Self-Adaptive Classifier for Efficient Text-stream Processing

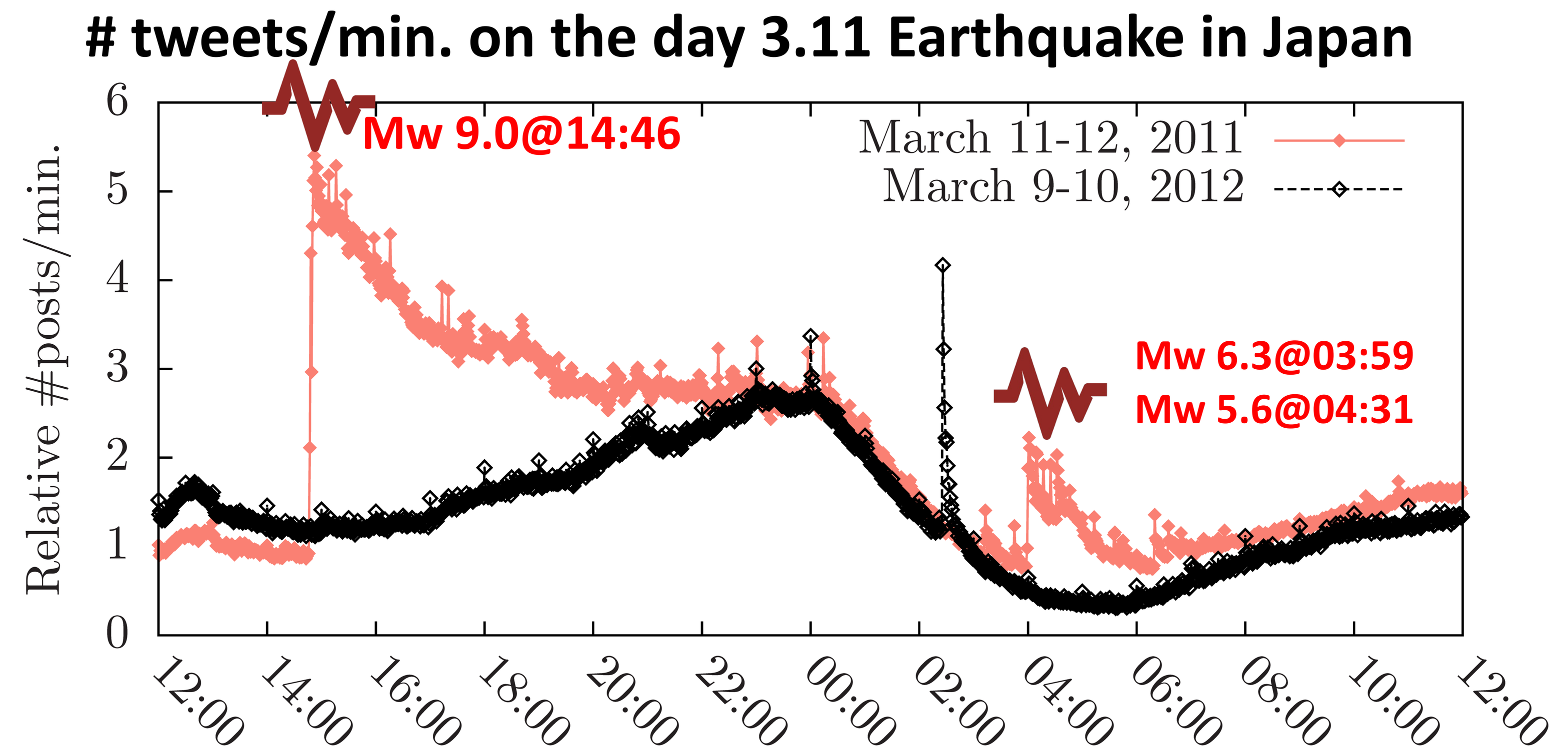
Naoki Yoshinaga (Univ. Tokyo) and Masaru Kitsuregawa (NII; Univ. Tokyo)

Introduction

A social text stream (e.g., twitter) mirrors the state of real world, so **analyzing a real-time text stream is beneficial** for reducing natural disasters, monitoring sentiment, predicting stock market etc.

