Who’s Who on Gnome Mailing Lists: Identity Merging on a Large Data Set

Erik Kouters

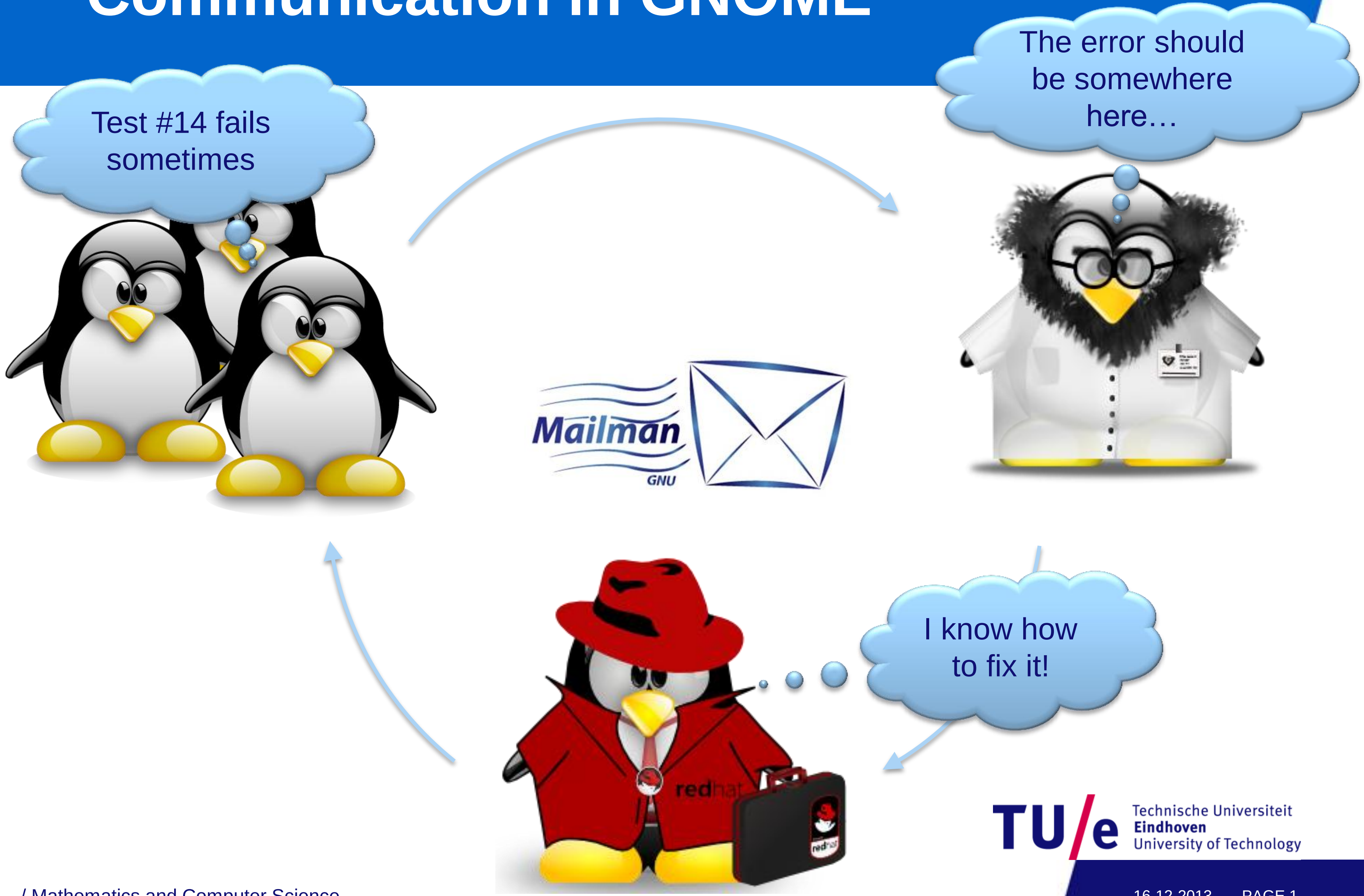
Bogdan Vasilescu

Alexander Serebrenik

Technische Universiteit
Eindhoven
University of Technology

Where innovation starts

Communication in GNOME



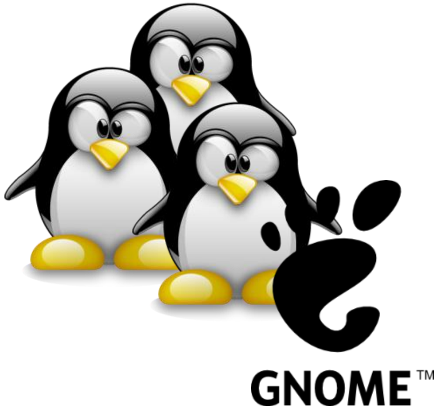
Test #14 fails sometimes

The error should be somewhere here…

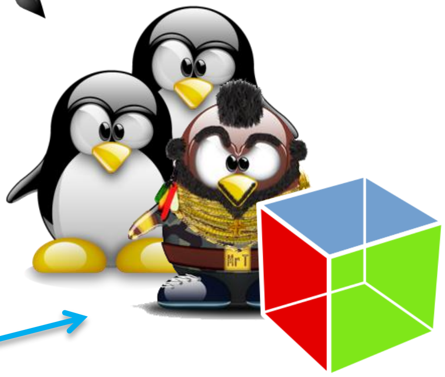
I know how to fix it!

The Problem

- **Contributors use different names, email addresses**



<"Mr. T", baracus@gmail.com>



<"Bosco Albert 'B. A.' Baracus", ba.baracus@yahoo.com>

The Data

github SOCIAL CODING

8618 aliases
4989 individuals

Mailman GNU

77,081 aliases
61,748 individuals

Differences

Names:

- Bosco Albert Baracus
- Baracus Bosco Albert
- B.A. Baracus
- B.A.
- B.A. Barracus
- Bosco A. Baracus
- Bosco “B.A.” Baracus
- Mr. T

Emails:

- b.baracus@domainA
- b.a.baracus@domainB
- b DOT baracus AT domainC
- bbaracus@domainD
- bosco@domainE

