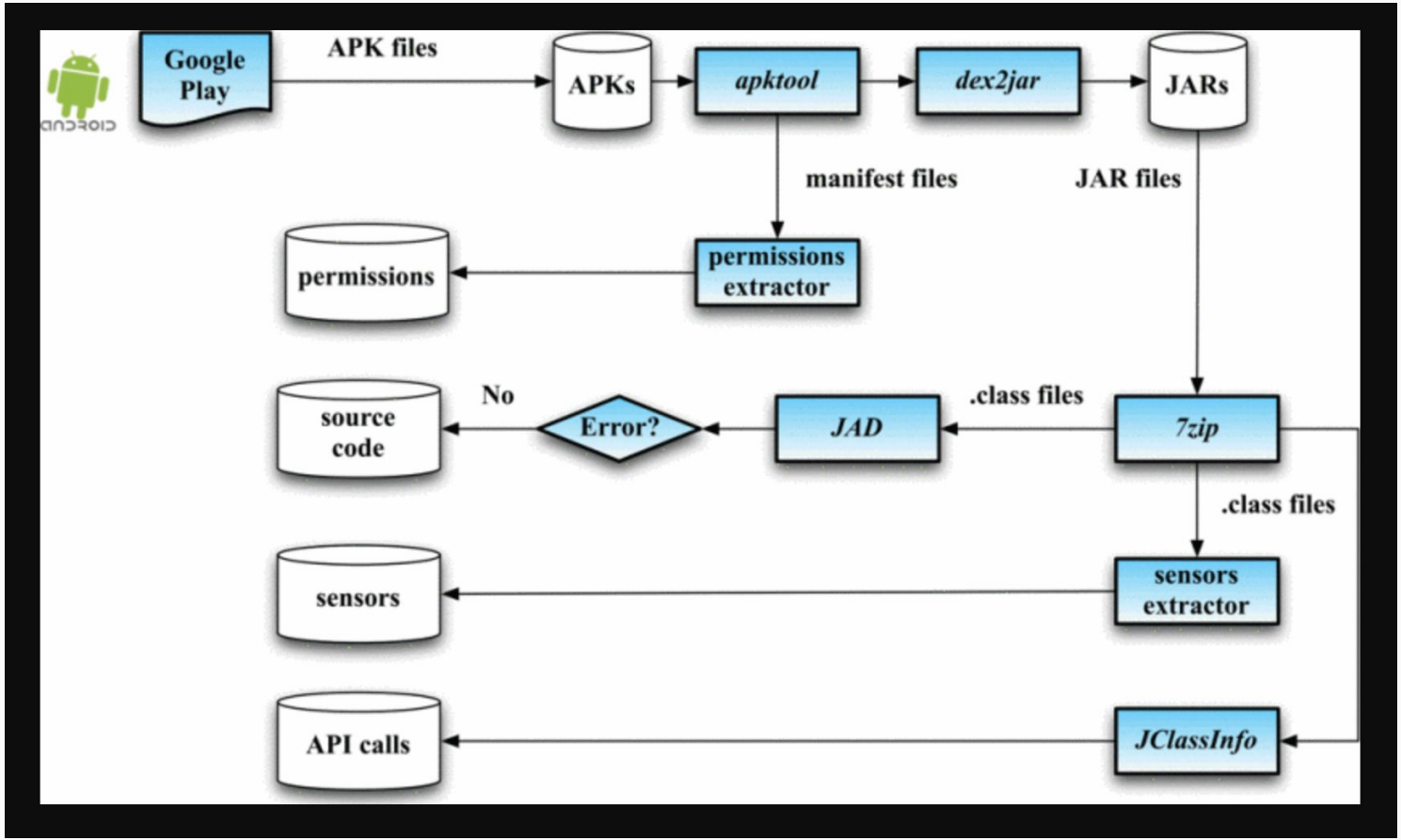
Effect size: measure of a study's practical significance

