Efficient Probabilistic Latent Semantic Analysis Through Parallelization

Raymond Wan[1,3] r.wan@aist.go.jp
Vo Ngoc Anh[2] vo@csse.unimelb.edu.au
Hiroshi Mamitsuka[1] mami@kuicr.kyoto-u.ac.jp

[1] Bioinformatics Center, Institute for Chemical Research, Kyoto University, Gokasho, Uji, 611-0011, Japan
[2] Department of Computer Science and Software Engineering, Faculty of Engineering, University of Melbourne, Victoria, 3010, Australia
[3] Computational Biology Research Center, AIST, 2-42, Aomi, Koto-ku, Tokyo, 135-0064, Japan

Abstract

Probabilistic latent semantic analysis (PLSA) is an effective technique for IR, but has one drawback: its dramatic consumption of both execution time and memory. In this work, we improve the efficiency of PLSA without changing its output by:

1) *re-evaluating the data structures used,*
2) *modifying how the EM algorithm is implemented, and*
3) *incorporating both shared and distributed memory parallelization.*

We evaluate our system on several text collections commonly used in the literature.

Background of PLSA

- Associates two types of data through a set of Z latent (hidden) states.
- In IR, these two data types could be the set of words W and documents D.
- Input: a co-occurrence matrix of size $|W| \times |D|$.
- Output: joint probability across Z latent states:

$$p(w,d) = \sum_{z \in Z} p(d|z)p(w|z)p(z)\,. \quad (1)$$

- Parameters of the above obtained from the Expectation-Maximization (EM) algorithm which iterates between the **E-step** and **M-step**.

Methods Considered

- Only keep the non-zero co-occurrence counts.
- Combine the E-step and M-steps (see below).
- Incorporate either or both shared (OpenMP) and distributed (MPI) memory parallelization (see right panel).

Combining the E-step and M-step removes the largest data structure ($p(z|w,d)$):

E-step
$$p(z|w,d) \propto p(d|z)p(w|z)p(z)$$
M-step
$$p(w|z) \propto \sum_{d \in D} n(w,d)p(z|w,d)$$
$$p(d|z) \propto \sum_{w \in W} n(w,d)p(z|w,d)$$
$$p(z) \propto \sum_{d \in D}\sum_{w \in W} n(w,d)p(z|w,d)$$

E-step + M-step
$$p^{(new)}(w|z) \propto \sum_{d \in D} n(w,d)\frac{p(d|z)p(w|z)p(z)}{p(w,d)}$$
$$p^{(new)}(d|z) \propto \sum_{w \in W} n(w,d)\frac{p(d|z)p(w|z)p(z)}{p(w,d)}$$
$$p^{(new)}(z) \propto \sum_{d \in D}\sum_{w \in W} n(w,d)\frac{p(d|z)p(w|z)p(z)}{p(w,d)}$$
where $p(w,d) = \sum_{z \in Z} p(d|z)p(w|z)p(z)$

We performed experiments with the following combinations and data sets:

	Baseline	None	OpenMP	MPI	All
Non-zero counts	Yes	Yes	Yes	Yes	Yes
Combine EM	No	Yes	Yes	Yes	Yes
Open MP	No	No	Yes	No	Yes (4)
MPI	No	No	No	Yes	Yes

	$\|W\|$	$\|D\|$
CRAN	7,479	1,398
CISI	9,814	1,460
MED	10,673	1,033
CACM	12,195	3,204

Parallelization

We distribute the computation of the EM algorithm across multiple CPUs using two paradigms: shared or distributed memory.

