Estimating Mobile Application Energy Consumption Using Program Analysis

Shuai Hao, Ding Li, William G.J. Halfond, and Ramesh Govindan

University of Southern California

Motivation

- Smartphones are popular

- Batteries don’t last that long

- Many user complaints

Help app developers understand energy implications of their implementation choices

Related Approaches

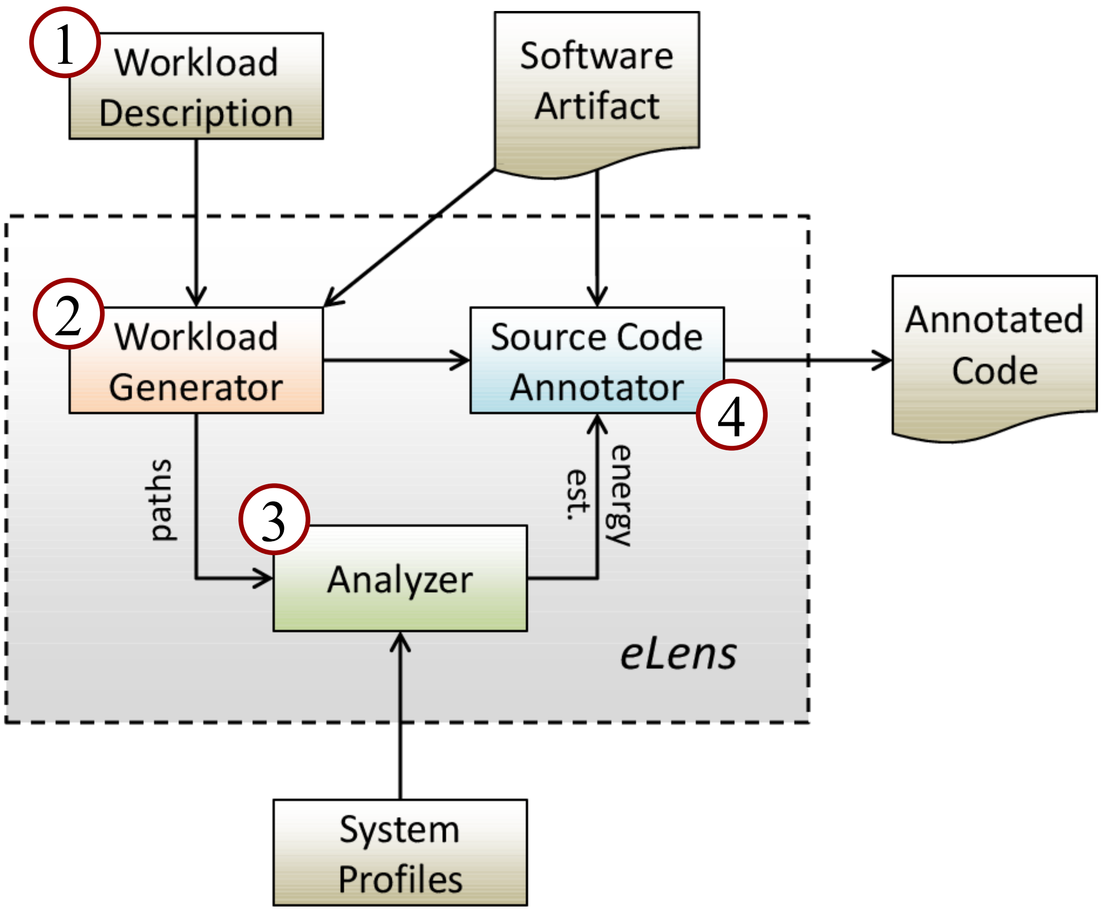
1. Underlying hardware/OS improvements
2. Cycle-accurate simulators
3. Field measurements
4. Whole program/method level feedback

eLens – Our Approach

Combine program analysis and per instruction cost modeling

1. Lightweight → no OS changes or specialized hardware required
2. Fine-grained → feedback at the *source* line level
3. Accurate → within 10% of ground truth
4. Fast → estimates within minutes

Overview of eLens



1. Generate workload
2. Identify corresponding executed paths
3. Compute power values for paths
4. Annotate source lines

Generating the Workload

Convert use cases to paths

- Use cases represent scenarios of interest to the developers
- Specified informally or formally
- Our approach: run instrumented version of the app and record runtime information