A result is statistically significant if p-value is less than α

CLANdroid architecture


Apps
1
Data Extractor*
2
TDM Builder
3
LSI
4
Search Engine
5

Data Extractor Workflow



A closer look at the empirical study

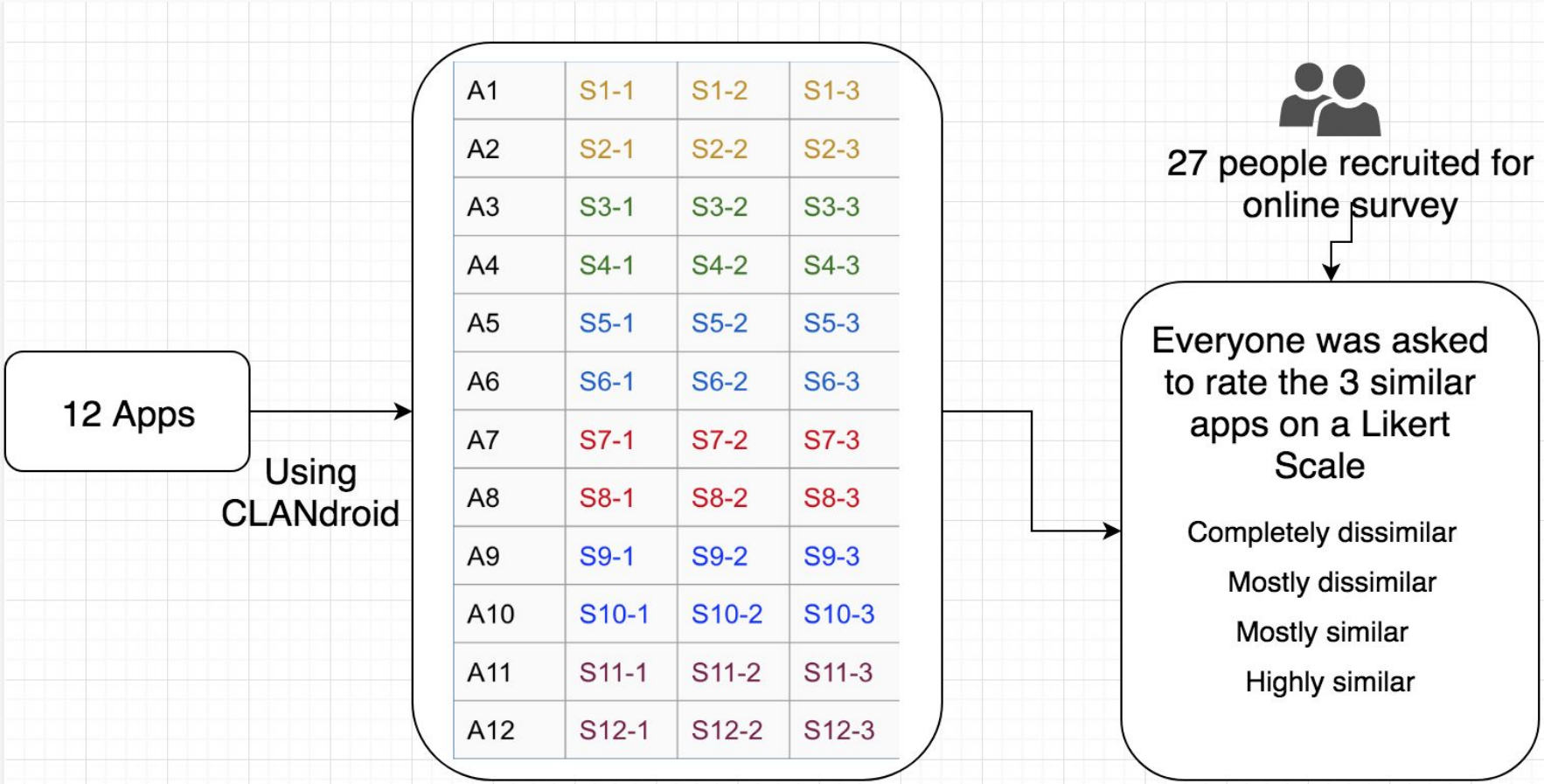
Dataset

- 14,450 free android apps downloaded from Play Store
- Results of the online survey
- Compared against goldset of similar apps provided by Google

