# Winndow Management using AI

Björn Linse and Axel Goteman

EDAN05: Intelligent Systems

Computer Science, LTH, Lund University

May 22, 2014

# The Idea

- Better window management using machine learning techniques.
- More efficient window switching
- Automatic grouping of windows

# Possible Methods

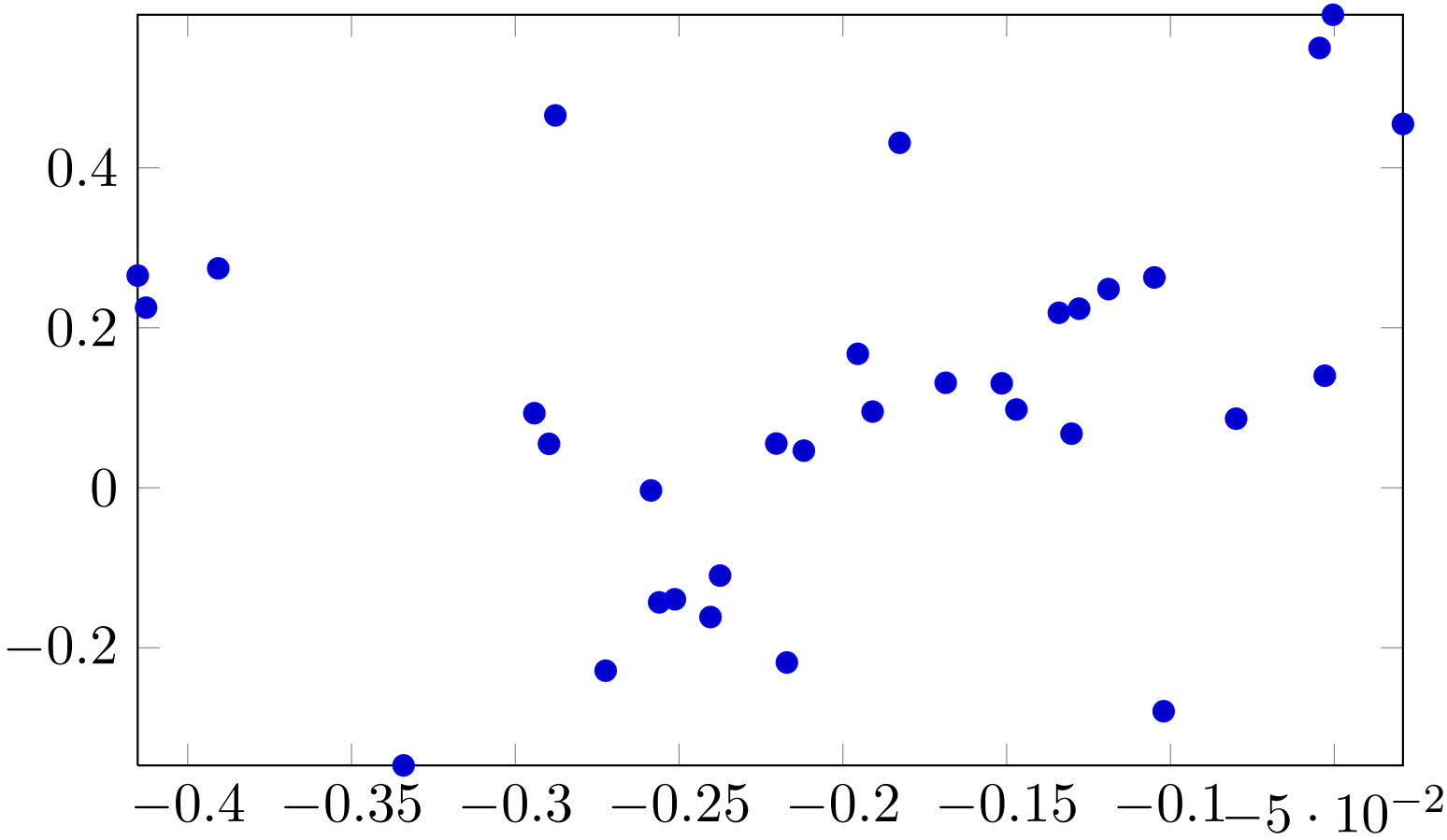
- Use clustering to detect window groups
- Predict window switching through feature extraction.

# Data collection

- We wrote a logger that kept track of all open windows
- collected data:
  - Window creation, switching and closing
  - Window title
  - Application names
  - Window position and size
  - Workspaces
- 7000 window switches in 30+ sessions on 2 machines.