Estimating a Path's Energy

$$\textit{Energy} = \sum_{h \in Hardware} \sum_{i \in path} C_h(i)$$

- Cost functions (C_h) for each component (h)
- Instruction's energy cost is either:
  - – Path-independent: "fixed-cost" energy
  - – Path-dependent: varies based on path

- Cost functions provided by a Software Environment Energy Profile (SEEP)

Software Environment Energy Profile

Provides platform-specific energy parameters

- Enables rapid analysis for multiple platforms
- LEAP based profiling
  - Runs Android 3.2
  - Samples at 10KHz
  - Synchronization pulses
  - Multiple hardware components

Instructions: Path-Independent

"Fixed-cost" instructions

- Energy varies by hardware component and power state
- Profiled on LEAP node

1. Invokes/Returns
2. Load/Stores
3. Arithmetic/Logic
4. Stack management
5. Jumps/Branches
6. Fixed-cost APIs

Instructions: Path-dependent

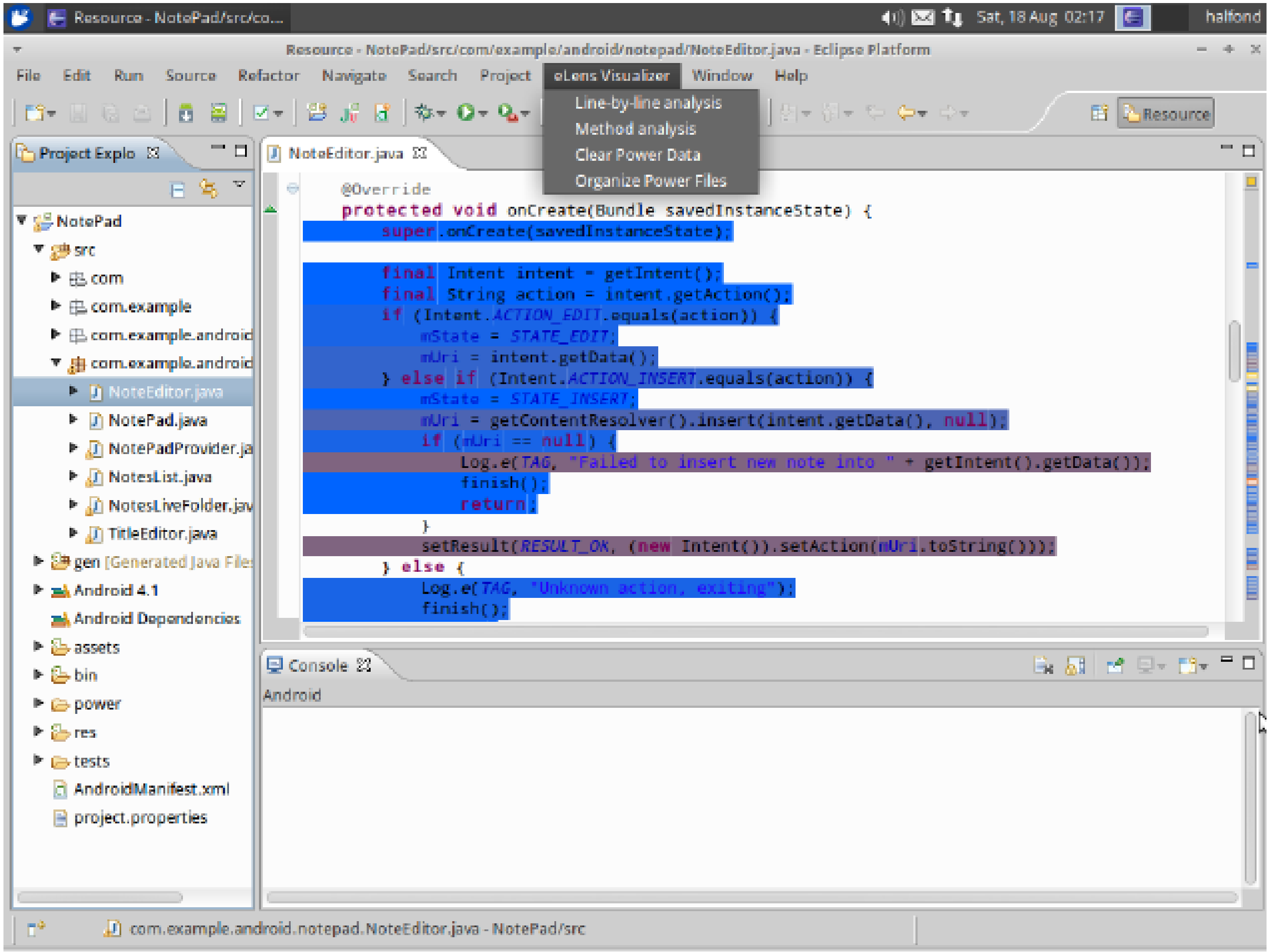
Based on information from other instructions in the path

