PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 102 OCTOBER 2014 37–46

The Machine Translation Leaderboard

Matt Post, Adam Lopez

Human Language Technology Center of Excellence, Johns Hopkins University

Abstract

Much of an instructor's time is spent on the management and grading of homework. We present the Machine Translation Leaderboard, a platform for managing, displaying, and automatically grading homework assignments. It runs on Google App Engine, which provides hosting and user management services. Among its many features are the ability to easily define new assignments, manage submission histories, maintain a development / test set distinction, and display a leaderboard. An entirely new class can be set up in minutes with minimal configuration. It comes pre-packaged with five assignments used in a graduate course on machine translation.

1. Introduction

Much of an instructor's time is spent on the management and grading of homework. For many types of learning, such as the grading of essays, this time is a necessary and critical component of the learning process. But there are also many types of assignments that are easily automatable. Time spent grading them, and in managing assignment infrastructure (whether physical or digital), are drains on the instructor's limited resources which could be better spent elsewhere.

For homework assignments in the data sciences, grading can be automated using sites like `kaggle.com`, where students can upload solutions to empirical problems posed by instructors. Unfortunately, kaggle does not allow instructors to use custom evaluation measures, which makes it a poor fit for fields like machine translation that use idiosyncratic, domain-specific metrics.

To help with this, we present the Machine Translation Leaderboard (MTL), a platform for managing, displaying, and automatically grading homework assignments. It runs on Google App Engine, which provides hosting, authentication, and user

 Corresponding author: post@cs.jhu.edu
Cite as: Matt Post, Adam Lopez. The Machine Translation Leaderboard. The Prague Bulletin of Mathematical Linguistics No. 102, 2014, pp. 37–46. doi: 10.2478/pralin-2014-0012.

management services tied to a Google account. Students use a web interface to upload the results of their assignments (not code), which are then automatically graded and displayed. Among the MTL's many features are the ability to easily define new assignments, manage student submissions and submission histories, automatically grade them, maintain a development / test set distinction, and display a competitive leaderboard. An entirely new class can be setup in minutes, with minimal configuration. Packaged with the MTL are five assignments: alignment, decoding, evaluation, reranking, and inflection. Each of these assignments includes baseline (default) outputs and upper bounds, and provides ample room for improvement on their metrics through student exploration of both standard and novel approaches to these classic machine translation problems.

The MTL served as the foundation of a combined graduate and undergraduate course in machine translation, taught by the authors at Johns Hopkins University in the spring of 2014. The time we saved allowed us to focus on other aspects of teaching, including a course emphasis on scientific writing.

2. Quick-Start Guide

A new class with the default set of assignments can be setup in minutes, with minimal configuration.

1. Create an account at `appengine.google.com`
2. Create a new application. The name ("Application Identifier") you choose will become part of the URL for your class, e.g., "leaderboard" will result in a class URL of `leaderboard.appspot.com`. We will use the variable `$APPID` to refer to your choice.
3. Install the Google App Engine SDK for Python from `https://developers.google.com/appengine/downloads`
4. Clone the repository

```
$ git clone https://github.com/mjpost/leaderboard.git
$ cd leaderboard
```

5. Edit `app.yaml`, changing the value of the "application" key to the identifier you chose above

```
application: $APPID
```

6. Deploy.

```
$ appcfg.py --oauth2 update .
```

The first time you do this, a web browser window will open, prompting you to authenticate.

You can now access the leaderboard interface for uploading assignment results by loading your course's base URL (`$APPID.appspot.com`), and you can access the leaderboard itself at `$APPID.appspot.com/leaderboard.html`.

2.1. Administrative accounts

By default, the Google account used to create and host the leaderboard is the administrative account. This account has the following special permissions:

- Viewing submissions from all students, regardless of their privacy settings.
- Viewing scores for all submissions on hidden test data.
- Submitting baseline, default, and oracle entries.
- Accessing back-end administration.

You may wish to grant administrative permission to co-instructors and TAs:

1. Navigate to the back-end administration page:
   `appengine.google.com/dashboard?&app_id=s~$APPID`
2. From the navigation menu, click on "Permissions" under "Administration"
3. Add the Google accounts of your colleagues

2.2. Setting Deadlines

By default, only the first assignment is enabled, and its deadline has already passed. To enable assignments, you must do three things:

1. Uncomment the assignment's scorer in the file `leaderboard.py`.

```
scorer = [
  scoring.upload_number,
  # scoring.alignment,
  # scoring.decode,
  # scoring.evaluation,
  # scoring.rerank,
  # scoring.inflect,
]
```

2. Adjust the assignment's due date. Edit `scoring/upload_number.py` and set the value of the `deadline` variable:

```
deadline = datetime.datetime(2014, 07, 17, 23, 59)
```

3. Deploy the changes:

```
$ appcfg.py --oauth2 update .
```

2.3. The Leaderboard

Visit `$APPID.appspot.com/leaderboard.html` to view the leaderboard. It displays a grid whose columns are assignments and whose rows are student entries. The rows are sorted from best to worst according to the most recent assignment.

