MT Marathon 2015 (Prague)

Proposed Projects

Start by **sharing the document with yourself** (so that Google records edits well).
Then simply start a new entry or add your comments anywhere, in the text or on side.
Projects can be proposed until the first day of MT Marathon, but announcing them earlier might attract more participants, come better prepared etc.

Overview

- The following links are mostly to the MTM15 wiki, use the username and password ‘**mtm**’ for read-only access.
- All project boaster slides
- All project mid-week slides
- All project final slides

Thanks for your participation in the projects!
The project svn will remain open for years, so you may want to browse it or upload any updates at any time.

svn co https://svn.ms.mff.cuni.cz/svn/mtmarathon-2015/trunk/projects/

If you don’t have an account at our svn server yet, your username is name.surname and your password is the (lowercase) string between ‘@’ and the first dot in your e-mail address, e.g. ‘gmail’.
https://svn.ms.mff.cuni.cz/trac/mtmarathon-2015/browser/trunk/projects/boaster-slidesan
e-mail to Ondřej Bojar (bojar@ufal.mff.cuni.cz) with the output of the command
`htpasswd -sn your.name.surname` # pick a password
for a password reset.

List of Projects

Sample Project

(John the Proposer; include e-mail if you want to)

- The bright idea.
- The goal.
- Prospective participants: John the Proposer

Commonspector

(Ondřej Bojar, Aleš Tamchyna)

Please e-mail Ondrej and Ales {bojar,tamchyna}@ufal.mff.cuni.cz to meet and get started.

- Prospector is an extension of Eman, our variant of EMS, that automatically searches through MT experiment setups.
- The goal of the project is to use Prospector for community service:
    - Think of Large Hadron Collider where you submit your experiments and (a thousand years later), the results will appear for you on a web page.
    - Or think of a cruise-control extension that regularly checks the actual performance of Moses releases on a few language pairs.
- Prospective participants: Ondřej, Aleš

Appraise++

(Christian Federmann)

Please e-mail Christian <chrife@microsoft.com> and we’ll go from there…

Adding more evaluation tasks/data views/support for WMT16 to Appraise while porting the codebase over to Django 1.8; will require Python, Git, and, preferably, Django knowledge. I’ll be happy to help people merge their code into the Appraise repository.

- Prospective participants:
    - Christian
    - Maja Popović, remotely

MT-ComparEval

Please e-mail Martin Popel <popel@ufal.mff.cuni.cz>.

- Improve the tool (https://github.com/choko/MT-ComparEval), adapt it to your needs.

    - see https://github.com/choko/MT-ComparEval/issues/ (add new issues)
    - support external metrics (computed outside the tool), e.g. METEOR
    - support for Hjerson and highlighting based on human error annotation
- Improve the public MT-ComparEval server with WMT translations http://wmt.ufal.cz
- Integrate this into http://matrix.statmt.org/ or Appraise++ (the project above).

Requirements: nothing (yes, we need also people for testing and discussing the design etc. For improving the code, PHP/JavaScript skills are welcome.)
See also: http://ufal.mff.cuni.cz/pbml/104/art-klejch-et-al.pdf
Participants: Martin Popel, Ondřej Klejch (author of MT-ComparEval), Roman Sudarikov

Define a common interface between Moses EMS and tuners

Please e-mail Matt Post <post@cs.jhu.edu>
Moses' Experiment Management System is really nice, and even extensible through custom edits to the experiment.meta file (there is, for example, some latent support for David Chiang's python-based Hiero system). However, it's not perfectly general, but in fact has hard-coded Moses assumptions in several places, particularly when it comes to running Moses tuners (e.g., the Hiero system hard-codes a separate path around the Moses tuners to run its own). It would be really useful if there were a common API to Moses tuners so that swapping out compliant decoders was as simple as changing the "decoder" variable at the top of an EMS config file. This project will (a) determine what exactly would need to change (b) write out a spec that requires minimal modifications to Moses itself (c) implement this for Joshua, and any other decoder, and (d) convince the powers-that-be to accept our commits.