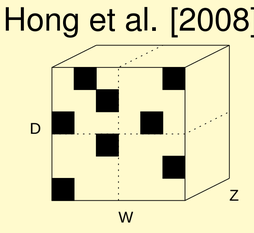
- shared memory – uses the cores of a single computer
- distributed memory – uses one or more cores on multiple computers within a network

	Shared	Distributed
Machines	Single CPU	Multiple CPU
Network	N/A	Possible
Granularity	Fine (loop-level)	Coarse (function-level)
Data structures	Shared	Explicitly transmitted
Disk access	Competing	Separate
Standard	OpenMP	MPI
API	OpenMP	Open MPI

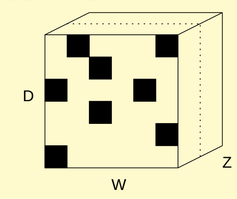
Related Work

Hong et al. [2008] only considered OpenMP and also:

Hong et al. [2008]:



Our work:

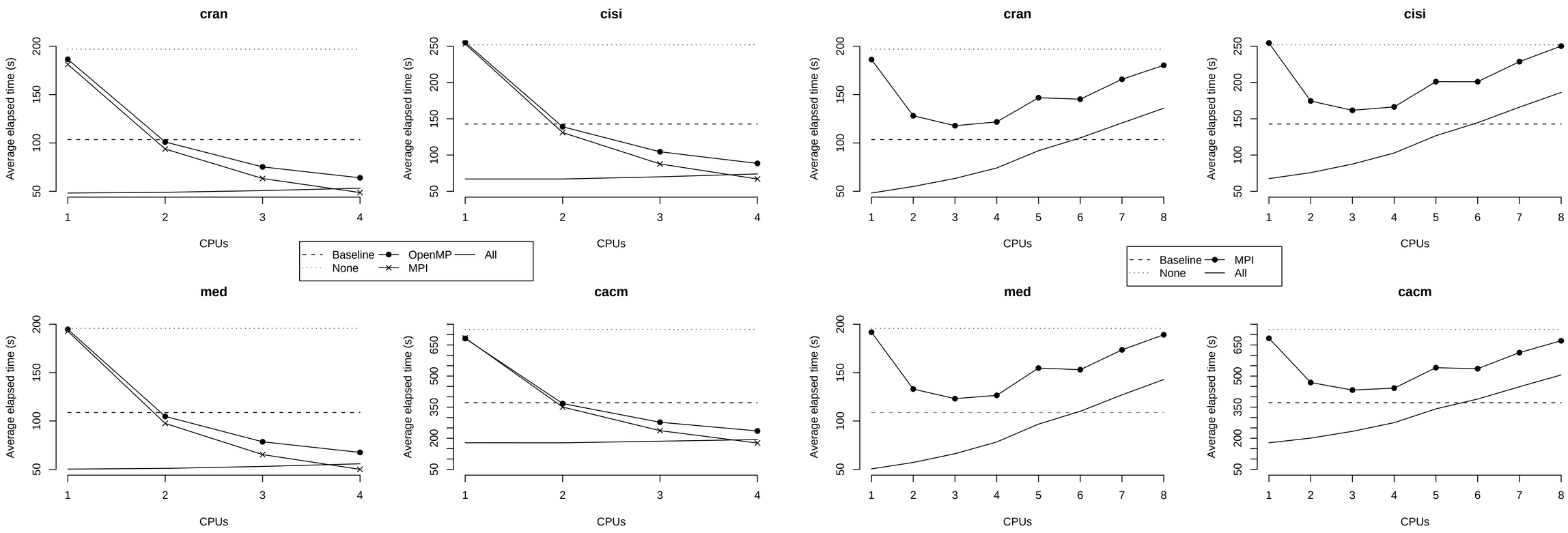


Future Work

Our next step is to extend our evaluation to larger data sets (ideally, TREC collections) and to combine our work with other existing techniques that will change the output of PLSA (e.g., vocabulary reduction).

Source code: `http://www.cbrc.jp/~rwan/`

Results: •Combining E-step and M-step reduces memory usage but increases running time.
•Parallelization decreases running time, with more predictable behavior for executations on a single computer (perhaps due to the network overhead).



(a) Single computer

(b) Network of 8 computers

The 5th Asia Information Retrieval Symposium (**AIRS 2009**), October 21-23, 2009, Sapporo, Hokkaido, Japan