Note that students have the option to hide their results from the leaderboard. This settings only hides their results from other students; accessed from an administrative account, the leaderboard displays everyone's results, denoting hidden students with strikeout text. Therefore, if you are displaying the leaderboard on a project, be sure to logout from your `appspot.com` account before accessing the leaderboard.

3. Modifying and Creating Assignments

3.1. Data Model and API

We have pre-packaged five assignments with the leaderboard, but it is easy to add new assignments. Before doing so, it is useful to understand the basic data model and API implemented in `leaderboard.py`. It defines two types of database records using Google's NDB (entity database) API. The first is a Handle record, which corresponds to a user that appears on the leaderboard.

```
class Handle(ndb.Model):
  user = ndb.UserProperty() # handle with no user belongs to admins
  leaderboard = ndb.BooleanProperty()
  handle = ndb.TextProperty()
  submitted_assignments = ndb.BooleanProperty(repeated=True)
```

Handles are managed by the leaderboard code and need not be modified by assignment code. New assignment types are more likely to interact with the Assignment record, which corresponds to a single student submission for an assignment.

```
class Assignment(ndb.Model):
  handle = ndb.KeyProperty()
  number = ndb.IntegerProperty()
  filename = ndb.StringProperty()
  filedata = ndb.BlobProperty()
  score = ndb.FloatProperty()
  test_score = ndb.FloatProperty()
```

```
percent_complete = ndb.IntegerProperty()
timestamp = ndb.DateTimeProperty(auto_now_add=True)
```

When a student uploads a solution to an assignment, the leaderboard code sets several of these fields, including `handle`, `number`, `filename`, `filedata`, and `timestamp`. An assignment will mostly interact with `score`, `test_score`, and `percent_complete`. Rather than modify these fields directly, it is preferred to modify them through a callback function that each new assignment type must provide with the following signature:

```
def score(data, assignment_key, test=False)
```

When a student uploads their results, this function is invoked twice: once with `test=False` and once with `test=True`, to provide development and test set scores, respectively. Each call provides the full contents of the uploaded file in the `data` field and the NDB key of the new assignment record in the `assignment_key` field. Although the callback may use this key to update the `score` or `test_score` fields directly, this is not the preferred way to update scores. Instead, `score(...)` should return it as the first value of a two-element tuple that the caller expects. The second element of the tuple is used to update `percent_complete` of the Assignment record. So, if an assignment can be scored quickly, the correct behavior is to simply return `(s, 100)`, where `s` is the computed score.

However, in special cases computing an assignment's score may require some time to compute, and the leaderboard provides functionality to handle this. In this case, the `score(...)` callback may return `(float("-inf"), 0)`, to indicate that the score has not yet been computed. The callback should then invoke a new background task to complete the scoring behavior, and this function should update `percent_complete` periodically until the score is computed. The leaderboard uses this information to display a progress bar to the student. An example can be found in `scoring/decode.py`.

3.2. Pre-packaged Assignments

The Leaderboard comes pre-packaged with five assignments. Instructions for each assignment appear on our course webpages (`http://mt-class.org/jhu/`), under the Homework tab.

3.2.1. Assignment 1: Align

In this assignment, the input is a parallel text and students must produce word-to-word alignments.[1] The pre-packaged version of the assignment scores alignments

[1] `http://mt-class.org/jhu/hw1.html`

Table 1. List of included assignments. All file locations are relative to the `scoring` subdirectory.

Assignment	Description	File
Setup	Make sure everything is working	`upload_number.py`
Alignment	Implement a word aligner	`alignment.py`
Decoding	Maximize the model score of a decoder	`decode.py`
Evaluation	Choose the better of MT system outputs	`evaluation.py`
Reranking	Rerank k-best lists by adjusting feature weights	`rerank.py`
Inflection	Choose the appropriate inflection for each of a sequence of lemmas	`inflect.py`

against 484 manually aligned sentences of the Canadian Hansards.[2] The alignments were developed by Och and Ney (2000), which we obtained from the shared task resources organized by Mihalcea and Pedersen (2003). We use the first 37 sentences of the corpus as development data and the remaining 447 as test. The scorer is implemented in `scoring/alignment.py` with data in `scoring/alignment_data`. To score against a different dataset, simply change the data files.

