Novel Applications of AI Techniques in Database Management

Johannes Wünsche

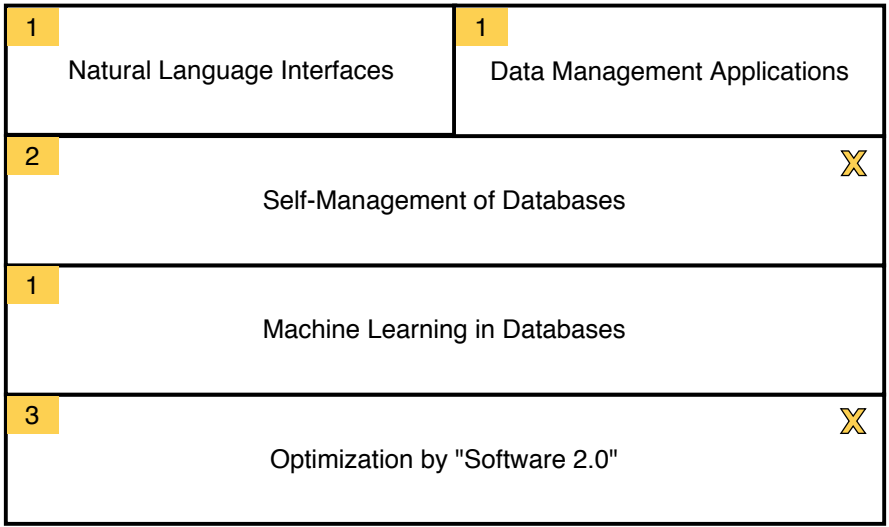
Faculty of Informatics, Otto-von-Guericke University, Magdeburg

Agenda

1. Overview
2. Short Summary of Natural Language Interfaces, ML in DB and Data Managment Applications
3. Self-Management of Databases
4. Software 2.0

Overview

Overview


1
Natural Language Interfaces
1
Data Management Applications
2
Self-Management of Databases
X
1
Machine Learning in Databases
3
Optimization by "Software 2.0"
X

Short Summary of Natural Language Interfaces, ML in DB and Data Managment Applications

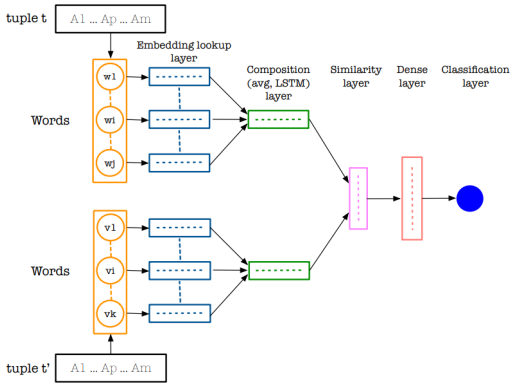
Natural Language Interfaces & ML in DB

Natural Language Interfaces

- Idea of using AI to interpret NL questions or requests and react accordingly
- quite old concept improved by modern hardware and technology

Machine Learning in Databases

Data Management Applications



Entity Resolution

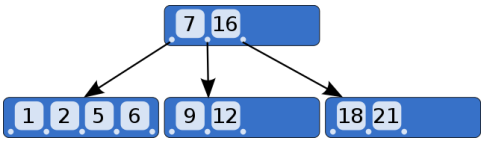
- Finding of Records that refer to the same entity
- Required if shape of data is not unitary

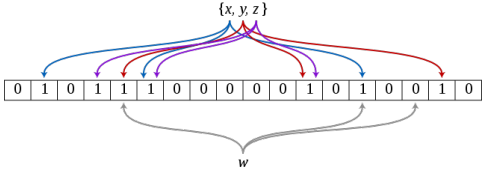
Self-Management of Databases

Self-Management of Databases

- *Tuning of parameters of Database Management Systems*
  - like cache amount and frequency of writing to storage
  - implementations like OtterTune from Database Research Group at Carnegie Mellon University
- *Elastic Scaling of Machine Allocation*
  - avoid latency spikes by action prediction through time-series prediction
  - implementations like P-Store by Taft, MIT
- *Learned Index Structures*
  - a "model can learn the sort order or structure of lookup keys and use this signal to effectively predict the position or existence of records"
  - alternative technology to exisiting Bloom-Filters or B-Trees

