HW6: Paper from 2014

i) *Reference*:

Alessandra Gorla, Ilaria Tavecchia, Florian Gloss, Andreas Zeller
Checking App Behavior Against App Descriptions
Proceedings of the 36th International Conference on Software Engineering, 2014

ii) *Keywords*:

1) **Latent Dirichlet Allocation (LDA):** Using LDA on the app descriptions, CHABADA identifies the main *topics* ("theme", "map", "weather", "download") for each application.
2) **Compact Language Detector:** Detect the most common language in a document and delete text in all other languages.
3) **Stemming:** A common NLP technique to identify the word's root and make words such as "playing", "player", and "play" all match to the single common root "play".
4) **Elements Silhoutte:** The silhouette of an element is the measure of how closely the element is matched to the other elements within its cluster, and how loosely it is matched to other elements of the neighboring clusters.

iii) Paper Contents:

1) **Motivational Statements**: Checking whether a program does what it claims to do is a long-standing problem for developers and computers too. Whenever we install a new app, we run the risk of the app being "malware". The question thus is not whether the behavior of an app matches a specific pattern or not; it is whether the program behaves as advertised.
2) **Related Work**: (i) The author refer the work of Zhou and Jiang, who use permissions requested by applications as a filter to identify potentially malicious applications whereas CHABADA identifies outliers even without knowing what makes malicious behavior.
   (ii) Host and Ostwold learn from program corpora which verbs and phrases would normally be associated with specific method calls, and used these to identify misnamed methods. Pandita identify sentences that describe code contracts from more than 2,500 sentences of API documents. In contrast, CHABADA works on end-user documentation, which is decoupled from the program structure.
3) **Hypotheses**: (i) By clustering apps by description topics, and identifying outliers by API usage within each cluster, our CHABADA approach effectively identifies applications whose behavior would be unexpected given their description.
   (ii) CHADABA is set to highlight and point out sensitive data and its consequences to standard users in such a way that is easier to grasp for naïve audience who is mainly the victim of malware.
   (iii) We have identified several examples of false and misleading advertising; and as a side effect, obtained a novel effective detector for yet unknown malware.
4) **Future work**: CHADABA can used more advanced methods such as API interaction, flow of information between APIs to characterize what the app does. The state of the art in natural lan- guage processing can retrieve much more than just topics. Looking at dependencies between words (such as conjunc- tions, subject-verb, verb-object) could retrieve much more detailed patterns.

iv) Needs Improvement:

1) Since the authors rely on static API usage, they suffer from limitations that are typical for static analysis. In particular, they might miss behavior induced through *reflection,* i.e. code generated at runtime.
2) The sample of 22,521 apps is based on free applications only. I believe that not considering paid apps makes the dataset biased. Thus the results obtained are conservative and can be improved through a good fraction by including paid apps.
3) The authors extract the API calls statically, which may cause them to include API calls which are never executed by the app.