3.2.2. Assignment 2: Decode

In this assignment, the input is a fixed translation model and a set of input sentences, students and they must produce translations with high model score.[3] The model we provide is a simple phrase-based translation model (Koehn et al., 2003) consisting only of a phrase table and trigram language model. Under this simple model, for a French sentence f of length I, English sentence e of length J, and alignment a where each element consists of a span in both e and f such that every word in both e and f is aligned exactly once, the conditional probability of e and a given f is as follows.[4]

$$p(e, a|f) \propto \prod_{\langle i,i',j,j'\rangle \in a} p(f_i^{i'}|e_j^{j'}) \prod_{j=1}^{J+1} p(e_j|e_{j-1}, e_{j-2}) \quad (1)$$

To evaluate output, we compute the conditional probability of e as follows.

[2] `http://www.isi.edu/natural-language/download/hansard/`

[3] `http://mt-class.org/jhu/hw2.html`

[4] For simplicity, this formula assumes that e is padded with two sentence-initial symbols and one sentence-final symbol, and ignores the probability of sentence segmentation, which we take to be uniform.

$$p(e|f) \propto \sum_{a} p(e, a|f) \tag{2}$$

Note that this formulation is different from the typical Viterbi objective of standard beam search decoders, which do not sum over all alignments, but approximate $p(e|f)$ by $\max_a p(e, a|f)$. Though the computation in Equation 2 is intractable (DeNero and Klein, 2008), it can be computed in a few minutes via dynamic programming on reasonably short sentences, a criterion met by the 48 sentences we chose from the Canadian Hansards. The corpus-level probability is then the product of all sentence-level probabilities in the data. Since this computation takes more than a few seconds, we added functionality to the leaderboard to display a progress bar, which can be reused by other custom scorers following the methods used in `scoring/decode.py`. The corresponding datasets are in `scoring/decoding_data`. To score against a different dataset, simply change the data files.

3.2.3. Assignment 3: Evaluate

In this assignment, the input is a dataset in which each sample consists of a reference sentence and a pair of translation outputs. The task is to decide which of the translation outputs is better, or if they are of equal quality.[5] Hence the task is a three-way classification problem for each input, optionally using the reference to compute features (which might include standard evaluation measures such as BLEU). To evaluate, results are compared against human assessments of the translation pairs, taken from the 2012 Workshop on Machine Translation (Callison-Burch et al., 2012). In our homework assignments, we also provided a training dataset for which human assessments are provided so that students can train classifiers for the problem. The scorer is implemented in `scoring/evaluation.py` with data in `scoring/eval_data`. To score against a different dataset, simply change the data files.

3.2.4. Assignment 4: Rerank

In this assignment, the input consists of n-best lists of translations and their associated features produced by a machine translation system on a test corpus. The task is to select the best translation for each input sentence according to BLEU, computed against a hidden reference sentence. The n-best lists were provided by Chris Dyer as an entry for the Russian-English translation task in the 2013 Workshop on Machine Translation (Bojar et al., 2013). The scorer is implemented in `scoring/rerank.py`, and the corresponding datasets are in `scoring/rerank_data`. To score against a different dataset, simply change the data files.

[5] `http://mt-class.org/jhu/hw3.html`

Table 2. Assignment five (inflection) statistics.

split	sentences	tokens	lemmas
train	29,768	518,647	35,701
dev	4,042	70,974	11,304
test	4,672	80,923	11,655

3.2.5. Assignment 5: Inflect

In this assignment, the input is a sequence of lemmatized Czech words. The task is to choose the correct inflection for each word (reminiscent of Minkov et al. (2007)). For example:

(1) Oba tyto úkoly jsou vědecky i technicky mimořádně obtížné .
oba‘2 tento úkol být vědecky_(*1ý) i-1 technicky_(*1ý) mimořádně_(*1ý) obtížný .
‘Both of these tasks are scientifically and technically extremely difficult.’

The data comes from the Prague Dependency Treebank v2.0, which is distributed through the Linguistic Data Consortium.[6] The homework assignment[7] contains instructions and a script to format the data directly from the LDC repository directory.

