Parallel Natural Language Processing Algorithms in Scala

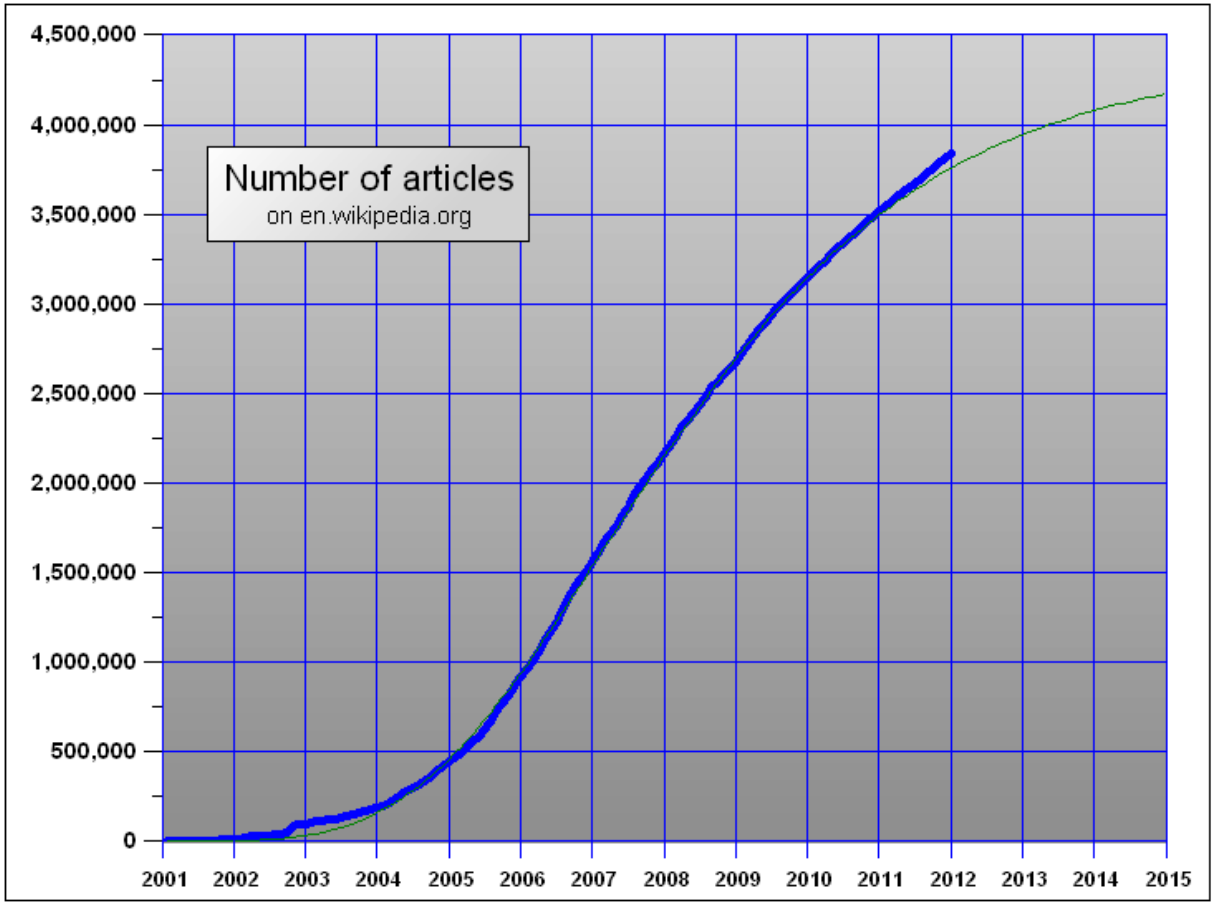
Stanislav Peshterliev

EPFL

January 24, 2012

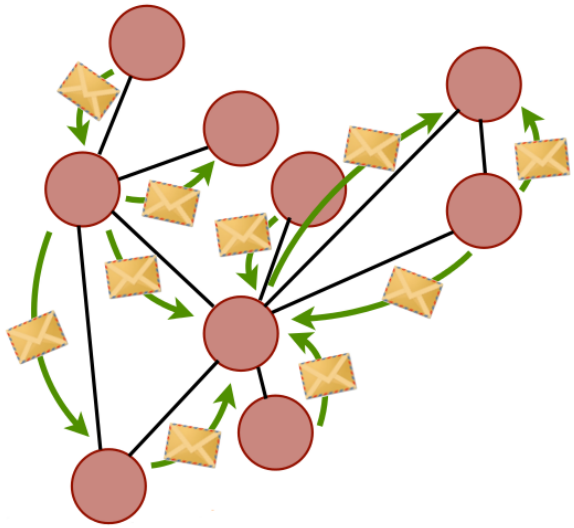
Motivation

- Rapid growth of data in natural language



Goals

- Parallization based on Menthor[Haller and Miller, 2011]



- Implement Maximum Entropy[Berger et al., 1996] and Naive Bayesian[Rennie et al., 2003]
- Benchmark two different parallelization strategies

Outline

- Motivation
- Goals
- Outline
- Text Categorization
- Classification Algorithms
  - Maximum Entropy
  - Naive Bayes
- Parallelization
  - Maximum Entropy
  - Naive Bayes
  - Strategy 1: Vertex for every sample
  - Strategy 2: Vertex for set of samples
- Experimental results
  - Data sets
  - Benchmarks

Text Categorization

- Determine the category of a document
- Predefined categories and training data set
- Spam filtering - spam or not spam

Classification Algorithms - Maximum Entropy

- The probability should be as uniform as possible, i.e. have maximum entropy

$$P_\Lambda(c|s) = \frac{1}{Z(s)} \exp(\sum_i \lambda_i f_i(s,c)) \quad (1)$$

- Improved Iterative Scaling