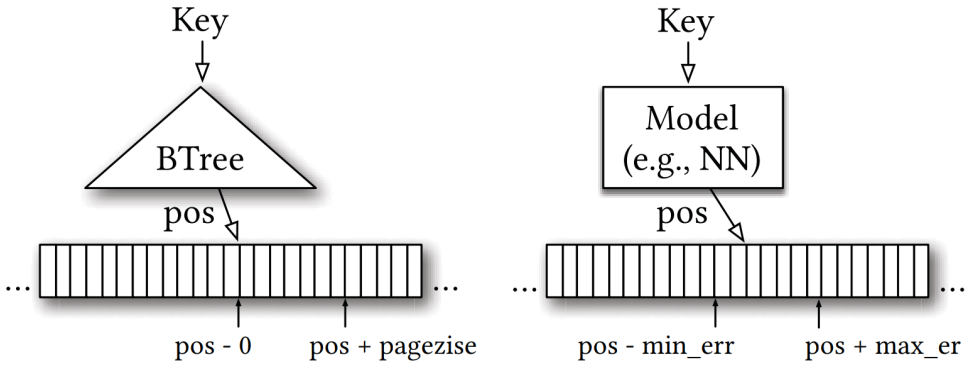
B-Tree Bloom-Filter





Learned Index Structures

What's the difference between Learned Index Structures and existing Models?



Learned Index Structures by Example

Software 2.0

Intro

- Classical Software Dev major downsides like difficult optimization of code and human error
- S2.0 doesn't base on declarative programming and tries to learn the desired functionality with a approximate base net
- enabled by development of Neural Networks in last 20 years allowing >100 layers deep networks
- program space is restricted for future training (backpropagation, gradien descent)
- many real world problems easier to detect desirable behaviour than to write a specific program

Advantages

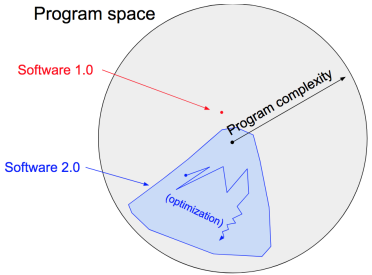
Higher Portability

- Smaller operation set
- Matrix Multiplication and thresholding at zero required
- small instruction set of chips with pretrained nets allows for

Better Performance

- allows for better performance and correctness predictions because closer implementation in hardware less core primitives are needed
- modules can be introduced to a single module reducing communication overhead by sacrificing clarity of separation, which is due to the human unlike nature of so2 sacrificed beforehand either way
- well trained neural nets outperform code implementation

Advantages



Better Runtime Predictability

- requires same amount of memory each iteration $\rightarrow$ low probability of infinite loops or locks
- speed well adjustable $\rightarrow$ speed can easily improved by reducing performance

Disadvantages

Unintuitivity

- can be treated as different new paradigm $\rightarrow$ requires rethinking of development style
- even though the network may work well, for humans difficult to understand
- developing in so2 is unintuitive and not well developed
- requires manually curating, maintaining, cleaning and labelling of datasets
- may not be applicable easily to all problems

Disadvantages

Nonrecognizable errors

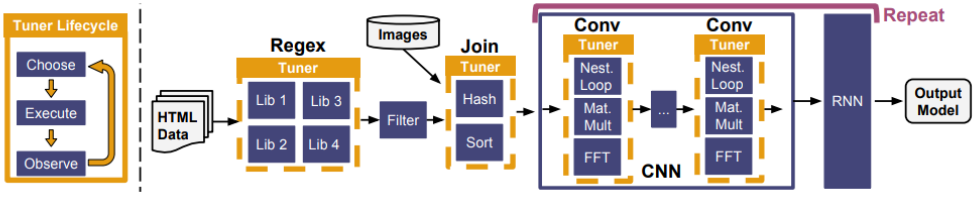
- errors may occur unpredictable
- can silently fail due to changed biases (hard to track since a large amount of them are being trained)

Lack of Tools

- no tools exist that support the development process like IDE, highlighting and alike for classical software

Example

Cuttlefish



TODO: Conclusion

TODO

Thank you for your attention.

Do you have any questions? Ideas?
Be free to ask them.