Baseline: A Library for Rapid Modeling, Experimentation and Development of Deep Learning Algorithms Targeting NLP

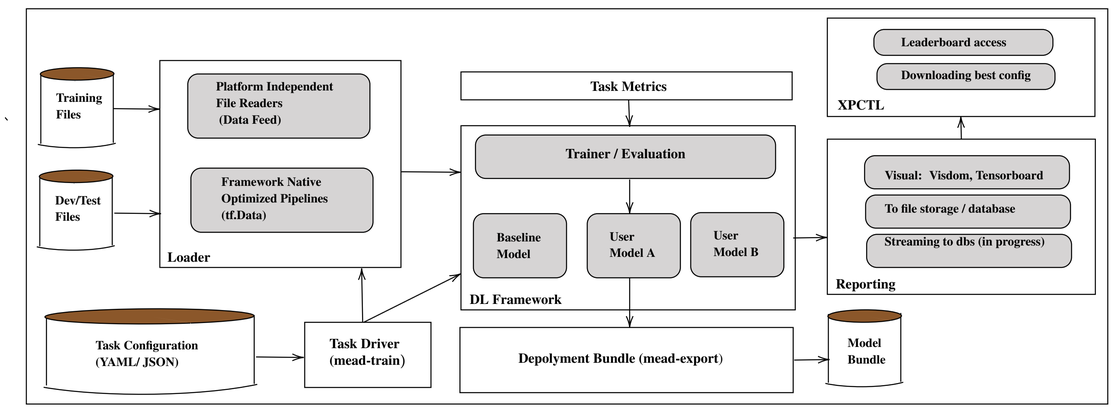
Daniel Pressel, Sagnik Ray Choudhury, Brian Lester, Yanjie Zhao, Matt Barta
Interactions Digital Roots, 330 E Liberty St, Ann Arbor, MI 48104
Email: {dpressel, schoudhury, blester, yzhao, mbarta}@interactions.com

Introduction

NLP is dominated by deep learning, but has some recurring issues:

- **Research NLP codes are stand alone applications** : Error-prone and time consuming development process, especially for common parts (data loading, training and evaluation).
- **Weaker baselines**: Performances reported in the literature are hard to reproduce: computationally expensive to find the exact hyper-parameters and evaluation codes are often incorrect.
- **Non-tractable experiments** : Experimental configurations (dataset, trainer, model hyper-parameters) are not tracked properly leading to wrong comparisons.

Baseline provides a plug and play architecture to solve these problems:



Core: Architecture & Supported Tasks

Supported NLP tasks: classification, sequence tagging, language modeling and encoder-decoder.

- **Data loader**: Readers for common file formats for all four tasks.
- **Models**: Model implementations for each task in TensorFlow, PyTorch and experimental support for DyNet.
- **Strong baselines**: The implemented models are **strong baselines**: exact hyper-parameters are provided for the best results. The repository has the most updated results for each task.
- **Utility functions**: Code for common evaluation metrics, common architecture layers and paradigms.
- **Trainer**: Multiple optimizers, early stopping, and various learning rate schedules.

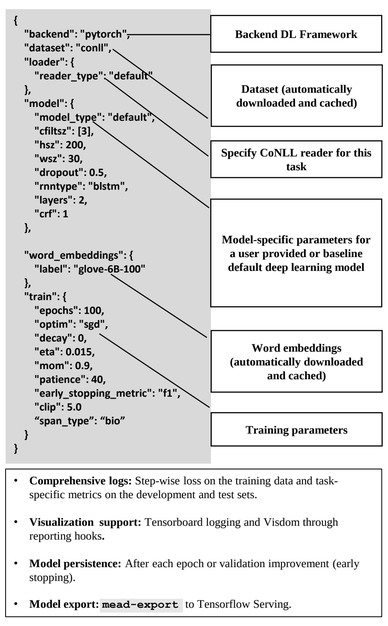
Model architecture development is the most common use case and is extremely easy with Baseline:

- Write a python file with your model (in one of the supported frameworks).
- Override the methods `create_model()` and `load_model()` in the base class.
- Save it in the format `<task-name>_<model-name>.py` and put the file in your PYTHONPATH.
- Pass `model-name` as an argument to the trainer program.

All readers and trainers can be extended similarly.

MEAD: Modeling, Experimentation & Development

Train with MEAD (`mead_train`) from a configuration file (JSON/YAML), automatically store experiment results including hyper-parameters (uniquely identified by SHA1 hash) to XPCTL, discussed in the next section.