Identity merge algorithms:

- The “noisier” the data, the worse they perform

Large Data Set

- Boy George
- George Michael
- Michael Jackson
- Jackson …

- The larger the data, the more overlap in names

Technische Universiteit
Eindhoven
University of Technology

Scalability

- **Performance of identity merging algorithm?**

Existing Algorithms

Simple Algorithm – Goeminne & Mens (2011)

< B.A. Baracus, b.a.baracus@domainA >
< B.A. Baracus, mister_t@domainB >
✓

< B.A. Baracus, b.a.baracus@domainA >
< B. Baracus, mister_t@domainB >
✗

< Bosco Baracus, bosco@domainA >
< Bosco Doe, bosco@domainB >
✗

Existing Algorithms

Bird’s Algorithm – Bird et al. (2006)

< Bosco Baracus, b.a.baracus@domainA >
< B.A., bbaracus@domainB > ✓

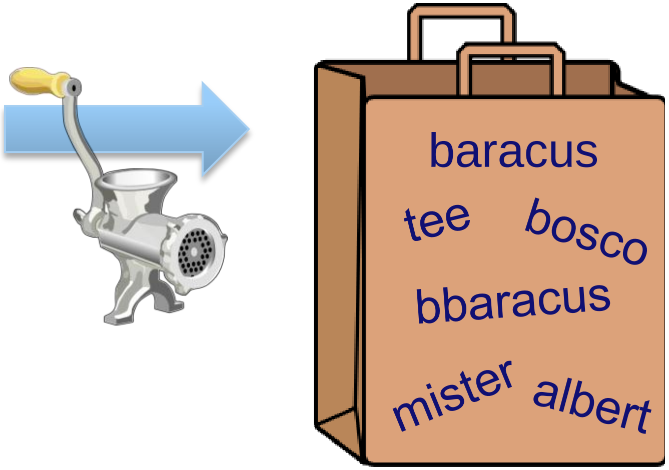
< Bosco Baracus, b.a.baracus@domainA >
< Bosco Baracuda, albert@domainB > ✗
Baracus ~Levenshtein Baracuda

< Bosco Baracus, b.a.baracus@domainA >
< Baracus Bosco, mister_t@domainB > ✗
Bosco !~Levenshtein Baracus