3.3. Creating New Assignments

The MT Leaderboard is easily extended with new assignments:

1. Create an entry for the assignment in list `scorer` near the top of `leaderboard.py`:

```
scorer = [
  scoring.upload_number,
  scoring.new_assignment,
  # scoring.alignment,
  # scoring.decode,
  # scoring.evaluation,
  # scoring.rerank,
  # scoring.inflect,
]
```

2. Next, create a file `scoring/new_assignment.py`. This file must define four variables (the assignment name, a text description of its scoring method for the

[6] `https://catalog.ldc.upenn.edu/LDC2006T01`

[7] `http://mt-class.org/jhu/hw5.html`

leaderboard header, a boolean indicating the scoring method sort order, and the assignment deadline) and two functions, `score(...)` and `oracle()`

```
$ cd scoring
$ cp upload_number.py new_assignment.py
# Edit new_assignment.py
```

3. Place any data in the directory `scoring/new_assignment_data/`.

That's it. Assignments become available as soon as they are listed in the main `leaderboard.py` script, and students can upload assignments as long as the deadline hasn't passed.

4. Case Study

In our spring 2014 class at Johns Hopkins, we received 307 submissions from 17 students over five assignments using the leaderboard as described here. Students responded positively to the leaderboard, for instance commenting that "The immediate feedback of the automatic grading was really nice". Some students used the leaderboard grader for a large number of experiments, which they then reported in writeups. For further information on how we incorporated the leaderboard into our class, empirical results, and student responses, see Lopez et al. (2013).

The MTL is only one component of a good class on machine translation. The time we saved was put into other tasks, including an emphasis on scientific writing: Students were required to submit a thorough ACL-style writeup of every homework assignment, including a description of the problem, a description of their approach, and quantitative and qualitative analysis of their findings. These writeups were graded carefully, and students received feedback on their writing. We also required students to submit their code, which we manually reviewed.

5. Future Work

With the display of a leaderboard sorted by student scores against hidden development data, the MT Leaderboard provides a competitive environment in hopes of motivating students to experiment with different approaches. However, competition isn't always the best motivator; some of our students chose to hide their handles from the leaderboard. We considered but did not implement a more cooperative approach in which students work together to improve an oracle computed over all of their submissions. For example, in the alignment setting, we could select, for each sentence, the one with the best AER across all students, and then compute AER over the whole set. At submission time, the student could then be shown how much their submission increased the oracle score. This idea could also be extended to system combination, for example, by having student submissions vote on alignment links.

Acknowledgements

We thank Chris Dyer for improving our assignments, and the students of our 2014 class at Johns Hopkins for testing everything out.

Bibliography

Bojar, Ondřej, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. Findings of the 2013 workshop on statistical machine translation. In *Proc. of WMT*, 2013.

Callison-Burch, Chris, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. Findings of the 2012 workshop on statistical machine translation. In *Proc. of WMT*, 2012.

DeNero, John and Dan Klein. The complexity of phrase alignment problems. In *Proc. of ACL*, 2008.

Hajič, Jan, Jarmila Panevová, Eva Hajičová, Petr Sgall, Petr Pajas, Jan Štěpánek, Jiří Havelka, Marie Mikulová, Zdeněk Žabokrtský, and Magda Ševčíková Razímová. Prague Dependency Treebank 2.0. LDC2006T01, Linguistic Data Consortium, Philadelphia, PA, USA, ISBN 1-58563-370-4, Jul 2006, 2006. URL `http://ufal.mff.cuni.cz/pdt2.0/`.

Koehn, Philipp, Franz J. Och, and Daniel Marcu. Statistical phrase-based translation. In *Proc. of NAACL*, 2003.

Lopez, Adam, Matt Post, Chris Callison-Burch, , Jonathan Weese, Juri Ganitkevitch, Narges Ahmidi, Olivia Buzek, Leah Hanson, Beenish Jamil, Matthias Lee, Ya-Ting Lin, Henry Pao, Fatima Rivera, Leili Shahriyari, Debu Sinha, Adam Teichert, Stephen Wampler, Michael Weinberger, Daguang Xu, Lin Yang, and Shang Zhao. Learning to translate with products of novices: a suite of open-ended challenge problems for teaching MT. *Transactions of the Association for Computational Linguistics*, (1):165–178, 2013.

Mihalcea, Rada and Ted Pedersen. An evaluation exercise for word alignment. In *Proc. on Workshop on Building and Using Parallel Texts*, 2003.

Minkov, Einat, Kristina Toutanova, and Hisami Suzuki. Generating complex morphology for machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, volume 45, pages 128–135, 2007.

Och, Franz Josef and Hermann Ney. Improved statistical alignment models. In *Proc. of ACL*, 2000.

Address for correspondence:
Matt Post
`post@cs.jhu.edu`
Human Language Technology Center of Excellence
Johns Hopkins University
810 Wyman Park Drive
Baltimore, MD 21211