  - **Inputs:** Set S of labeled samples and a set of feature functions f_i.
  - Estimate expected value of f_i on the training samples.
  - Initialize all the parameters λ_i's to be zero.
  - Iterate until convergence:
    - Calculate the expected class labels for each sample with $P_\Lambda(c|s)$
    - For each parameter λ_i: Find $\delta_i = \frac{1}{M} \log \frac{\sum_{s \in S} f_i(s|c(s))}{\sum_{s \in S} \sum_c P_\Lambda(c|s) f_i(s|c)}$, and set $\lambda_i = \lambda_i + \delta_i$
  - **Output:** A classifier that predicts the class label of sample s.

Classification Algorithms - Naive Bayes

- Based on the Bayes's Rule

$$P(C|S) = \frac{P(S|C)P(C)}{P(S)} = \frac{P(S|C)P(C)}{\sum_{c \in C} P(D|C = c)P(C = c)}$$

- Probability estimation

$$P(c) = \frac{N_c}{N}$$

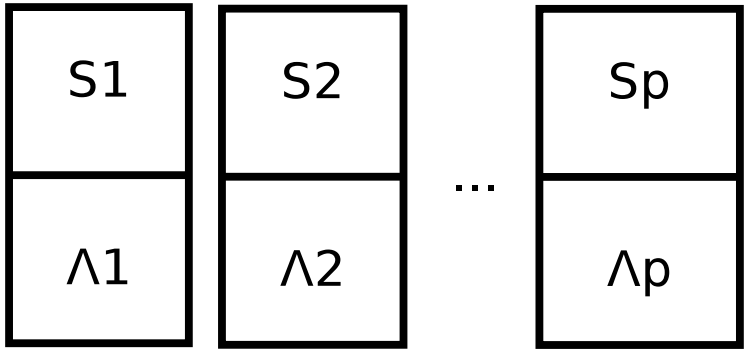
$$P(s|c) = \prod_{w} P(w|c)^{tf_{w,s}}$$

$$P(w|c) = \frac{tf_{w,c}}{|c|}$$

  - N_c - # of samples that have class c
  - $tf_{w,s}$ - # of times term w occurs in sample s
  - $tf_{w,c}$ - # of times term w occurs in class c
  - $|c|$ - total number of terms in class C

Parallelization - Maximum Entropy

- Mixture Weight Method [Mann et al., 2009]

