MTMonkey: A Scalable Infrastructure for a Machine Translation Web Service

Aleš Tamchyna, Ondřej Dušek, Rudolf Rosa, Pavel Pecina {tamchyna,odusek,rosa,pecina}@ufal.mff.cuni.cz
Institute of Formal and Applied Linguistics, Faculty of Mathematics and Physics, Charles University in Prague

- **MT web service** for cross-lingual information retrieval in the medical domain
- Real-time requests through a RESTful **JSON API**
- Variety of clients, several language pairs
- Written in **Python**

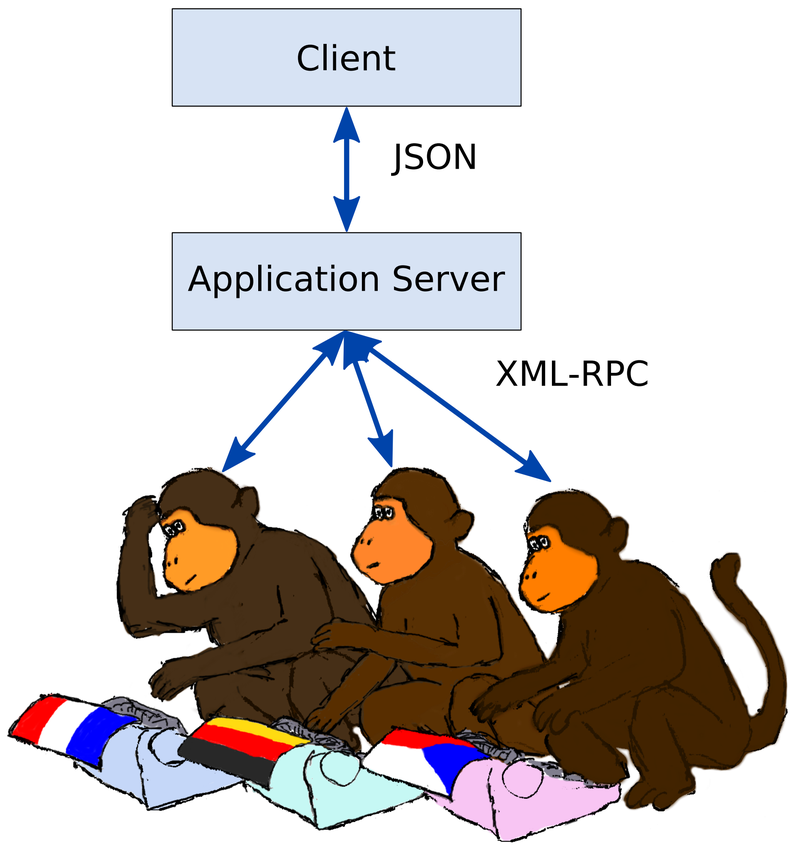
https://github.com/ufal/mtmonkey

The Khresmoi Project

Automated information extraction from biomedical documents

- Semantic search adapted to user requirements
- Automated analysis and indexing of medical images in 2D (X-Rays), 3D (MRI, CT), and 4D (MRI with a time component)
- Linking information extracted from biomedical texts and images to structured information in knowledge bases
- Support of **cross-language search**, including multilingual queries, and returning **machine-translated pertinent excerpts**
- Adaptive user interfaces to assist in formulating queries and interacting with results

Overall Architecture



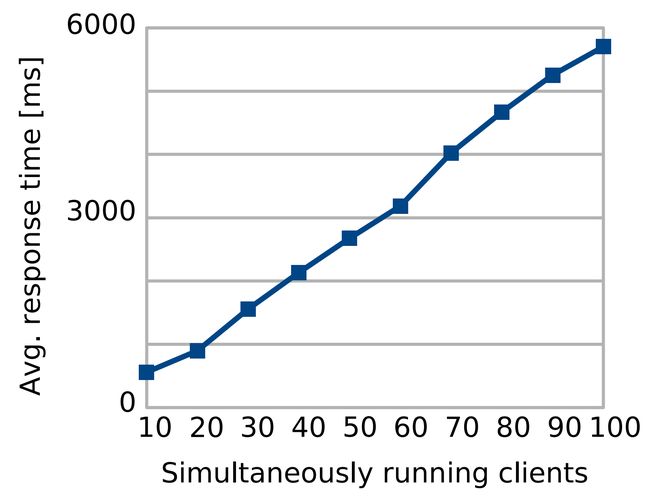
Workers: MT Monkeys

Load Balancing

- There can be more workers per language pair
- Load balancing via simple **round robin**

