Christos' template - v14: METAPAPER: Spotting Fake Reviews in App Stores

first name
SCS CMU
first@cs.cmu.edu

second name
SCS CMU
second@andrew.cmu.edu

Christos Faloutsos
SCS CMU
christos@cs.cmu.edu

March 16, 2023

Abstract

How can you find strange nodes in a who-calls-whom graph? Spotting anomalies in graphs is an important topic. Our METAPAPER method has the following properties (a) *Scalability*, being linear on the input size (b) *Effectiveness*, spotting 90% of the anomalies in real data (c) *Parameter-free*, requiring no user-defined parameters. Experiments on *3GB* of real data from epinions.com illustrate the benefits of our method.

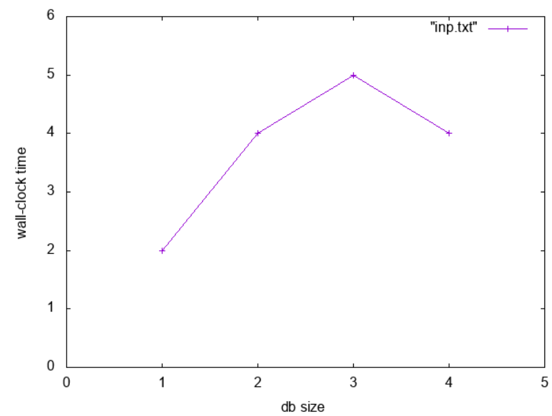

Figure 1: **METAPAPER wins**: Execution time for METAPAPER, on epinions.com

1 Introduction

Given a large count of reviews for products, how can one spot the fake ones. On line reviews are important. They are often faked, for monetary gain. How to spot the truth?

Here we propose METAPAPER, a method to spot fake reviews. The main idea behind our method is a principled way to merge several warning signals. Figure 1 shows the results of our method where METAPAPER outperforms the competition by up to 999%

The advantages of our method are

- **Scalability** : it scales linearly with the input size
- **Effectiveness**: it gives very good reconstruction error, on real data
- **Parameter-free** it requires no user-defined parameters.

Reproducibility: we publish our data and our code, at `www.cs.cmu.edu`

The outline of the paper is typical: we give the survey (Section 2), the proposed method (Section 3), experiments (Section 4), and conclusions (Section 5).

2 Background and Related Work

There is a lot of work on app reviews, and we group it in the following sub-sections.

2.1 Fraud detection

Bla-bla-bla - fake citation: [3] [2]

2.2 Anomaly detection

bla-bla - oddball, subdue etc [1]

However none of the above methods fullfils all the specs of our method: (a) scalability (b) effectiveness. Table 1 contrasts METAPAPER against the state of the art competitors.

Property \ Method	*method1*	*method2*	METAPAPER
Scalability			✔
Effectiveness			✔
Parameter-free			✔
other-stuff			✔

Table 1: **METAPAPER matches all specs**, while competitors miss one or more of the features.

3 Proposed Method

In this section we present the proposed method, we analyze it and provide the reader with several interesting -at least in our opinion- observations. Table 2 gives the list of symbols we use.

Symbols	Definitions
G	a graph
A	adjacency matrix

Table 2: Symbols and Definitions

3.1 Intuition

The main idea behind our METAPAPER is to exploit network effects: if we see a bi-partite core, then we suspect fraud.

Figure 2 and 3 illustrates the intution

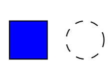

Figure 2: Sample tikz figure

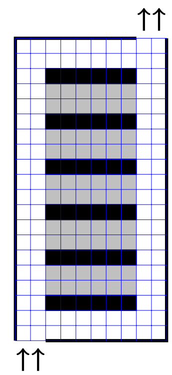

Figure 3: Another Sample tikz figure

3.2 Algorithm

Algorithm 1 shows the pseudo code and Algorithm 2 shows a variation

Data: a graph G
Result: the communities in G

```
initialization;
put graph G on stack;
pop stack;
while stack is not empty do
    process current graph;
    if has n^2 edges then
        add to output;
    else
        split in two;
        put both on stack;
    end
    pop stack;
end
```