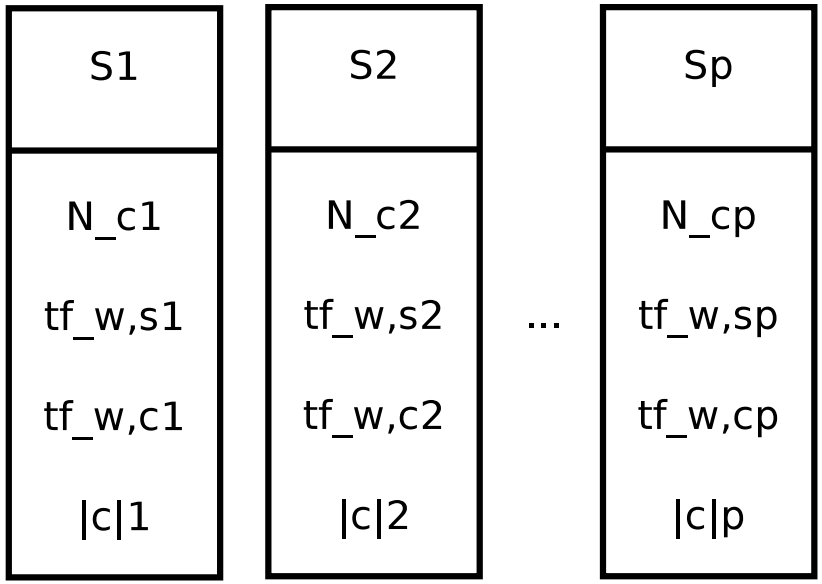


$$\Lambda_\mu = \sum_{k=1}^{p} \frac{1}{p} \Lambda_k$$

- Not good for small training sets

Parallelization - Naive Bayes

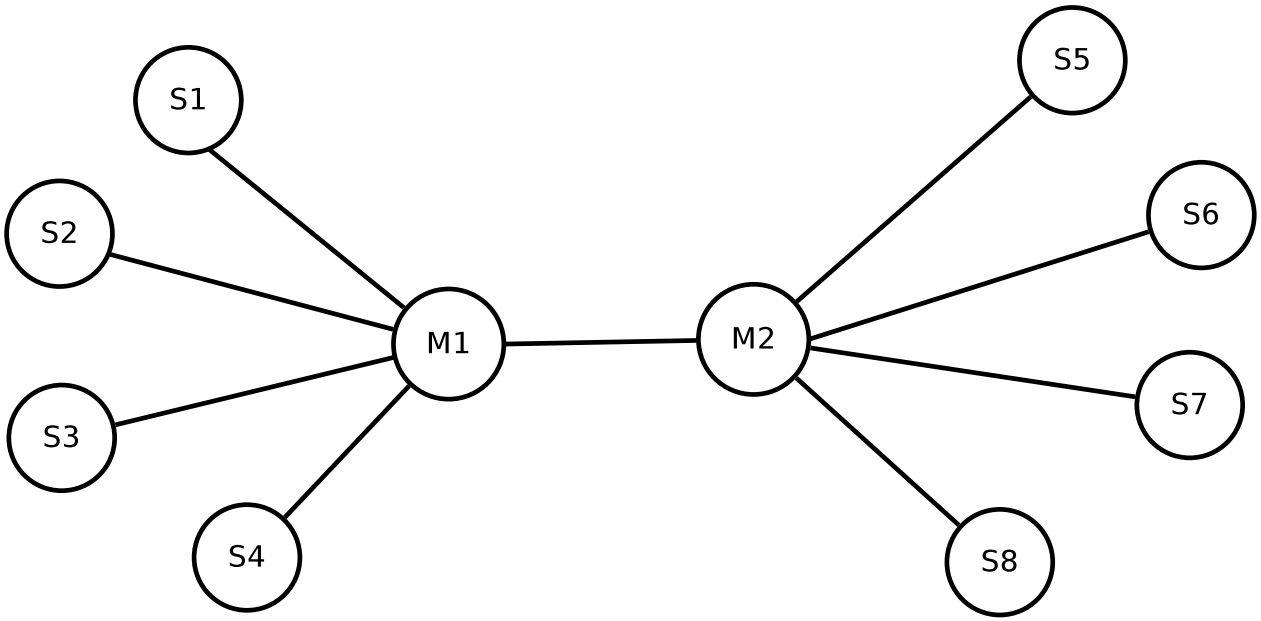
- Split the training data and then mix



- No lose of accuracy

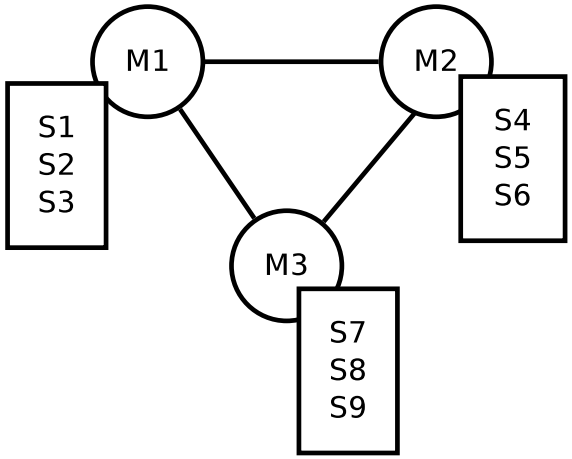
Parallelization - Strategy 1: Vertex for every sample

- Every sample is a vertex
- Master vertex for aggregation
- $|S| + p(p-1)$ message exchanges
- Signification communication overhead



Parallelization - Strategy 2: Vertex for set of samples

- Vertex for each partition
- The vertex has set of samples
- $p(p-1)$ message exchanges
- Low communication overhead



Experimental results - Data sets

- Movie Reviews[Pang and Lee, 2004]
  - 2000 movie reviews: 1000 positive and 1000 negative
  - 100 features