- Four general categories:
  1. Array allocation
  2. Argument data
  3. Implementing class
  4. External data
- Propagate certain types of information along paths

Example Energy Calculation

ID	Instruction	Cost Functions (nJ)		
		CPU	RAM	WiFi
0	aload_0	1	1	.5
1	arraylength	1	1	.5
2	iconst_2	1	1	.5
3	if_icmpeq 14	2	1	.5
6	getstatic “Network.out”	2	1	.5
9	ldc “Usage…”	1	1	.5
11	invokevirtual “println”	20	10	10
14	return	1	1	.5

Visualization



Evaluation

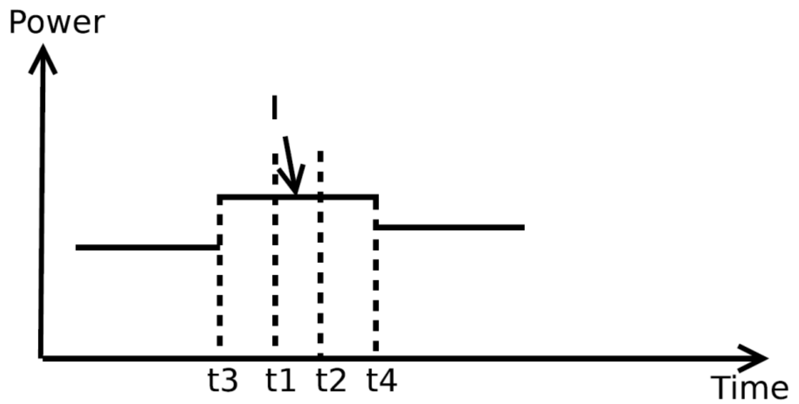
RQ1: What is the accuracy of the energy estimates?

RQ2: How much time is needed to estimate the energy consumption?

RQ3: Is time profiling equivalent to energy estimation?

Challenges to Obtain Ground Truth

1. Apps compatible with LEAP node
2. LEAP sampling interval
3. Idle time dominates execution time
4. Isolation of application energy