Algorithm 1: FakeCom: Community detection algorithm

3.3 Complexity Analysis

Theorem 1 METAPAPER *requires time linear on the input size*

Proof 1 *from Eq *bla*, with lagrange multipliers*

3.4 SQL implementation

In fact, we can use SQL to implement our algorithm:

Data: this text
Result: how to write algorithm with LaTeX2e
initialization;
while *not at end of this document* **do**
    read current;
    if *understand* **then** `// nice!`
        go to next section;
        current section becomes this one;
    else
        go back to the beginning of current section;

Algorithm 2: How to write algorithms

```
select name, ssn
from student
where name = 'smith'
  and grade = 'A';
```

4 Experiments

Here we report experiments to answer the following questions

Q1. **Scalability**: How fast is our METAPAPER

Q2. **Effectiveness**: How well does METAPAPER work on real data?

The graphs we used in our experiments are described in the table 3.

4.1 Q1 - Scalability

In figure 4 we present the experimental results for the real-world datasets we used.

4.2 Q2 - Effectiveness

In Figure bla, we show the precision/recall of METAPAPER for the epinions.com dataset. Notice that

4.3 Discussion - practitioner's guide

Given the above, we recommend that practitioners choose for the parameter values of METAPAPER, and for their datasets.

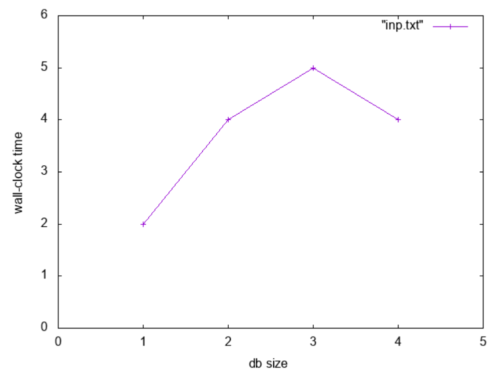

Figure 4: **METAPAPER scales lineary**: response time vs input size

4.4 Discoveries - METAPAPER at work

Thanks to our method, we processed real data and noticed the following observations

Observation 1 METAPAPER *works better than expected, on real data*

The reason is that we assumed uniformity, while real data have skewed distributions (Pareto-like), and thus favor our approach.

Observation 2 METAPAPER *works faster than expected*

bla-bla

5 Conclusions

We presented METAPAPER, which addresses the fake-review problem, using network effects. The main idea is to spot anomalous graph substructures, using belief propagation

The advantages of the method are

1. **Scalability**: it scales linearly with the input size, as shown in Figure 4 and Lemma
2. **Effectiveness**: it gives excellent precision, on real world data
3. **Parameter-free**: it requires no user intervention - METAPAPER sets all parameters to reasonable defaults, and it is insensitive to the exact choices anyway

We also presented experiments on *3GB* of real data, where METAPAPER outperformed the competitors

Nodes	Edges	Description
Real-world Networks		
13,579	37,448	AS Oregon
23,389	47,448	CAIDA AS 2004 to 2008

Table 3: Summary of real-world networks used.

by 10 percentage points of accuracy, and 3x faster execution time.

Reproducibility: We have already open-sourced our code, at `www.cs.cmu.edu`

Acknowledgements: We would like to thank Christos Faloutsos for his *MetaPaper* list of suggestions on the presentation.

References

[1] L. Akoglu, M. McGlohon, and C. Faloutsos. oddball: Spotting anomalies in weighted graphs. In *PAKDD (2)*, volume 6119 of *Lecture Notes in Computer Science*, pages 410–421. Springer, 2010.

[2] C. Faloutsos. Signature files. In *Information Retrieval: Data Structures & Algorithms*, pages 44–65. 1992.

[3] H. Tong and U. Kang. Big data clustering. In *Data Clustering: Algorithms and Applications*, pages 259–276. 2013.