Algorithm	Accuracy
Maximum Entropy	86.33
Naive Bayes	85.62

Experimental results - Data sets

- 20 Newsgroups [1]
  - 25000 document
  - 18846 training examples, and 7532 test examples
  - 20 categories
  - 5000 features

Algorithm	Accuracy
Maximum Entropy	57.44
Naive Bayes	92.12

[1] people.csail.mit.edu/jrennie/20Newsgroups

Experimental results - Data sets

- Wikipedia INEX 2009 collection [Schenkel et al., 2007]
  - Prepared for evaluating information retrieval tasks
  - 2,666,190 articles from Wikipedia; we use a subsets of 40,000 articles
  - 331 categories taken from Dbpedia[2]
  - 10000 features

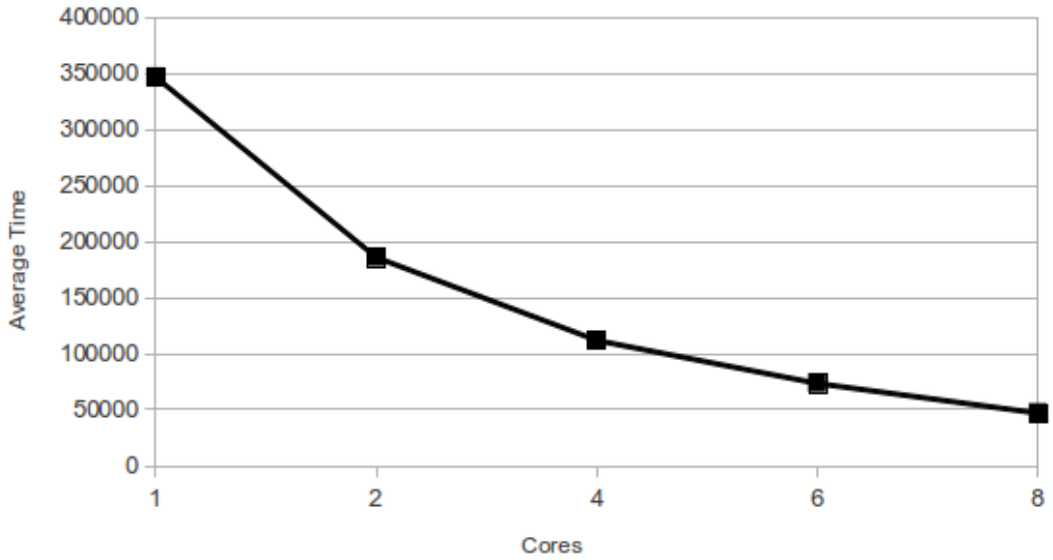
[2] dbpedia.org/About

Experimental results - Benchmark Methodology

- 8 cores machine
- 40 000 articles from Wikipedia
- 5 runs for every core, ignore the first one