Challenge: **The content and volume of flow changes dramatically in a text stream**, reflecting a change in the real world

 # tweets/min. on the day 3.11 Earthquake in Japan



Proposal

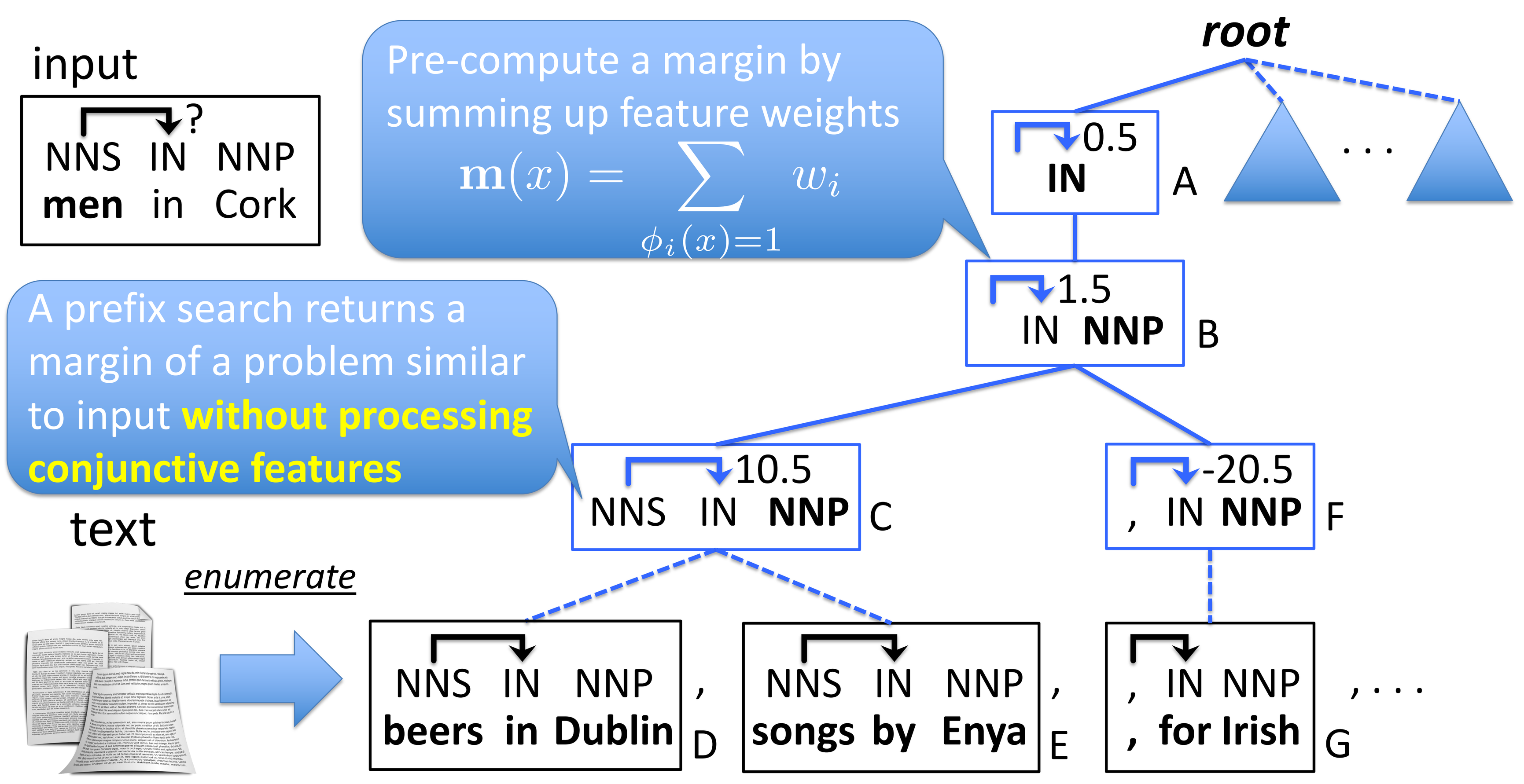
We **dynamize** a linear classifier based on feature sequence trie **[Yoshinaga & Kitsuregawa '09]** so that **it adaptively speeds up classification while processing a text stream**

Classification based on feature sequence trie [Yoshinaga & Kitsuregawa, EMNLP '09]

Use of **conjunctive features** (e.g., n-grams) improves accuracy but slows down processing time in NLP tasks