- **Comprehensive logs:** Step-wise loss on the training data and task-specific metrics on the development and test sets.
- **Visualization support:** Tensorboard logging and Visdom through reporting hooks.
- **Model persistence:** After each epoch or validation improvement (early stopping).
- **Model export:** `mead-export` to Tensorflow Serving.

```
$ head -n 10 mead/config/embeddings.json
[
 {
   "label": "glove-twitter-27B",
   "file": "http://nlp.stanford.edu/data/glove.twitter.27B.zip",
   "sha1": "dce69c404025a8312c323197347695e81fd529fc",
   "dsz": 200
 },
```

```
$ head -n 10 mead/config/datasets.json
[
 {
   "train_file": "train.txt",
   "valid_file": "valid.txt",
   "test_file": "test.txt",
   "download": "https://www.dropbox.com/s/5g8en2jc9951omu/ptb.tar.gz?dl=1",
   "sha1": "56aacd9bd3aeffb34a9536e8de2341a8d6770f7b",
   "label": "ptb"
 }
```

```
$ mead-train --config config/sst2.json
using /home/dpressel/.bl-data/ as data/embeddings cache
TensorFlow backend
Clean
files for https://www.dropbox.com/s/7jyi4pi894bh2qh/sst2.tar.gz?dl=1 found in
cache, not downloading
[train file]: /home/dpressel/.bl-
data/31eb669609c65af3aa68a381fc760c4eaf801917/stsa.binary.phrases.train
[valid file]: /home/dpressel/.bl-
data/31eb669609c65af3aa68a381fc760c4eaf801917/stsa.binary.dev
[test file]: /home/dpressel/.bl-
data/31eb669609c65af3aa68a381fc760c4eaf801917/stsa.binary.test
Max sentence length 56, requested length 100
files for https://s3.amazonaws.com/mordecai-geo/GoogleNews-vectors-
negative300.bin.gz found in cache
embedding file location: /home/dpressel/.bl-
data/be9bfda5fdc3b3cc7376e652fc489350fe9ff863
```

XPCTL

- XPCTL (experiment control) can be used to track and compare the results with the previous ones by providing a command line interface to a database.
- For experiments run through MEAD, XPCTL provides commands to upload the configuration files and logs to a database. This database works as a leaderboard.
- The results for a problem (<task, dataset>) can be sorted by any evaluation metric, filtered for particular users and limited by a number.
- Configuration files can be downloaded.
- The current implementation supports common databases including MongoDB, PostgreSQL and SQLite, base classes provide extensibility for others.

```
$ xpctl putresult classify config/sst2.json reporting-22003.log A-baseline
the config file at config/sst2.json doesn't exist, provide a valid location
dpressel@dpressel:~/dev/work/baseline/python$ xpctl putresult classify mead/config/sst2.json reporting-22003.log A-
baseline
db connection successful with [host]: localhost, [port]: 27017
updating results for existing task [classify] in host [localhost]
results updated, the new results are stored with the record id: 5afcd32cb5536c56a6c8c5bf
```

```
$ xpctl --config ~/xpctlcred.json results classify test SST2 --metric=acc
db connection successful with [host]: localhost, [port]: 27017
                        id username       label dataset                  sha1 date acc
3  5af34c9fb5536c53bb07bc46      None Kim2014-SST2-TF-2epochs   SST2 8ab6ab6ee8fdf14b111223e2edf48750c30c7e51  2018-05-
09T19:31:42.984408 0.875343
0  5af36f9bb5536c60d1e2ccc1  dpressel Kim2014-SST2-TF-2epochs    SST2 8ab6ab6ee8fdf14b111223e2edf48750c30c7e51  2018-05-
09T22:00:49.195327 0.874794
1  5af3c937b5536c75fb1529bc  dpressel Kim2014-SST2-TF-2epochs    SST2 8ab6ab6ee8fdf14b111223e2edf48750c30c7e51  2018-05-
10T04:23:19.586245 0.874245
2  5af34c0bb5536c533c9b6ecc      None Kim2014-SST2-TF-2epochs   SST2 8ab6ab6ee8fdf14b111223e2edf48750c30c7e51  2018-05-
09T19:29:15.696297 0.871499
4  5a0e0104b5536c46d142532f  dpressel self-attn-l3-h6    SST2 72f2cce2ee4bcc755e527e03e05788442b658355  2017-11-16
21:20:04.735000 0.851099
```

Conclusion & Future Work

- Baseline's goal is to help automate the frustrating parts of the process of model development and deployment to allow researchers to focus on innovation.
- It is currently used in a production environment for various problems, attesting to the fact that it is suitable for a complete research-to-deployment pipeline.
- The library has implementations for many strong baseline models which we will continue to update and improve as the state-of-the-art changes (see the repository for most updated results).
- Future versions of the software will attempt to improve the development process further by assisting with automatic parameter tuning and model selection, support for more deep learning frameworks, improved visualization, and a simpler cross-platform deployment mechanism.

Acknowledgements

Thanks to Patrick Haffner, Nick Ruiz and John Chen at Interactions LLC for valuable discussions.