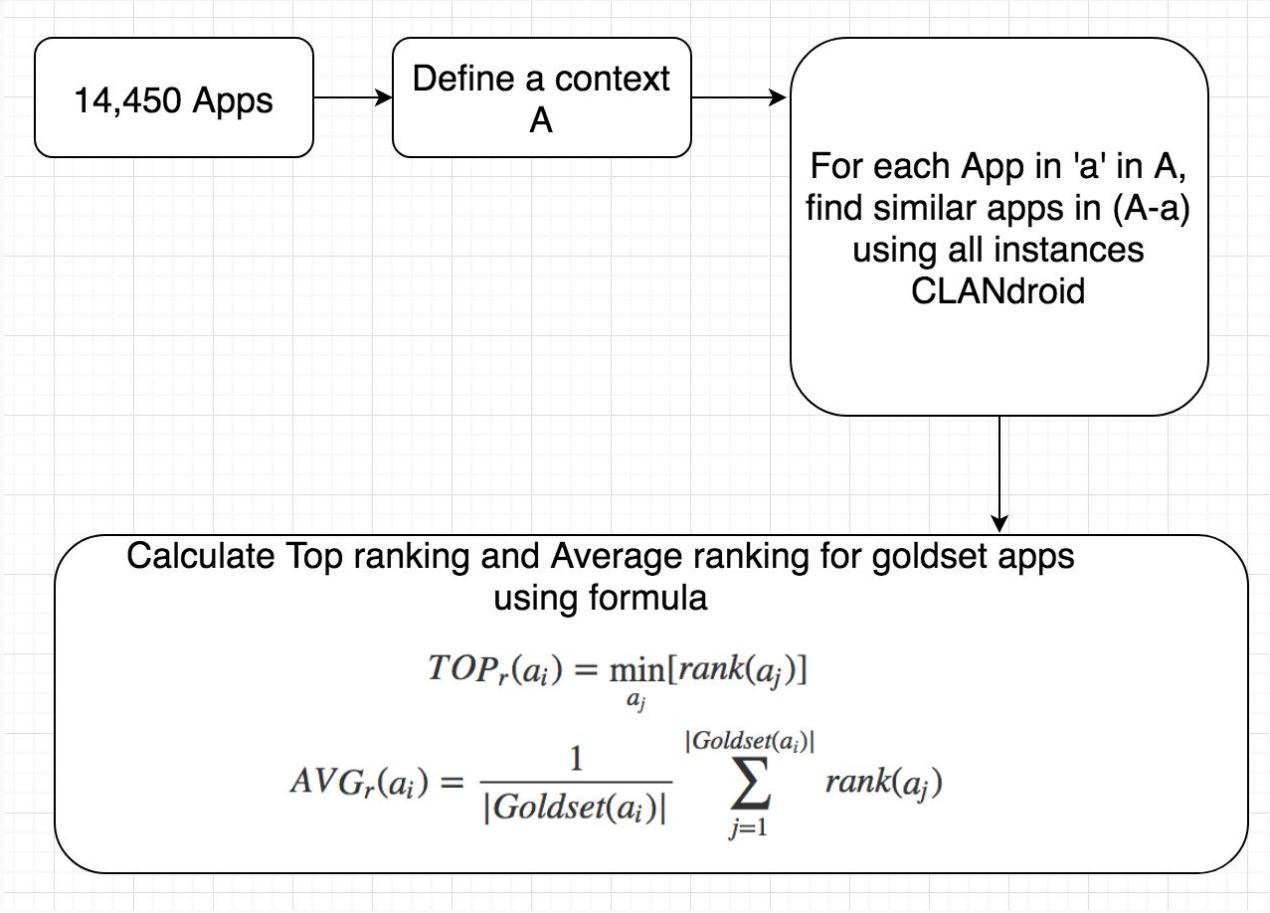
Research Questions

1. What semantic anchors used in CLANdroid produce better results when compared to the others?
2. How orthogonal are the apps detected by CLANdroid as compared to Google Play?
3. Do third-party libraries and obfuscated apps impact the accuracy of CLANdroid?