Then, solve **common classification problems** in advance to **quickly solve new problems as their instances**

- Use **global statistics** to select common problems
- Store problems in a feature sequence trie for fast retrieval

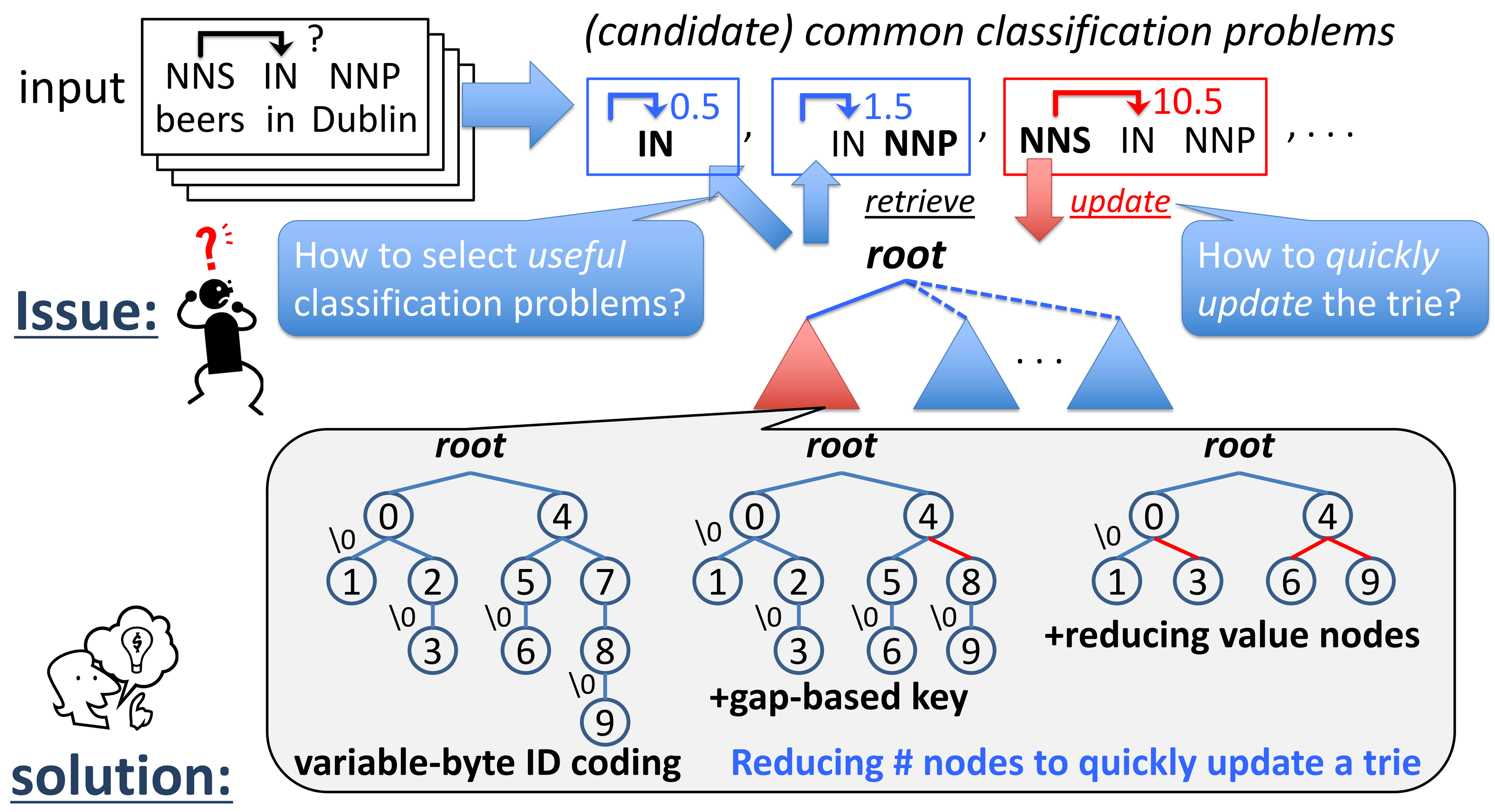


Problem: it cannot effectively speed up when a burst occurs and the topic (content) shifts in a text stream

Self-adaptive classification for text stream [this paper]

Idea: Keep updating a set of common classification problems while processing text:

1. build/enumerate common classification problems by **adding frequent features** in input one by one
2. get a margin if exists, o/w compute/store a margin



(candidate) common classification problems

A B C D A B C E A B F G . . .