Introduced Algorithm

<Bosco Albert Baracus, bbaracus@domainA>
<Mister Tee, bbaracus@domainA>

bbaracus@domainA:



	bbaracus@...	mrt@...		
bosco	1	..	..	..
albert	1	..	..	..
bbaracus	1	..	..	..
babaracus	8/9	1	..	..
mister	1	..	..	..

bbaracus ~Levenshtein babaracus = 8/9

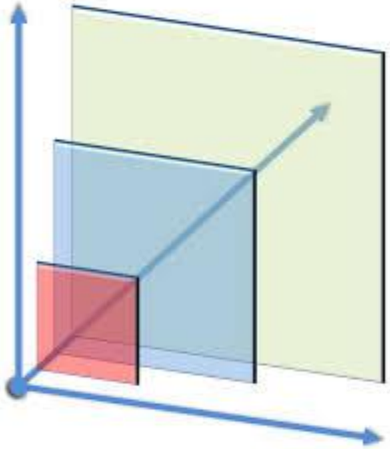
Introduced Algorithm

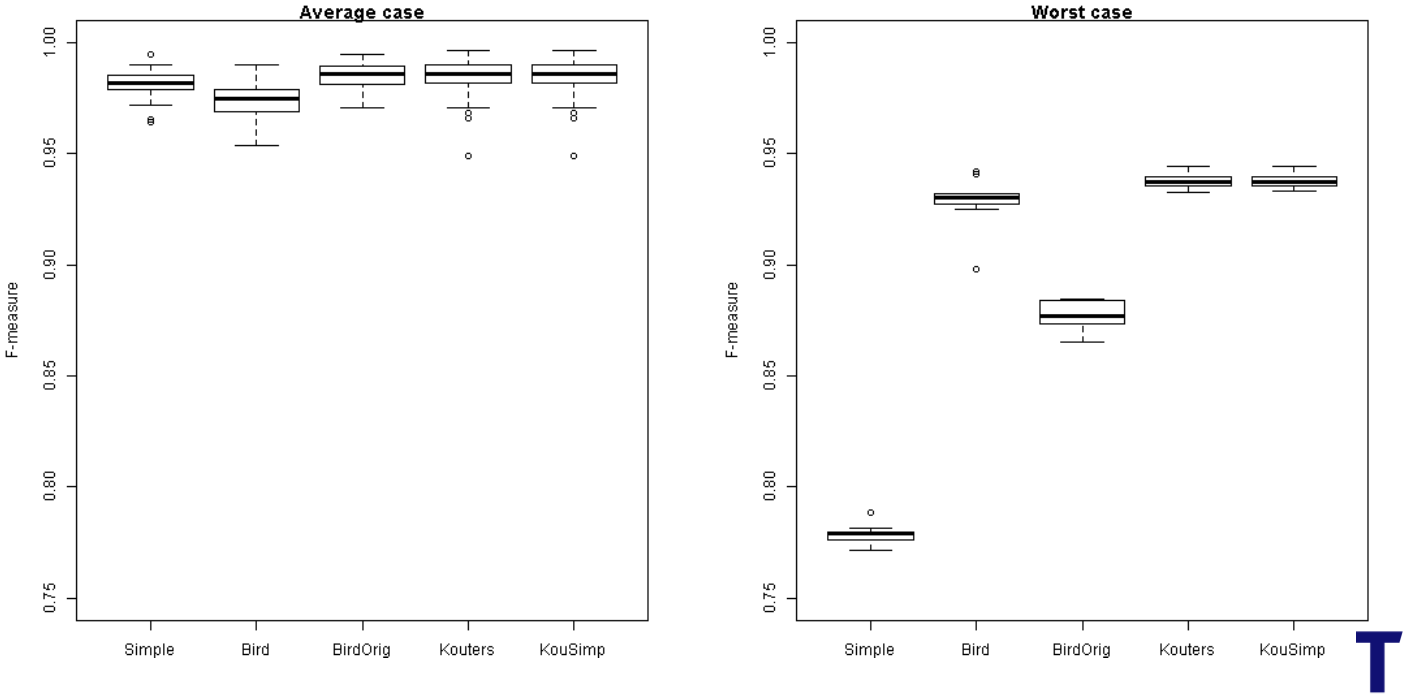
Skywalker
Skywalker
Luke
Luke
Luke
Luke
Luke
Luke