Study Design (for RQ1 and RQ3)



Study Design (for RQ2)



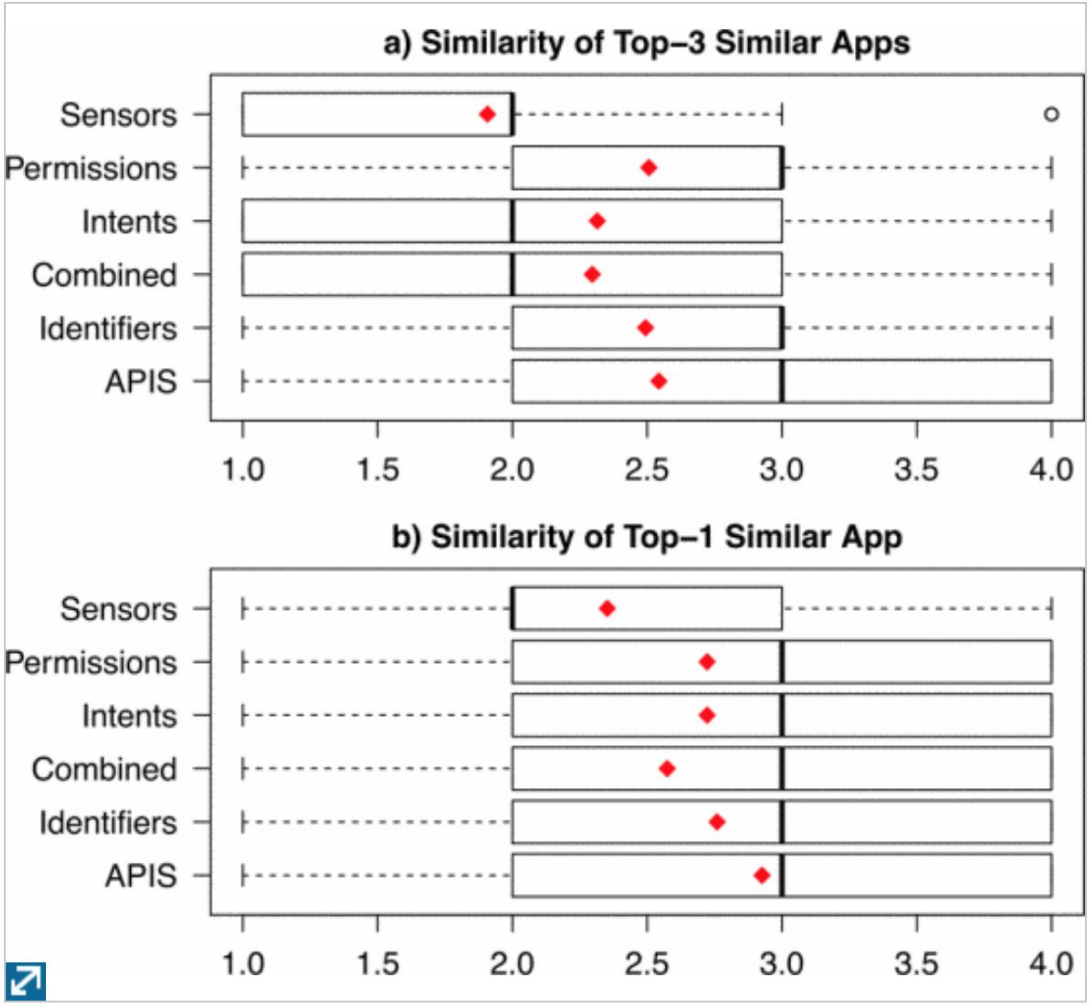
Analysis Method

- **For RQ1**: Kruskal-Wallis test with post-hoc test procedure for pairwise comparisons on each CLANdroid instance
- **For RQ2**: compared the TOP and AVG series of the CLANdroid instances with the Kruskal-Wallis test with post-hoc procedure
- For RQ3: pairwise comparisons using Mann-Whitney
  - With and without TPL
  - With and without obfuscation

Analysis Method

- Alpha level: 0.05
- Used Cliff's delta d effect size
  - negligible for $|d|<0.147$
  - small for $0.147\leq|d|<0.33$
  - medium for $0.33\leq|d|<0.474$
  - large for $|d|\geq0.474$)