Use cache algorithms to keep only k problems

when k=3, **LRU (Least Recently Used)** keeps E, C, B, while **LFU (Least Frequently Used)** keeps A, B, C

Stop enumeration as soon as the label (sign of margin) is fixed

use upper-/lower-bounds after adding the rest feature weights

Experiments

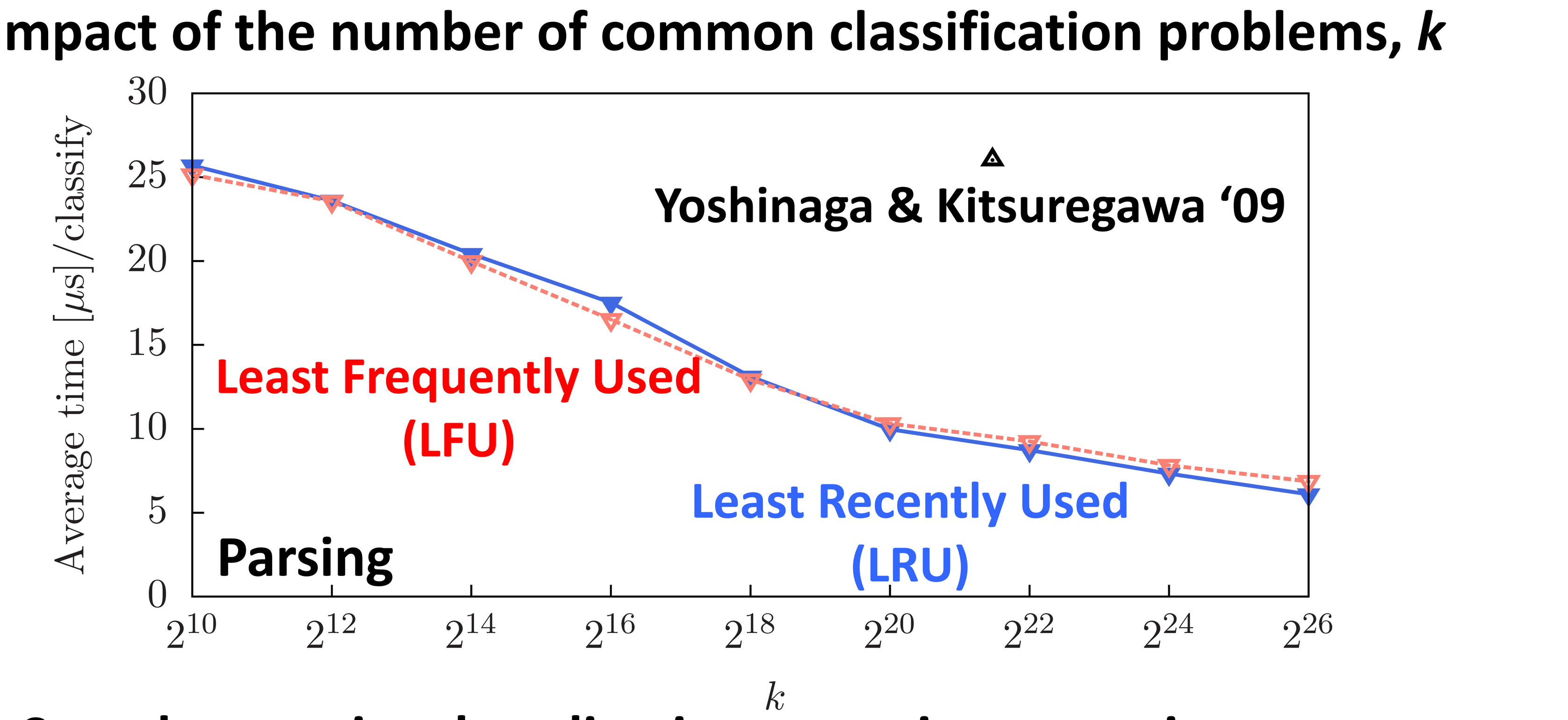
- **Data**: Tweet stream on 3.11 Earthquake (9M posts in Japanese)
- **Tasks:** base-phrase chunking / dependency parsing
- **Models**: pointwise chunker / shift-reduce parser [Sassano '04]
- **Base classifier**: PA-I with 3rd-order poly kernel