Requirements: Perl hacking skillz

Large-scale discriminative tuning on the training data

Please e-mail Matt Post <post@cs.jhu.edu>
Large feature sets are helpful to MT, but it is hard to reliably tune them on small tuning sets (especially lexicalized ones). Liang Huang has had some nice work tuning large features sets on jl) and (b) enabling introspection of the chart following decoding (to find and update against the worst violation).

Requirements: Python abilities (for the script and other experiment infrastructure), a bit of machine learning knowledge (though you can learn perceptrons on the spot). Either familiarity with Java/C++ and/or the inner workings of a phrase-based decoder will be necessary for part (b).

Prefetching and Pipelining/Batching for Language Models

Please e-mail Kenneth Heafield <mtm@kheafield.com>
Profiling language model queries shows that, by a factor of 10, the most expensive instruction is the load from a random memory address. Rather than execute these loads serially, we can

execute them in parallel from within one thread by prefetching. Specifically, the code calls __builtin_prefetch in gcc, does other work, and later returns to do the actual load from cache. That other work can be generating more language model queries, so we have multiple queries in flight going through a pipeline. This hardware memory fetching parallelism all happens within the same thread; multiple threads operate on separate sentences as usual.

For memory prefetching, the optimal pipeline depth is about 10 so there should be minimal degradation in search accuracy. The goal is to speed up decoding enough that one could simply raise the pop limit to attain better accuracy in less time.

Batching language model queries has been done for networked storage by Google (http://www.aclweb.org/anthology/D07-1090.pdf) and by Oliver Wilson in RandLM. However, there were separate code paths for local and networked models, which made it hard to maintain. Hieu deleted the networked code path in Moses. Since this project seeks to improve performance for local models as well, there will be only one pipelined code path.

Neural models can batch queries using the same high-level API (query submission and completion). This allows them to use matrix-matrix operations rather than a bunch of individual matrix-vector operations. Single-layer models that use premultiplication and approximate normalization are unlikely to benefit, but other models might.

The project can be divided into a few parts:

1. API for pipelined/batched queries.
2. Make perplexity computation as fast as possible using prefetching.
3. Decoding algorithm integration.

Requirements: C++. Parts 1 and 2 can be done without knowing any decoding algorithms. Part 3 requires understanding cube pruning/incremental decoding. No assembly knowledge required; this is all done at the C++ level with an intrinsic wrapper around the prefetch instruction.

Integrate MT-evaluation tools in translate5 / enhance translate5 MT-evaluation capabilities

Please e-mail Marc Mittag <marc ÄTT mittagqi DOT com>

translate5 is a web-based proofreading, postediting & translation evaluation environment.

All WMT-data since 2006 is stored at http://translate5-metashare.dfki.de/ in a translate5 instance
and offered researchers as a lookup base. In addition, translate5 is used to compare and annotate MT-data in a column-based approach.
On the other hand several Language Service Providers proofread translation data in translate5

in daily business since 2012.

By integrating evaluation tools with translate5, it can serve as a common platform for MT-research to store, evaluate and compare data. In addition translate5 itself can be extended at some points to better serve evaluation purposes.

Depending on the interests of participants, one or more of the following issues could be addressed in this project:
- integrating MT-ComparEval with translate5 via API
- integrate tools for calculating Meteor, BLEU and F-measure and store and visualize their results
- visualize logged time in translate5 interface (logging of editing time exists)
- log key strokes in translate5 and visualize them in the interface
- Generate MQM score (MQM-annotations and statistical overview exists in translate5)

For more information and community spaces see
http://www.translate5.net/

Requirements: Javascript and/or php skills

VW-based feature functions for document-level context

Please email Pierre Lison <plison@ifi.uio.no>

Vowpal Wabbit is a highly efficient, out-of-core learning algorithm supporting a variety of losses functions and regularizers (see https://github.com/JohnLangford/vowpal_wabbit for details). Discriminative models based on Vowpal Wabbit have recently developed to perform phrase-sense disambiguation for SMT based on contextual features (see https://ufal.mff.cuni.cz/pbml/101/art-tamchyna-et-al.pdf).