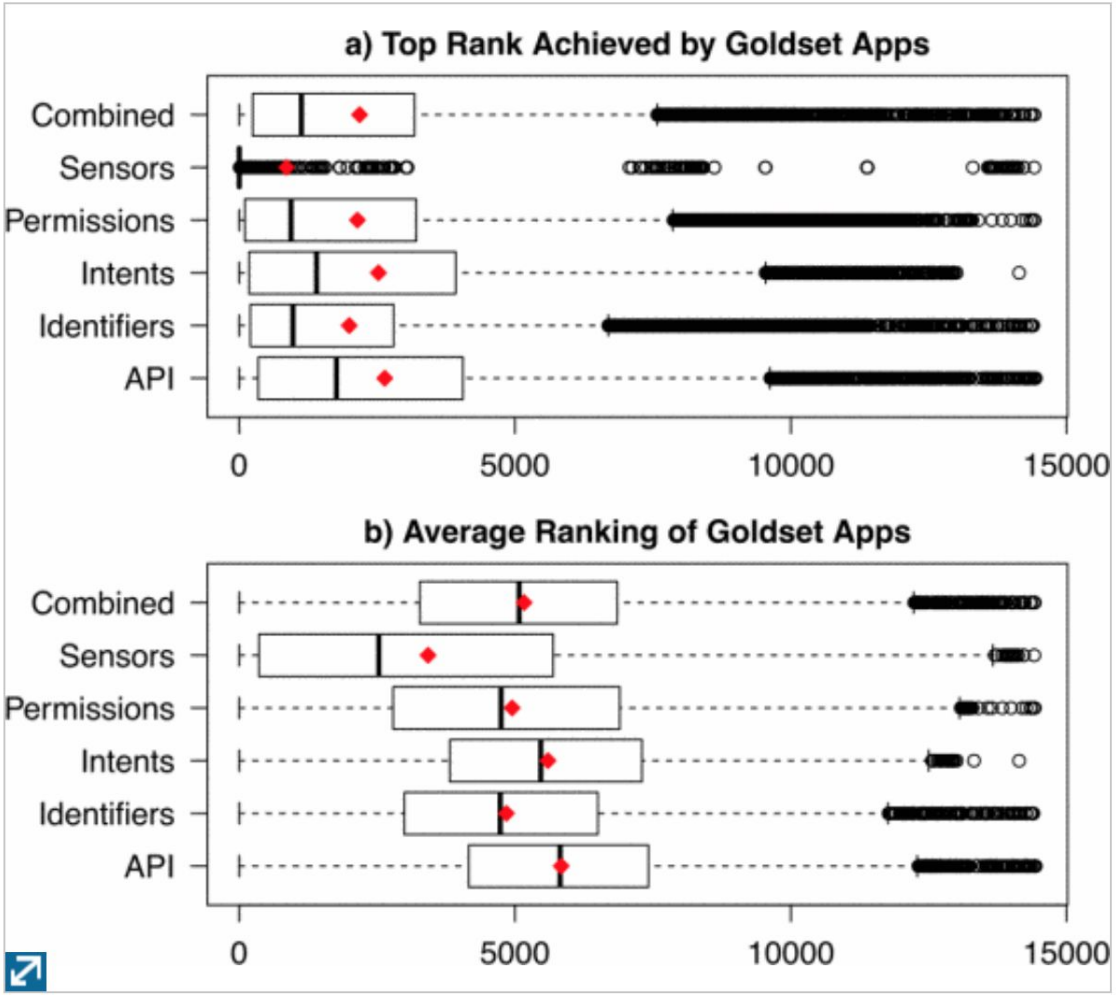
Results - RQ1


a) Similarity of Top−3 Similar Apps
Sensors
Permissions
Intents
Combined
Identifiers
APIS
1.0
1.5
2.0
2.5
3.0
3.5
4.0
b) Similarity of Top−1 Similar App
Sensors
Permissions
Intents
Combined
Identifiers
APIS
1.0
1.5
2.0
2.5
3.0
3.5
4.0