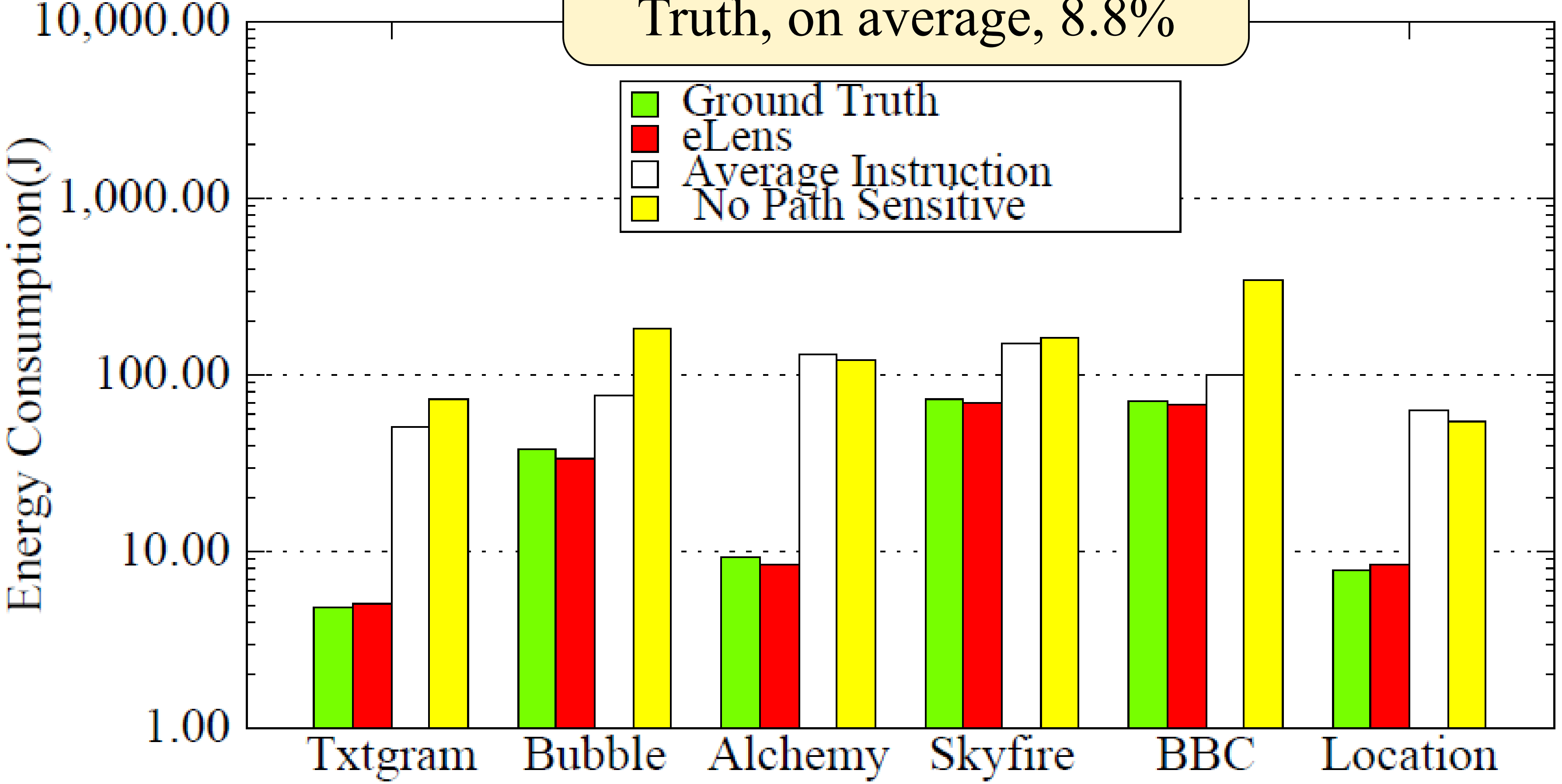


Subject Applications

App	Classes	Methods	Bytecodes	Description
BBC Reader	590	4,923	293,910	RSS Reader for BBC News
Bubble Blaster II	932	6,060	398,437	Bubble blasting game
Classic Alchemy	751	4,434	467,099	Science game
Location	428	3,179	232,898	Provide location
Skyfire	684	3,976	274,196	Web browser
Textgram	632	5,315	244,940	Text editor