- Common names are weighted down

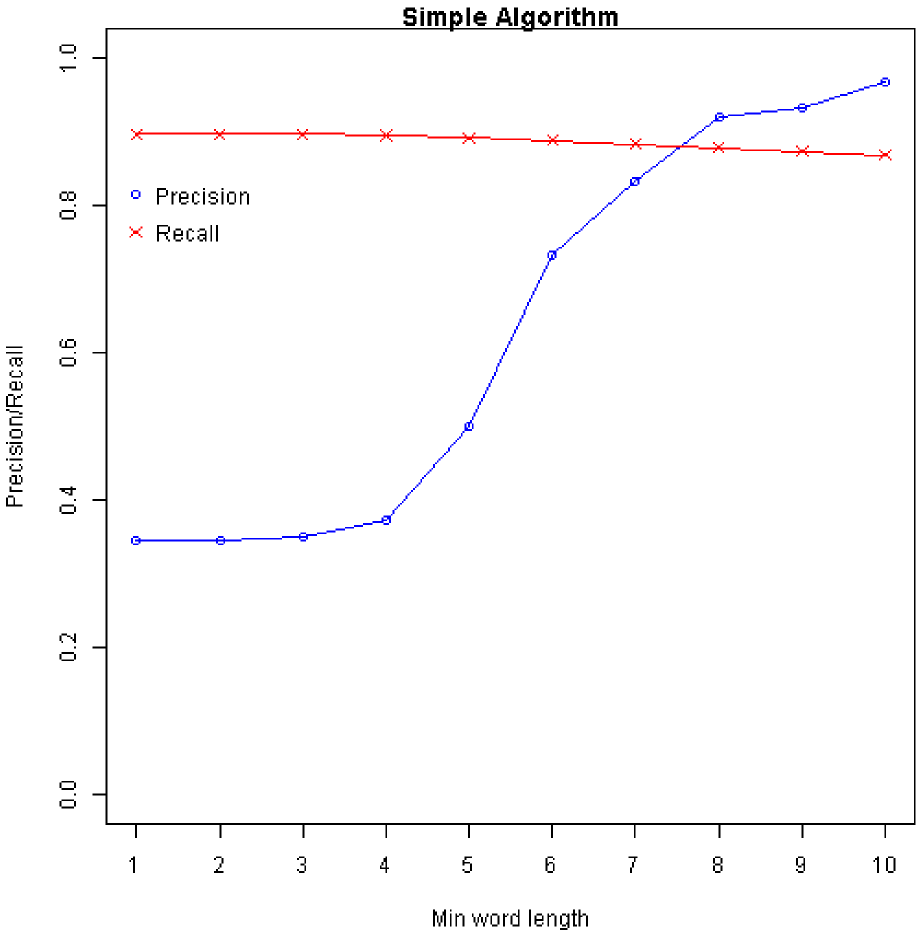
Introduced Algorithm

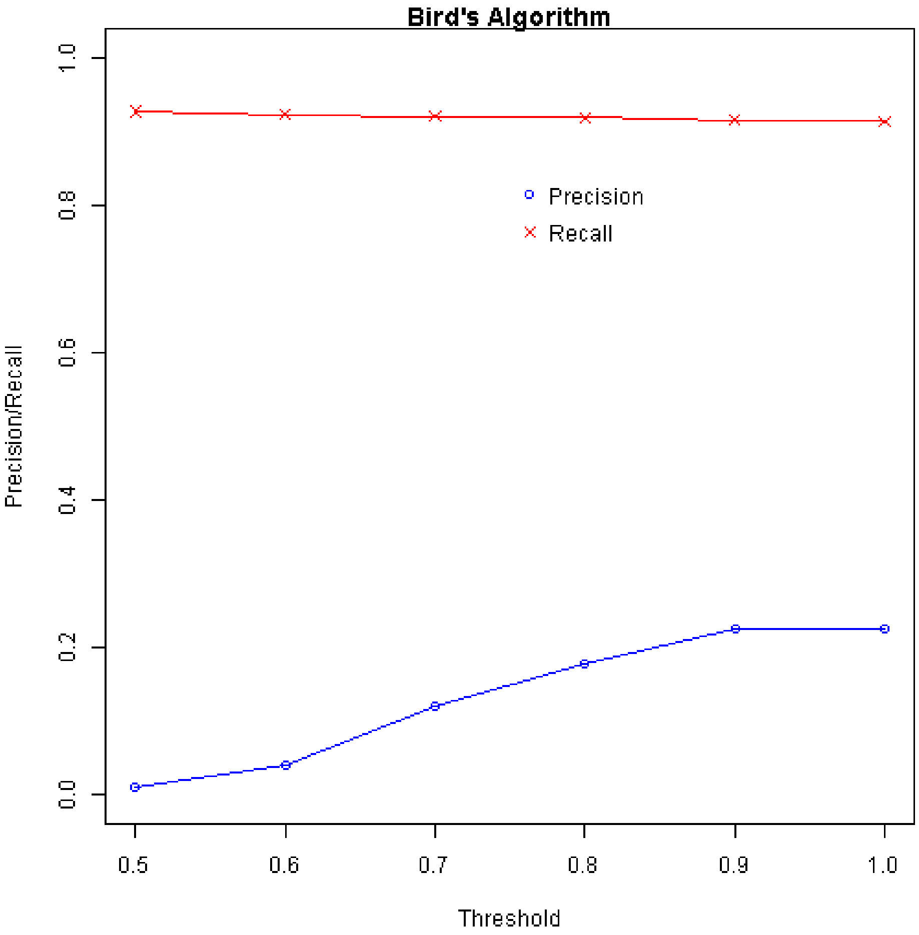
- **ICSM ERA 2012**
  - **Data set: Git logs**
  - **Singular Value Decomposition (SVD)**
    - **Remove noise**