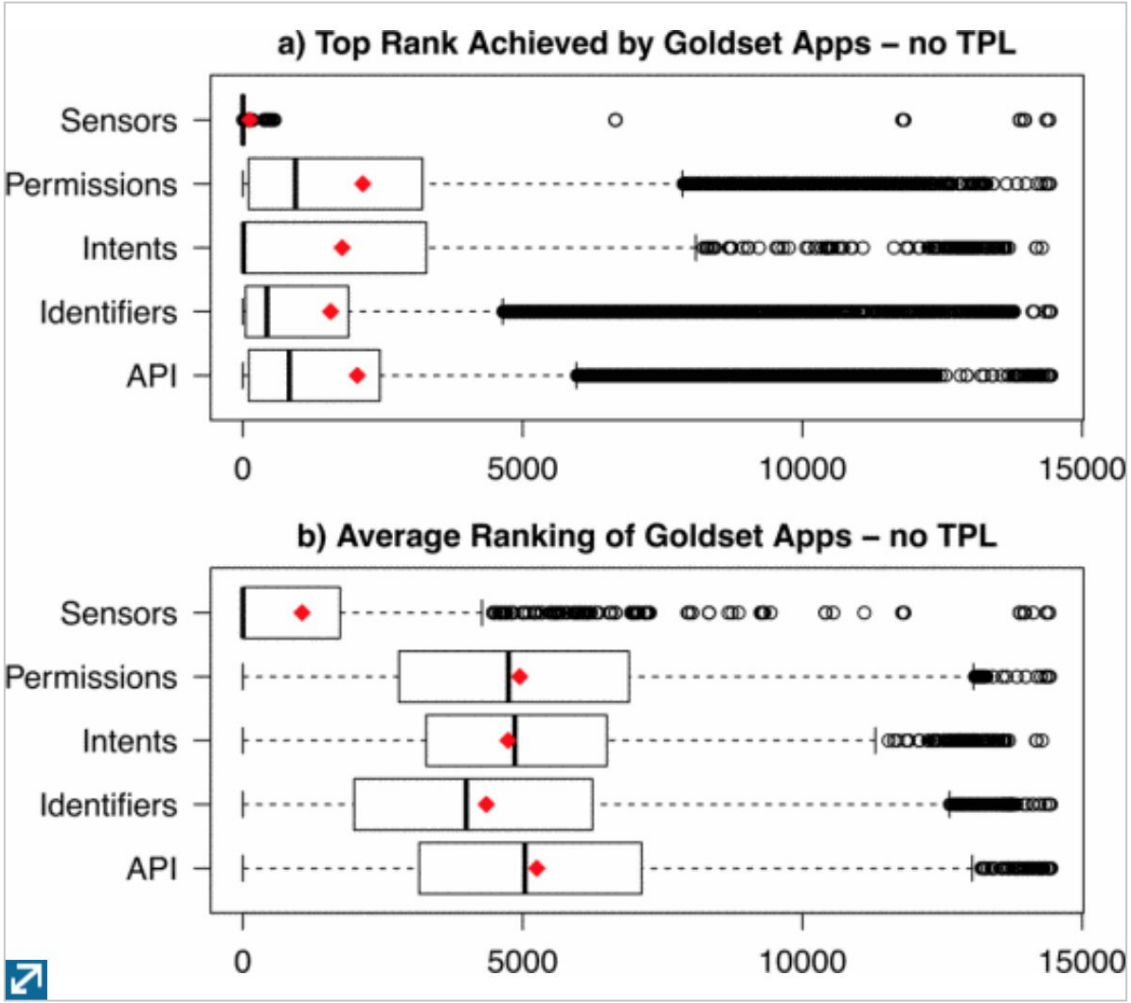
Pairwise comparison, alpha = 0.0033

Control	**Treatment**	p-value	Cliff's $\|d\|$
$CLANdroid_{API}$	$CLANdroid_{Sens}$	2.72e-07	0.31790
$CLANdroid_{Ident}$	$CLANdroid_{Sens}$	7.46e-07	0.30571
$Combined$	$CLANdroid_{Sens}$	0.00076	0.20694
$CLANdroid_{Int}$	$CLANdroid_{Sens}$	0.00064	0.20995
$CLANdroid_{Perm}$	$CLANdroid_{Sens}$	9.79e-07	0.30251

Results - RQ2


a) Top Rank Achieved by Goldset Apps
Combined
Sensors
Permissions
Intents
Identifiers
API
0
5000
10000
15000
b) Average Ranking of Goldset Apps
Combined
Sensors
Permissions
Intents
Identifiers
API
0
5000
10000
15000

Results - RQ3


a) Top Rank Achieved by Goldset Apps – no TPL
Sensors
Permissions
Intents
Identifiers
API
0
5000
10000
15000
b) Average Ranking of Goldset Apps – no TPL
Sensors
Permissions
Intents
Identifiers
API
0
5000
10000
15000