Load Testing

- Testing on sentences from the medical domain
- Each client sends 10 requests and reports the average response time
- Each test is repeated 10 times and averaged
- 4 worker instances per translation direction

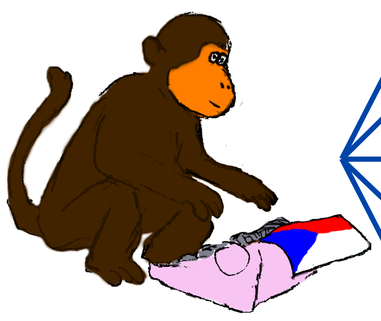


- Response time grows linearly with the number of simultaneous requests
- For small numbers (1-10 clients), the response time is constant, around 0.5 s

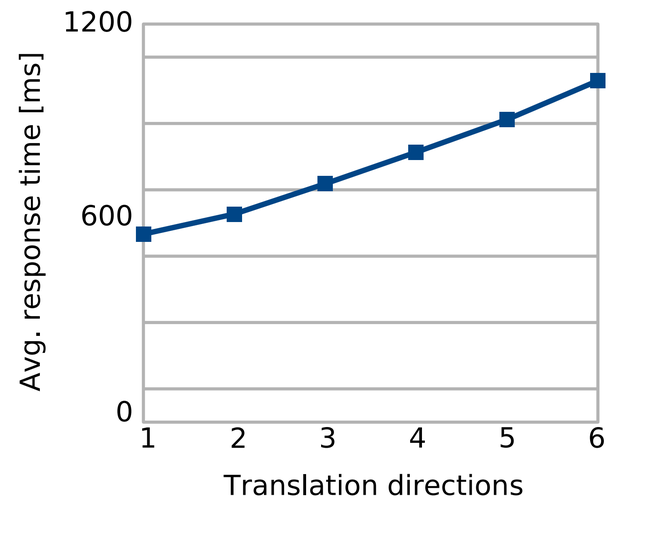
Fault Recovery

- **Scheduled self-tests** (using *cron* on the worker machines) with automatic restart on error
- Scheduled external testing with e-mail notification on error
- **Automatic updates** of workers, Moses code + Moses models

Workers



- Sentence splitter (en) — Python re-implemenation of Moses (EMS) split-sentences.pl
- Tokenizer with lowercasing — Python script/module
- Translator (en→cs) — Moses server
- Recaser (cs) — Moses server
- Detokenizer (cs) — Python re-implementation of Moses detokenize.perl



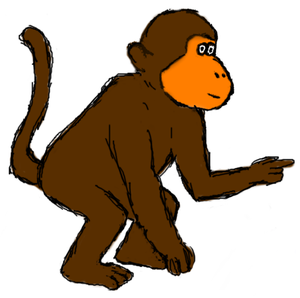
- Adding more translation directions does not degrade performance much

Moses Systems

- Moses for **translation** and **recasing**
- Independent instances of Moses server
- Communication via XML-RPC
- Binary phrase-tables
- Lazy-loading binary Ken LMs
- Fast start-up, low memory consumption (OS handles caching)

A Robust Tokenizer

- MTMonkey handles requests from various sources
- Input tokenization is not always identical
  ⇒ Need a universal tokenizer
- Aggressive: **splits on any punctuation**
- Language-independent
- Could decrease MT quality but reduces data sparsity
- (Almost) any input tokenization will be split identically



Presented at MT Marathon, September 9–14, Prague, Czech Republic.

http://khresmoi.eu

Supported by the EU FP7 grant n°257528 (KHRESMOI) and Czech Ministry of Culture grant DF12P01OVV022 (NAKI/Amalach). Using resources of the Czech Ministry of Education project LM2010013 (LINDAT-Clarin).