Several feature functions based on Vowpal Wabbit are already integrated into Moses, but all functions developed so far have concentrated on sentence-internal features, such as bag-of-words or bigrams from the source or target. It would be interesting to investigate whether one can can follow a similar approach to exploit features extracted from the *document context* (that is, from outside the current sentence). This would allow us to incorporate a rich set of interesting linguistic features to the SMT pipeline.

CloudLM: a Cloud-based Language Model for Machine Translation

(Jorge Ferrández, remotely Antonio Toral and Sergio Ortiz-Rojas)
Please e-mail Jorge Ferrández <jferrandez@prompsit.com>

- CloudLM is an open-source cloud-based LM intended for SMT, which allows to use distributed LMs. CloudLM relies on Apache Solr and provides the functionality of state-of-the-art language modelling (it builds upon KenLM), while allowing to build massive LMs (as the use of local memory is drastically reduced), at the expense of slower decoding speed.

- The goal of the project is to develop some improvements and experiments for CloudLM:
    - Enhance efficiency improving connections between Moses and Solr.
    - Try to make the system asynchronous.
    - Evaluation of CloudLM for huge LMs.
- CloudLM is a fork of Moses in GitHub:https://github.com/jferrandez/mosesdecoder/tree/cache-cloudlm
- Requirements: C++, Moses, Boost C++ libraries

Open-Source Unsupervised Morphology

Tom Kocmi (kocmitom@gmail.com), Ondřej Bojar (bojar@ufal.mff.cuni.cz)

- The goal of this project is to create open-source implementation of unsupervised morphology as described in the article http://www.aclweb.org/anthology/N15-1186. After implementation, we will experiment with languages mentioned in the article and additionally with Czech.
- Preferred programming language is Java, where we could use http://deeplearning4j.org/word2vec.html for SkipGram implementation, but it is possible to change the implementation language to Python in case of bigger interest.
- Advisor: Keith Stevens
- Prospective participants: Kocmi, Ondřej, Joachim, Federico

Better Unsupervised Compound Splitting

(Joachim Daiber, daiber.joachim@gmail.com)

- The goal of this project is to create and package an efficient implementation of compound splitting based on an unsupervised method working on word embeddings
- The method is similar to the method in the article http://www.aclweb.org/anthology/N15-1186, but applied to non-inflectional morphology
- Paper: http://jodaiber.github.io/doc/compound_analogy.pdf
- Prospective participants: Joachim, Kocmi

Rust programming language in MT

(Pēteris Ņikiforovs, mtm@peteris.rocks)

Rust is a modern systems programming language from Mozilla. It's got zero-cost abstractions, guaranteed memory safety, threads without data races, generics, optional GC, native Unicode/UTF-8 support. Basically, it's fast as C++ with nicer syntax and no segfaults. I think it's perfect for MT.

We could rewrite some of the moses perl scripts (truecaser, detokenizer, etc.) in Rust to learn the language and improve their performance.

Prospective participants: Pēteris, Shing (remotely, shing.zhan@gmail.com)

What to do:

1) Read the first three chapters of the Rust book (1. Introduction, 2. Getting Started, 3. Learn Rust (3.1. and 3.2.))
2) Port prepare-fast-align.perl to Rust (joins lines of two input files with |||)

Interactive Experiment Management System

(Pēteris Ņikiforovs, mtm@peteris.rocks)

The idea is to create an interactive GUI using web technologies for managing the workflow of machine translation experiments. Think of the Moses EMS graph but interactive with drag and drop. It would be like RapidMiner for MT. I've never used it but I guess it would be similar to LooneyBin.

The tool would be a single page web application written in JavaScript/ES6 (I'd prefer using React.js) which would generate a Makefile/bash script/qsub commands which could then be used to run the experiment. It would also be possible to see which steps are running, preview the input/output files right in the browser, etc.