Summarized results

- Except sensors, all other semantic anchors were good at detecting similar apps
  - APIs provided the highest number of apps rated as “highly similar”
- Google Play's detection mechanism is likely to be based not only on textual similarities of descriptions, but also on sensors
- Accuracy of CLANdroid is significantly (negatively) impacted by the inclusion of third-party libraries (TPL)
  - Code obfuscation has negative impact but less severe than TPL

Related Work

- AnDarwin by Crussell et al. [5], used **code methods** as semantic vectors
- DStruct [6], used **directory structure** of the app for similarity
- Chen et al. [7], used dependency graphs at **method level** to check for clones
- Desnos [8], used **method signatures** to detect similar Android apps, where the signatures were composed of string literals, API calls, control flow structures, and exceptions

Study	Purpose	Information Type	Platform	#apps	TPL	Market
Michail and Notkin [14]	Detecting similar libraries	Library source code	D	NA	NA	NR
Kawaguchi *et al.* [49]	Automatic Categorization	Source code identifiers	D	41	NA	SF
Crussell *et al.* [50]	Detecting cloned and rebranded apps	Java bytecode	M	>265K	YES	MM
Li *et al.* [51]	Using similarities to address security	File directories	M	>58K	NO	MM
Bajracharya *et al.* [52]	Source code retrieval	API calls from source	D	346	NA	E
Chen *et al.* [17]	Detecting cloned apps to address security	Methods from SMALI code	M	>150K	YES	MM
Cubranic *et al.* [53]	Recommending Software Artifacts	Issue-tracking	D	1	NA	E
Moritz *et al.* [54]	API search engine	API methods	D	13K	NA	NR
Gorla *et al.* [55]	Finding unadvertised behavior in apps	API invocations from SMALI	M	>22K	YES	GP
Desnos *et al.* [56]	Detection of similar apps	Custom method signatures	M	2	NO	GP
Ye *et al.* [57]	Context-aware Browsing	Component repository	D	NR	NA	NR
McMillan *et al.* [58]	Finding relevant functions	Function call graph	D	> 18K	NA	FB
Thung *et al.* [59]	Detecting similar applications	Collaborative tagging	D	>100K	NA	SF
Wang *et al.* [10]	Detecting cloned apps	API invocations from SMALI	M	>100K	YES	MM
Shao *et al.* [60]	Detecting cloned apps	Statistical and Structural features	M	>169K	YES	MM

Somethings to think about!

- Is there enough data to support that the existing ways of searching similar apps is not good enough? Need for large scale user study.
- Current approach works as a batch system. Can this be extended as a realtime service? Is it scalable?
- Only some results were statistically significant. Out those results, very few had good effect size
  - Should we run this experiment on a larger scale before drawing any conclusions?
- Length of the survey. Is it too long?
  - Longer a survey is, the less time respondents spend answering each question [9]

References

1. https://en.wikipedia.org/wiki/Latent_semantic_analysis
2. https://en.wikipedia.org/wiki/Statistical_significance
3. W. Xu, X. Sun, J. Hu, and B. Li, “REPERSP: Recommending Personalized Software Projects on GitHub,” in 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME), 2017, pp. 648–652.
4. M. Linares-Vásquez, A. Holtzhauer, and D. Poshyvanyk, “On automatically detecting similar Android apps,” in 2016 IEEE 24th International Conference on Program Comprehension (ICPC), 2016, pp. 1–10.
5. J. Crussell, C. Gibler, and H. Chen, “Scalable semantics-based detection of similar android applications,” in ESORICS’13, 2013.
6. S. Li, S. Hanna, L. Huang, E. Wu, C. Chen, and D. Song, “Juxtapp and dstruct: Detection of similarity among android applications,” De- partment of Computer Science, The University of Auckland, Tech. Rep. UCB/EECS-2012-111, 2012.
7. K. Chen, P. Liu, and Y. Zhang, “Achieving accuracy and scalability simultaneously in detecting application clones on android markets,” in ICSE’14, 2014.
8. A. Desnos, “Android : Static analysis using similarity distance,” in HICSS’12, 2012, pp. 5394–5403.
9. “How long should a survey be? What is the ideal survey length?,” SurveyMonkey. Available: https://www.surveymonkey.com/curiosity/survey_completion_times/.