Accuracy: App Level

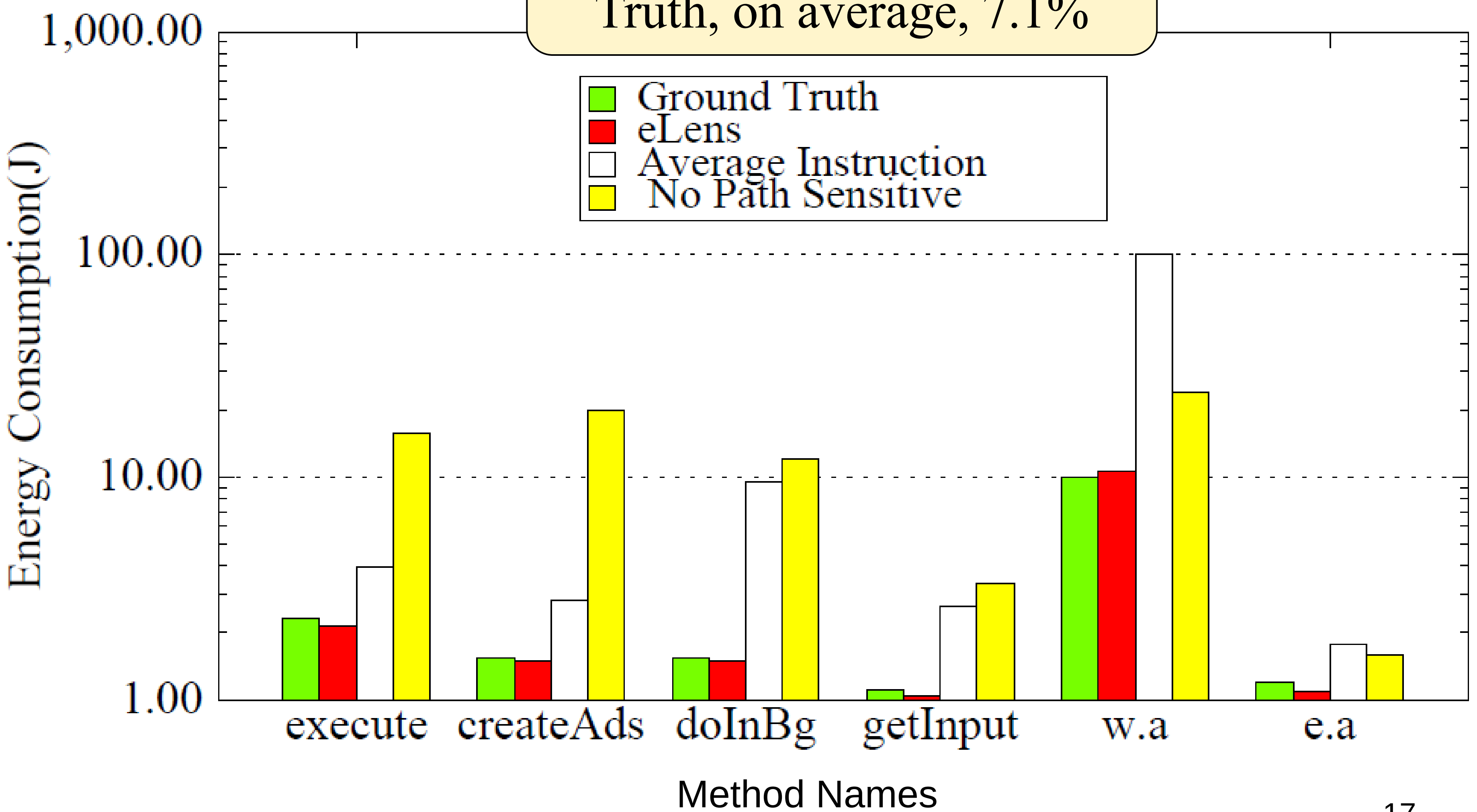
eLens differs from Ground Truth, on average, 8.8%



Subject Applications

Accuracy: Method Level

eLens differs from Ground Truth, on average, 7.1%



Accuracy: Hardware Components

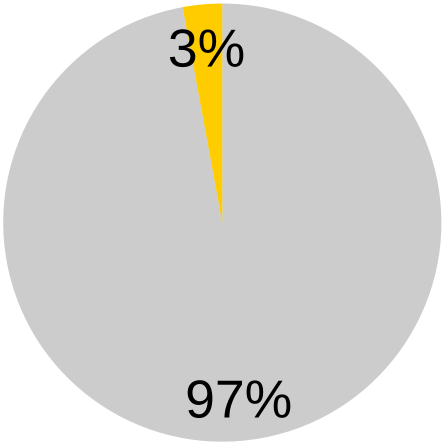
No more than 12% difference from Ground Truth

Application	Error Rate (%)			
	CPU	RAM	WiFi	GPS
BBC Reader	-6.2	5.9	-6.8	-
Bubble Blaster II	-11.5	3.5	-11.6	-
Classic Alchemy	-7.9	-6.9	-4.4	-
Location	-7.8	-8.4	-	8.1
Skyfire	-7.9	0.9	-8.4	-
Textgram	5.2	4.6	4.6	-

Runtime of eLens

Application	Runtime (s)	
	Instr.	Est.
BBC Reader	344	16
Bubble Blaster II	450	17
Classic Alchemy	886	17
Location	274	10
Skyfire	258	8
Textgram	269	6

Runtime Breakdown



Runtime ranged from 5 to 15 minutes

Is Time Equal to Energy?