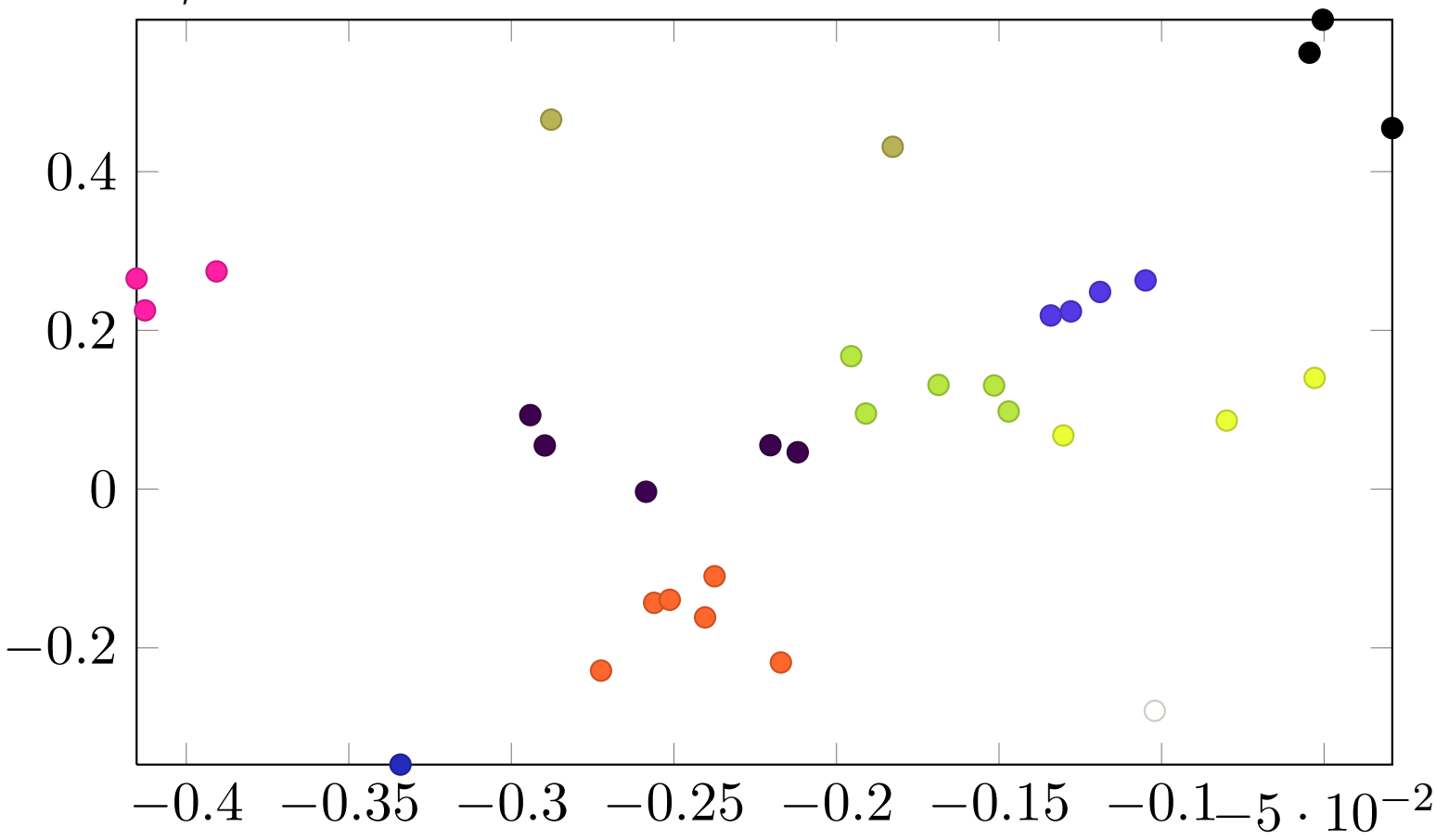
# Clustering

- Embed all windows in an $n$-dimensional graph
- Apply clustering algoritm like $k$-means



# Standard clustering methods

- We tried standard clustering methods like
- $k$-means, DBSCAN



# Rule-based Clustering

- Rules to cope with low amounts of data.
- Works in our case, but not (necessarily) in general
- In our case: strong pairs
  - editor + terminal
- Use bigram count:

$$S(w_i|w_j) = C(w_i|w_j) + C(w_j|w_i)$$

# Rule-based Clustering

## It works quite well (in one case!):

```
URxvt: % ~/LTH/mcore/cell
Gvim: dataflow.c (~/LTH/mcore/cell/spu) ((1) of 2) - GVIM1
URxvt: % ~/LTH/algimp/proj
URxvt: [power] ~/mcoretest
Gvim: twosys_fm.c (~/LTH/algimp/proj) - GVIM2
URxvt: # ~bjorn/LTH/mcore
URxvt: # /home/bjorn
===============
URxvt: % ~/LTH/intsys
Gvim: model.py (~/LTH/intsys) - GVIM
Chromium: New Tab - Chromium
URxvt: # /home/bjorn/.local/wmlogs
Chromium: fileadmin.cs.lth.se/cs/Education/EDAF15/labs/lab4/index.html - Chromium
===============
Chromium: Computer Science: Labs - Chromium
ScrollPipeViewer: ScrollPipeViewer V 1.38
ScrollPipeViewer: ScrollPipeViewer V 1.38
===============
Gvim: pres.tex (~/LTH/intsys/pres) - GVIM5
URxvt: % ~/LTH/intsys/pres
Evince: pgfplots.pdf - Package PGFPLOTS manual
```

# Feature switching

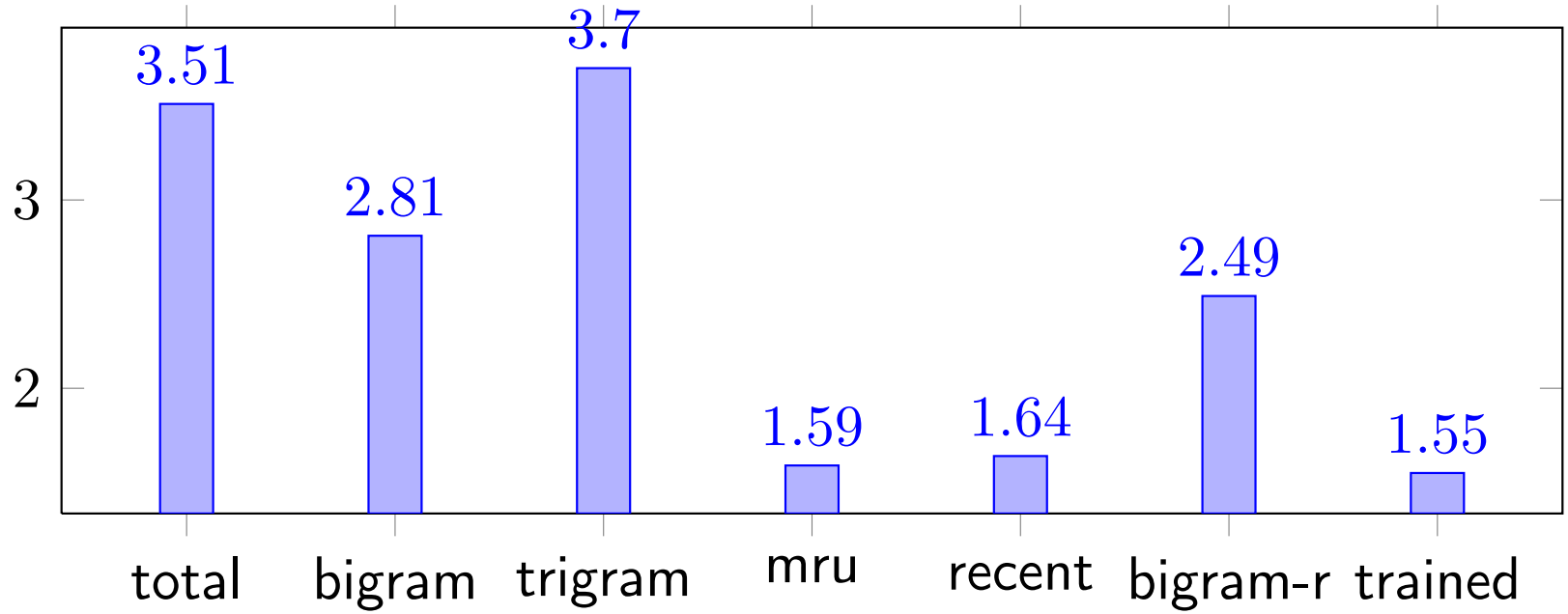
- Ranking by linear combination of features:

$$P(w_i) = \sum_j q_j r_j f_j(w_i)$$

- Features:
  - Bigram probabilities: $P(w_i|w_j)$
  - Total number of switches $N(w_i)$
  - Time-variant versions of the above "recent" and "bigram-recent"
  - Most-recently used
- Weight adjusted by "relevancy factor" $r_j$ (amount of data)

# Evaluation: Feature switching

- We trained the system on the logs:




- no significant improvement over MRU
- Bigram model not good (not enough data)

# Conclusion and Future possibilities

- Hard to predict switching better than MRU
- General clustering algoritms require more data

- Collect more data (from separete users)
- Investigate clustering methods more rigorously