Overall classification performance for tweets on 3.11 Earthquake

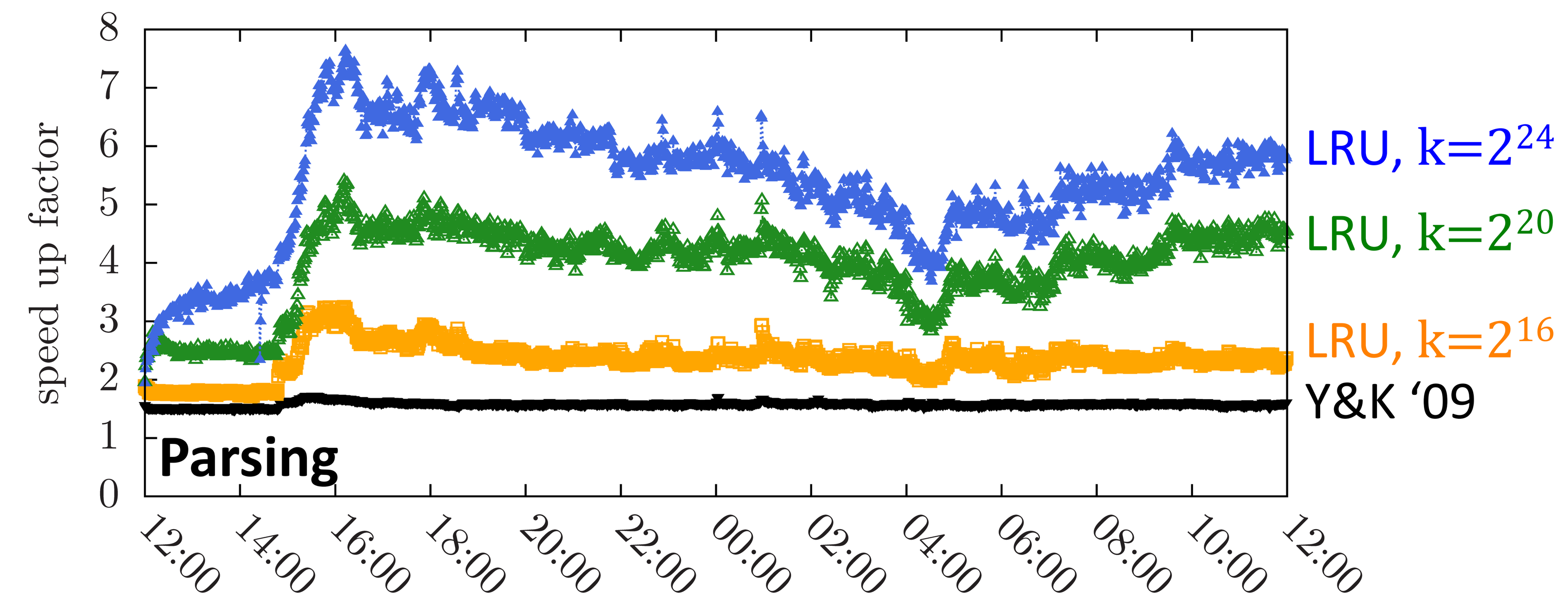
Method	Chunking		Parsing	
	Speed [ms/sent.]	Space [MiB]	Speed [ms/sent.]	Space [MiB]
Baseline [Kudo & Matsumoto '03]	0.0221	12.0	0.1187	31.5
[Yoshinaga & Kitsuregawa '09]	0.0118	30.5	0.0738	99.9
This paper (LFU, $k=2^{20}$)	0.0088	90.7	0.0293	113.4
(LFU, $k=2^{24}$)	0.0081	463.0	0.0222	904.3
(LRU, $k=2^{20}$)	0.0077	85.9	0.0283	108.9
(LRU, $k=2^{24}$)	**0.0070**	399.2	**0.0208**	840.9

Environment: Intel Core i7-3720QM 2.6GHz CPU server with 16GB RAM

Impact of the number of common classification problems, k



Speed-up against baseline in processing one-min. tweets



All the codes are available as open-source softwares at http://www.tkl.iis.u-tokyo.ac.jp/~ynaga/