Requirements: front-end dev skills (JavaScript, React.js, SVG)

Prospective participants: Pēteris, Matīss

Improve zoning in Moses

(Pēteris Ņikiforovs, mtm@peteris.rocks)

You can use <zone> tags in Moses to limit reordering. But it is not perfect:

<zone> **Prends le** </zone> bloc bleu .
Paņem klucīti **no** zilā .

If you know the insides of moses, you could help improve this.

MT in NLTK

(Proposer: Liling Tan, alvations@gmail.com)

NLTK has been a great help in educating the research community about NLP and also provide basic and useable NLP functionalities. Improving the code by adding functions and wrapper scripts to existing libraries would be a great educational resource to lower the steep learning curve in MT as well as feeling a sense of *philotimos* where we give back to the open source world. Get a chance to enter the world of Python and NLTK ***AWESOMENESS*** with MT!!!

Ideas to improve MT in NLTK

Basic level coding:

- create new corpus reader for non-breaking prefixes into an NLTK wordlist corpus
- porting moses tokenization regexes into python and packaging it as a new NLTK tokenizer

Intermediate level coding:

- Building a truecase model trainer and truecaser in NLTK
- Writing a pipeline code from IBM models in NLTK to GDFA and phrase extraction
- Writing a code to assign probabilities and interpolate/smooth the extracted phrases

Advance level coding:

- Write a proper decoder for MT in NLTK (References point(s): https://github.com/sfu-natlang/Kriya)
- Improve the ngram language model in NLTK (Reference point(s): https://github.com/AdolfVonKleist/SimpleLM)
- Write a Neural MT model with python/Theanos and push into the NLTK repository

Getting Started: https://goo.gl/5HabYa

Prospective Participants:

- Liling Tan
- Shing (remotely, shing.zhan@gmail.com)
- Steven Bird (code review only)

Multipass Decoding

(Hieu Hoang; hieuhoang@gmail.com)

● Integrating stateful feature functions can be done in 2 ways: 1. during the construction of the output lattice ('single pass'), or 2. rescoring the output lattice ('multipass'). Moses and Joshua implement (1), cdec implements (2). An implementation of multipass decoding has been developed for Moses. This project would iron out any bug in the implementation and come up with some uses for multipass decoding.
● Multipass decoding would be useful for integrating some advance features such as neural network language models, target language desegmentation, course-to-fine features. If you are want to work with these features, this project should be of interest to you.

Mining PDFs for parallel data

(Christian Buck cbuck@lantis.de)
A large part of the nice parallel data out there is hidden in .pdf files. Think manuals, technical specifications etc. Let's find parallel documents - starting with a large list of PDFs collected from CommonCrawl and collect all that precious data.
We will mostly rely on existing tools such as PDF2XML, Bitextor, and Hunalign.
Open questions:

- How best to extract text. Get all tools running on EC2 instance
- How to align documents when URL matching fail? Filesize? Metadata?
- Identify domains with large amounts of good data
- Build processing pipeline without storing all pdfs

Let's Dockerize Moses!

(Ulrich.Germann@gmail.com)

"Docker containers wrap up a piece of software in a complete filesystem that contains everything it needs to run: code, runtime, system tools, system libraries – anything you can install on a server. This guarantees that it will always run the same, regardless of the environment it is running in." (from the Docker web site)
The goal of this project is to wrap up a Moses server in a Docker app that can be easily downloaded and deployed (and more importantly, learn how to do it!) If things go well, we can add online system configuration via a web page and also dockerize MateCat, an OSS CAT interface.

Refactor the Moses Phrase Table Lookup

(Ulrich.Germann@gmail.com)
Currently, phrase-based Moses first creates a list of phrase table lookup requests and then sequentially queries all phrase tables for each of these requests. For parallelization and distributed very large systems, it would be useful to have each phrase table produce a phrase

dictionary for a given input and then merge these dictionaries.