Compare methods' time vs. energy

- Linear correlation?
  - → Pearson == 0
- Ranking similarity?
  - → Cosine similarity == .21

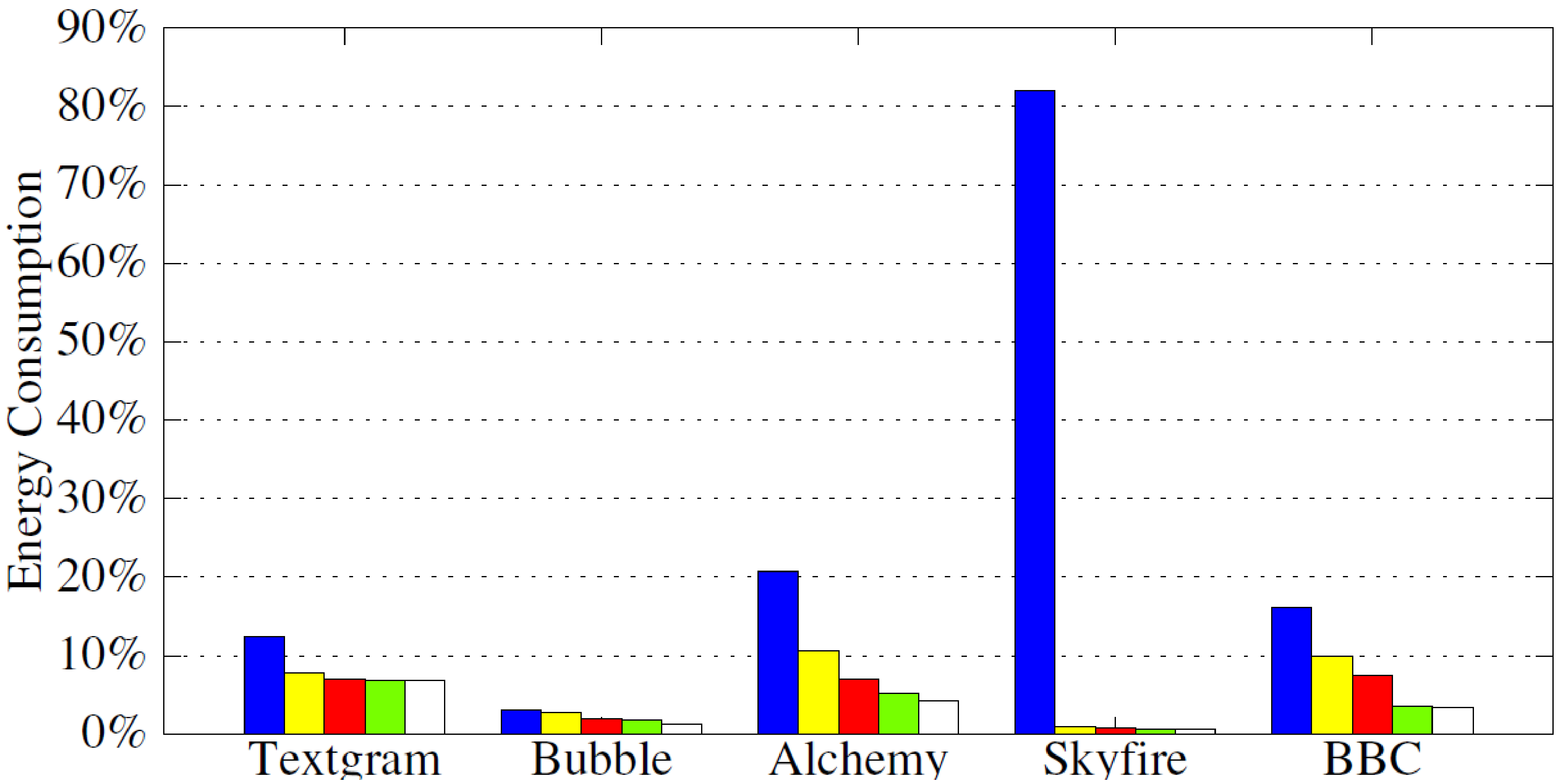
Strong indication of no linear relationship or ranking similarity.