Experimental results - Maximum Entropy Benchmarks

- Vertex for a set of sample
  - 1.65x times faster for every 2 cores on average
  - 1.86x times between 1 and 2 cores
  - 7.38x times between 1 and 8 cores



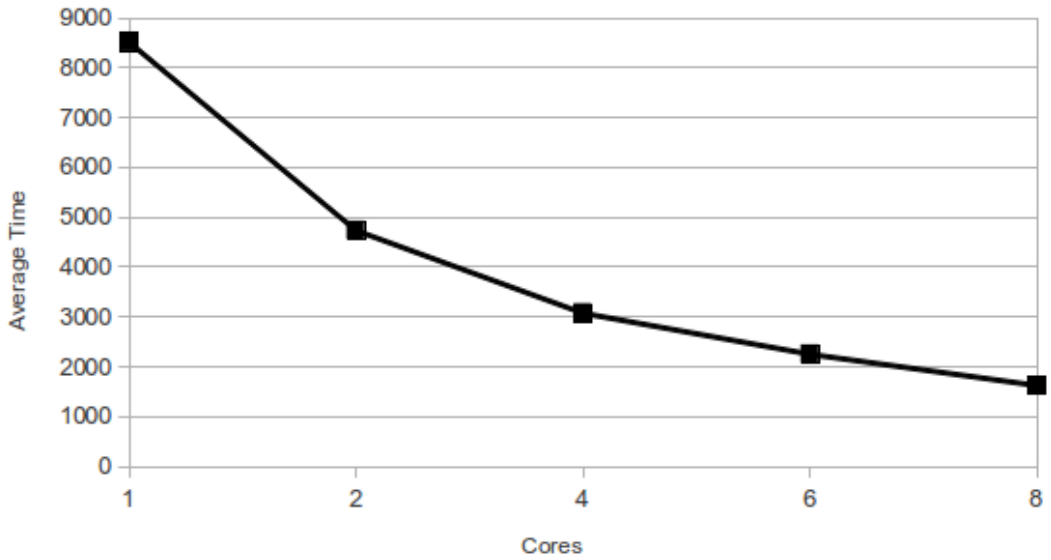
Experimental results - Maximum Entropy Benchmarks

- Vertex for every sample
  - 5241952 ms on 8 cores
  - 111x times slower
- Sequential
  - No improvement between 1 and 8 cores
  - Parallel version is 1.40x times faster

Cores	Avarage Time
1	67235
8	67140

Experimental results - Naive Bayes Benchmarks

- Vertex for a set of sample
  - 1.52x times faster for every 2 cores on average
  - 1.79x times between 1 and 2 cores
  - 5.25x times between 1 and 8 cores