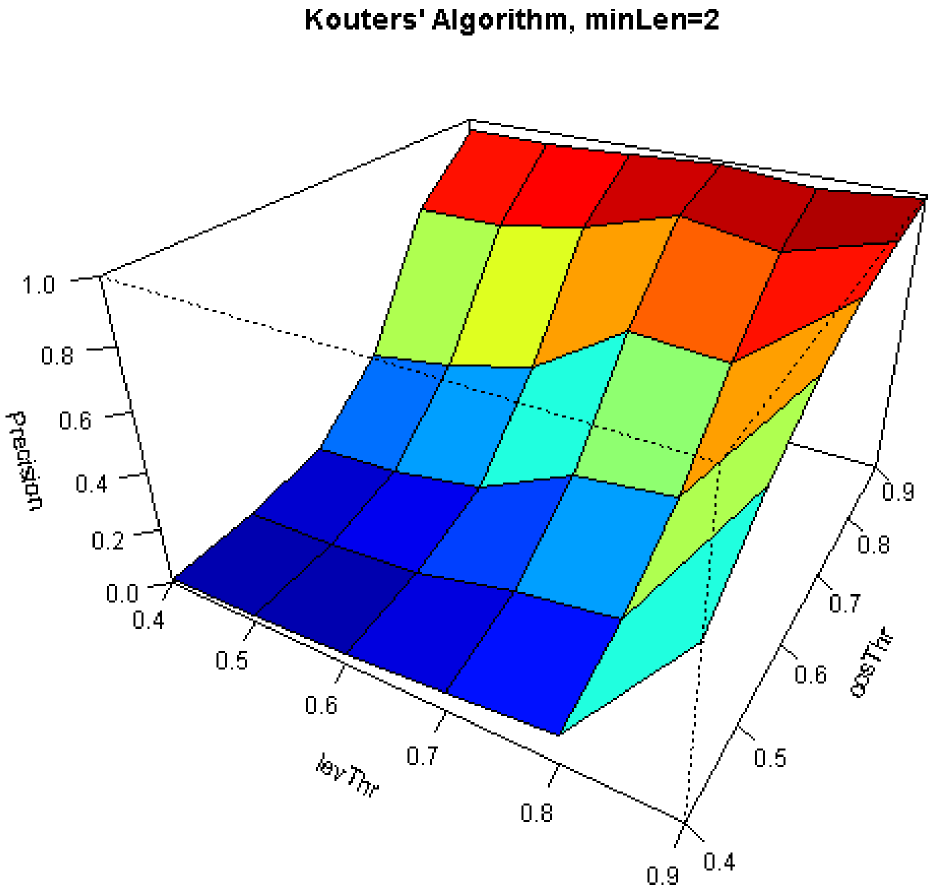


Results

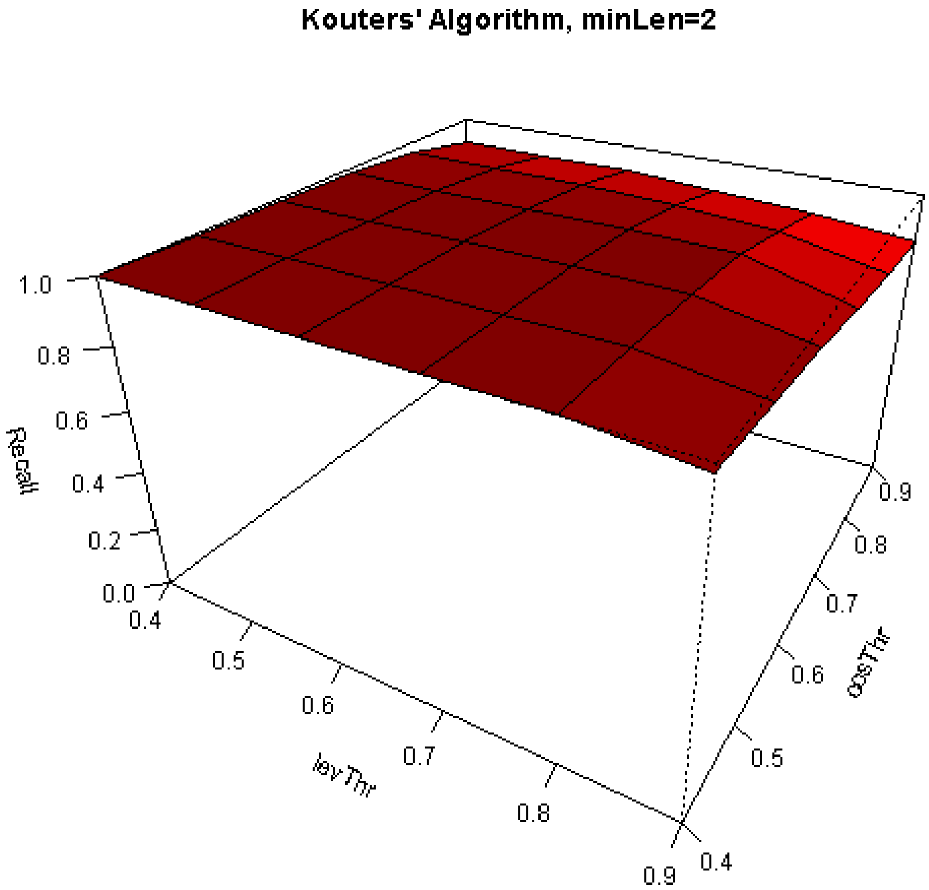

Simple Algorithm
Precision/Recall
0.0
0.2
0.4
0.6
0.8
1.0
Precision
Recall
1
2
3
4
5
6
7
8
9
10
Min word length


Bird's Algorithm
Precision/Recall
0.0
0.2
0.4
0.6
0.8
1.0
Precision
Recall
0.5
0.6
0.7
0.8
0.9
1.0
Threshold

Results



Kouters' Algorithm, minLen=2
Precision
1.0
0.8
0.6
0.4
0.2
0.0
levThr
0.4
0.5
0.6
0.7
0.8
0.9
qgsThr
0.4
0.5
0.6
0.7
0.8
0.9



Kouters' Algorithm, minLen=2
Recall
1.0
0.8
0.6
0.4
0.2
0.0
levThr
0.4
0.5
0.6
0.7
0.8
0.9
qgsThr
0.4
0.5
0.6
0.7
0.8
0.9

Conclusions

- Trade-off between precision and recall
- Simple Algorithm
  - High precision, average recall
  - Despite simple heuristics scales well
- Bird’s Algorithm
  - Low precision, average recall
  - Scales badly due to complex heuristics
- Kouters’ Algorithm
  - High precision, average recall OR
  - Average precision, high recall
  - Scales well