Conclusions

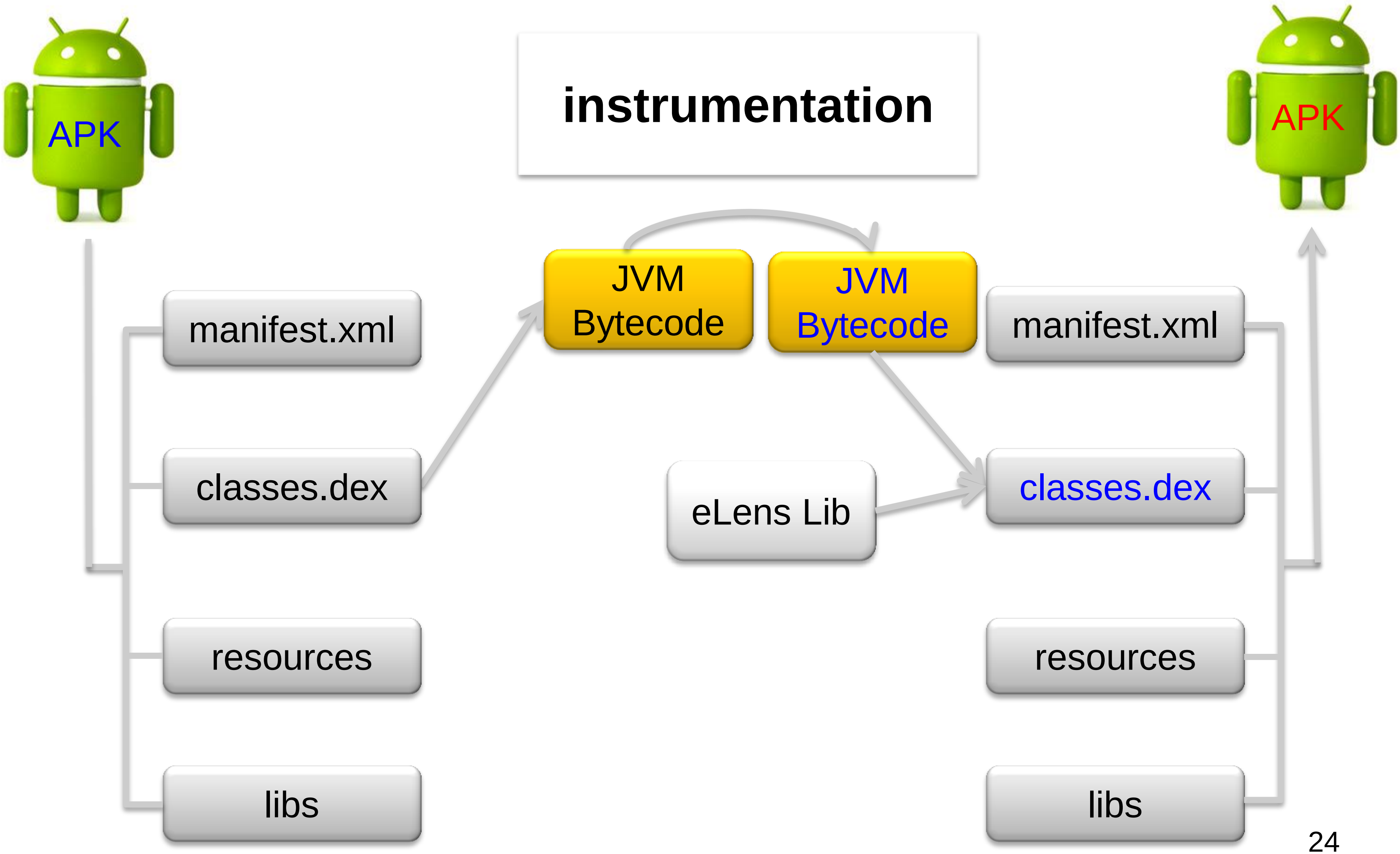
- eLens → estimate energy consumption
  - Uses program analysis and per-instruction cost modeling
  - Imposes minimal requirements on developer
- Evaluation
  - Accuracy was 8.8% at whole program level