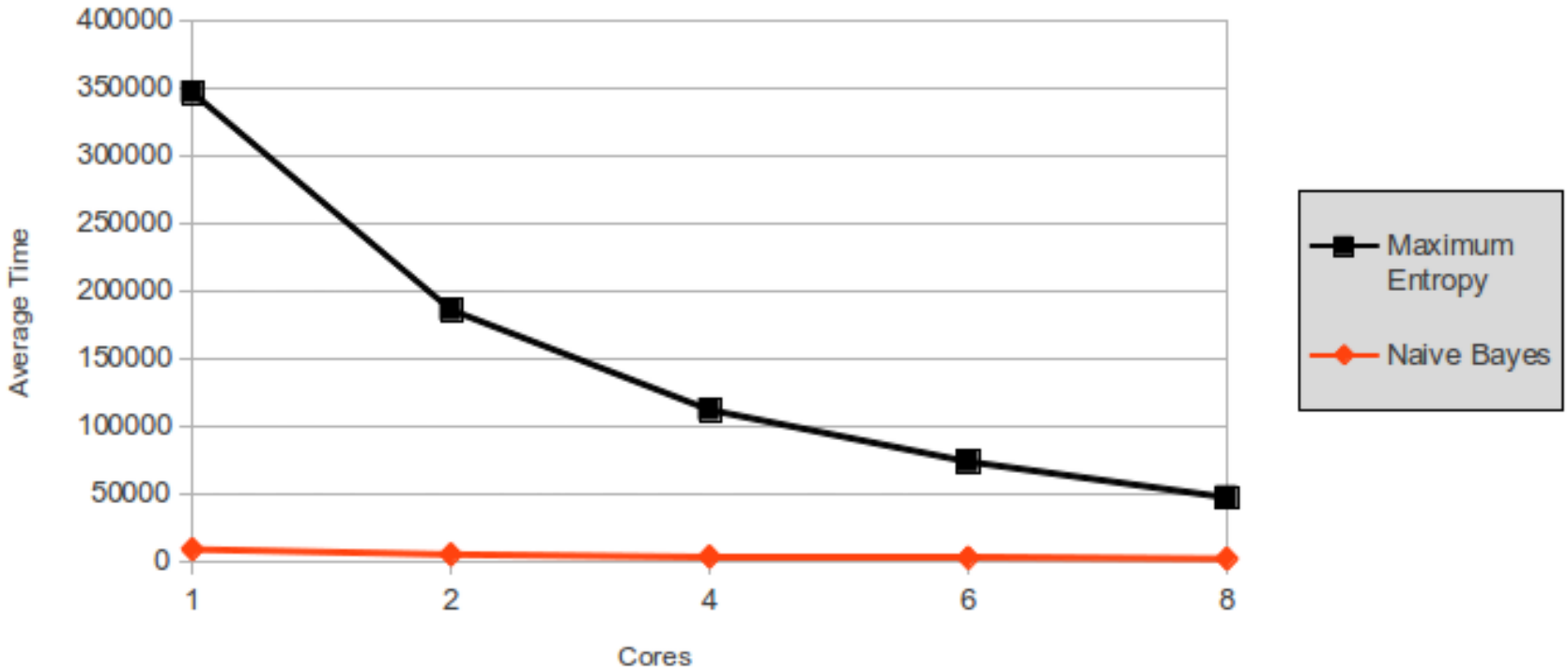


Experimental results - Naive Bayes Benchmarks

- Vertex for every sample
  - 62300.5 ms on 8 cores
  - 38x times slower
- Sequential
  - No improvement between 1 and 8 cores
  - Parallel version is 3.10x times faster

Cores	Average Time
1	5224.75
8	5579.75

Experimental results - Comparison



Conclusion

Menthor makes it easy to implement parallel machine learning algorithms without much effort and with excellent scalability.

The end

Questions ?

References I

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

Philipp Haller and Heather Miller. Parallelizing machine learning-functionally: A framework and abstractions for parallel graph processing, 2011. URL `http://infoscience.epfl.ch/record/165111`.

Gideon Mann, Ryan T. McDonald, Mehryar Mohri, Nathan Silberman, and Dan Walker. Efficient large-scale distributed training of conditional maximum entropy models. In Yoshua Bengio, Dale Schuurmans, John D. Lafferty, Christopher K. I. Williams, and Aron Culotta, editors, *NIPS*, pages 1231–1239. Curran Associates, Inc, 2009. ISBN 9781615679119. URL `http://books.nips.cc/papers/files/nips22/NIPS2009_0345.pdf`.

References II

Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the ACL*, 2004.

Jason Rennie, Lawrence Shih, Jaime Teevan, and David Karger. Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of ICML-03, 20th International Conference on Machine Learning*, Washington, DC, 2003. Morgan Kaufmann Publishers, San Francisco, US. URL `http://www.ai.mit.edu/~jrennie/papers/icml03-nb.pdf`.

Ralf Schenkel, Fabian M. Suchanek, and Gjergji Kasneci. YAWN: A semantically annotated wikipedia XML corpus. In Alfons Kemper, Harald Schöning, Thomas Rose, Matthias Jarke, Thomas Seidl, Christoph Quix, and Christoph Brochhaus, editors, *BTW*, volume 103 of *LNI*, pages 277–291. GI, 2007. ISBN 978-3-88579-197-3.

References III

URL http://subs.emis.de/LNI/Proceedings/Proceedings103/article1404.html.