Thank you

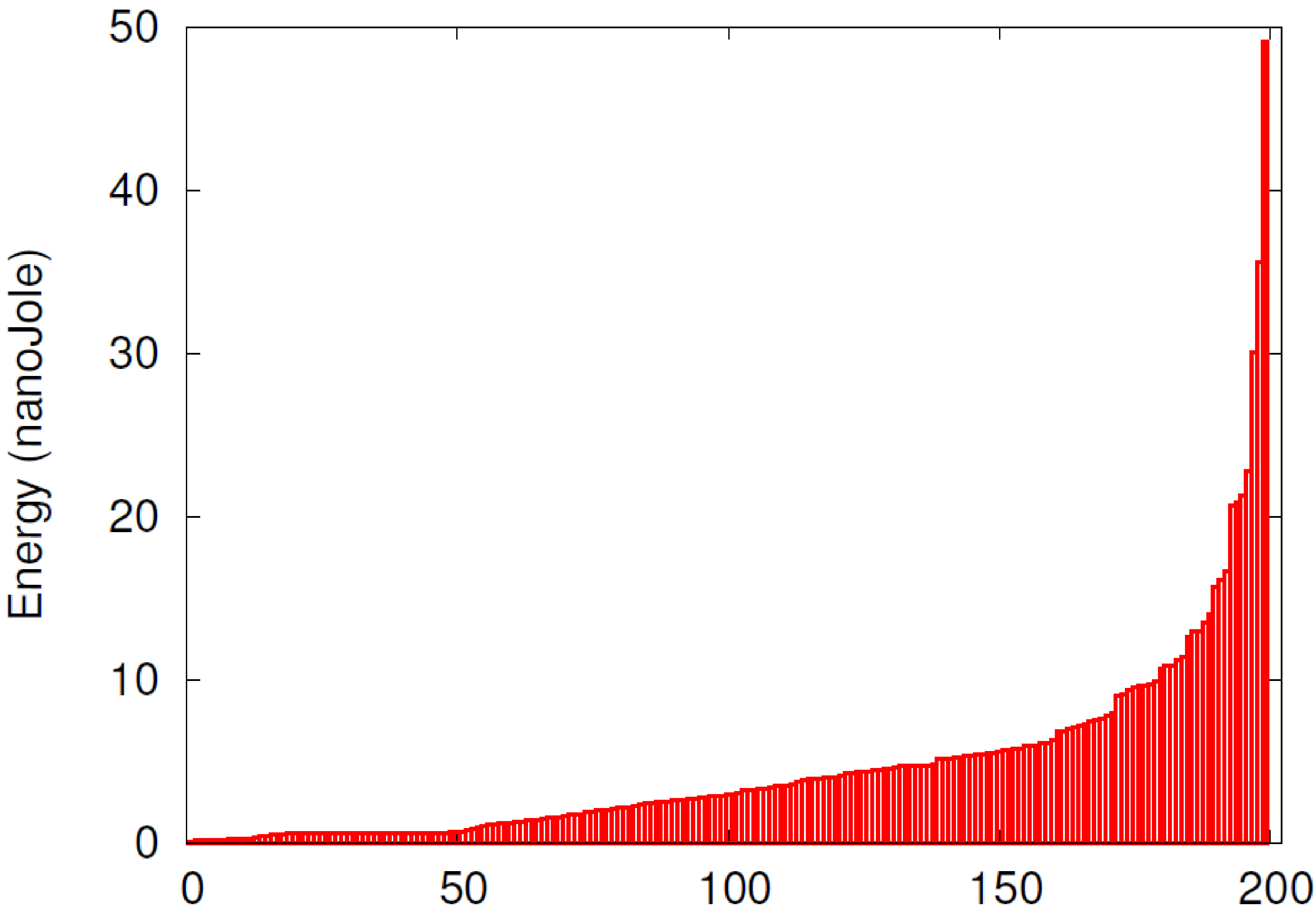
Case Study of Usefulness



Working with Market Apps



Bytecode Costs



Time vs. Energy