Baseline: Strong, Extensible, Reproducible, Deep Learning Baselines for NLP

Daniel Pressel, Brian Lester, Sagnik Ray Choudhury, Matt Barta, Yanjie Zhao, Amy Hemmeter
Interactions Digital Roots
{dpressel, blester, schoudhury, mbarta, yzhao, ahemmeter}@interactions.com

Abstract

Natural Language Processing is now dominated by deep learning models. Baseline[1] is a library to facilitate reproducible research and fast model development for NLP with deep learning. It provides easily extensible implementations and abstractions for data loading, model development, training, hyper-parameter tuning, deployment to production, and a leaderboard to track experimental results.

1 Introduction

In the past few years, deep learning models have become common in NLP literature, and comparing new models against deep baselines has become standard practice. The quality of baseline implementation varies drastically, leaving some uncertainty about the importance of results. In some cases, researchers compare against weak baselines which may yield large relative gains [1]. Deep learning implementations frequently provide their own from-scratch replication, including their own training pipeline, error metrics and models, which can introduce errors and cause variation in performance. Also, deep learning models have inherent randomness – models with the same hyper-parameters will have different performance across multiple runs and it is often unclear if reported performance is the mean or the max of all the models trained. Careful hyper-parameter tuning is critical for baselines to accurately represent the performance improvement of a new method [2]. Our software, Baseline [3], is designed to make it easy for researchers to create new deep models for common NLP tasks in a way that is reproducible and built on strong baselines. A short demonstration of the software is provided [2].

2 Baseline

Baseline provides the following benefits to researchers:

1. Offers a comprehensive pipeline from pre-processing to training and evaluation in an "a la carte" format that allows for easy swapping of datasets, models, and other aspects of the process
2. Makes it easy for users to test new techniques against strong baselines to get an accurate impression of their improvements
3. Encourages reproducible research using the experimental control module (`XPCTL`) which saves model results and experimental details and makes access easy through a simple leaderboard interface
4. Allows the user to choose between the three most popular deep learning frameworks for NLP – Tensorflow [4], PyTorch [5] and DyNet [6]

[1] `https://github.com/dpressel/baseline`
[2] `https://www.youtube.com/watch?v=AwTjp1RihvU`

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

5. Provides a tool to automatically tune the hyper-parameters for an experiment
6. Represents embeddings as arbitrary sub-graphs, providing an easy way to leverage complex embedding strategies such as contextual word embeddings
7. Provides tools to easily export the models to production environments such as `Tensorflow Serving`

The Baseline software (see figure 1 a) is comprised of a core API, a module for Modeling, Experimentation And Development (`MEAD`), Experiment Control (`XPCTL`), and Hyper-parameter Control (`HPCTL`).

Baseline's components offer support for all aspects of the deep learning process. At its core, Baseline makes it possible to extend every part of the training process, allowing the researcher to provide custom readers, vectorizers, embeddings, models and trainers with simple extension points. These components are designed to maximize reuse and minimize the amount of code required to create new models and architectures. `MEAD` automatically downloads datasets and embeddings from the original source on first use, backed by a configurable index and a local cache. It supports a generalized training facility with a simple JSON or YAML configuration that allows users full control over each component involved in training. Baseline has full Docker support which removes the possibility of dependency version mismatches, and makes it easy to run and isolate results.

To facilitate reproducible research, Baseline has the ability to save models, datasets, hyper-parameters, and other information required to repeat an experiment to a database. `XPCTL` creates a command line interface to enable results tracking and comparisons and stores the models in a persistent location. We currently support both Postgres and MongoDB backends. A user can retrieve the configuration file for any previously recorded experiment. The results of experiments can be sorted by any metric or filtered for particular users. `XPCTL` aggregates results from different instances of the same model and reports the mean and standard deviation across runs to accurately represent the performance metrics of an experimental model (see figure 1 b.). The `reporting` section within the settings file allows training through `MEAD` and automatically pushes experimental results to the leaderboard database.

Baseline provides a set of strong baseline models for classification, sequence tagging, encoder-decoder and language modeling, four of the most common tasks in NLP. It supports CNNs, LSTMs and Neural Bag of Words (NBoW) models for classification [7, 8, 9, 10], CNN-BLSTM-CRFs for tagging [11], LSTM word and character language modeling [12, 13, 14] and Seq2Seq encoder-decoders with attention [15, 16]. Provided baselines have comparable performance and options across all frameworks. In addition, extensions are also provided for State-of-the-Art models such as ELMo [17]. The most up-to-date results are available in the repository.

At the core of Baseline is a "go to the user" philosophy – it supports the most popular deep learning frameworks as backends (Tensorflow, PyTorch and DyNet), providing an implementation in each framework for each task and model. All backends perform comparably in terms of accuracy.

When using deep learning models, both baselines and proposed models require extensive hyper-parameter tuning to achieve optimal results [2]. An automatic approach to finding and validating good hyper-parameters is necessary to maintain high velocity during research. Using a `MEAD` configuration with sampling directives, our new module Hyper-parameter Control (`HPCTL`) launches and manages parallel jobs to perform the hyper-parameter search and allows culling of low performing jobs. While random search is often superior to grid search [18] `HPCTL` can also perform a grid search over any combination of parameters. `HPCTL` has access to previous results to allow for cascading sampling procedures and allows users to supply their own sampling functions, in keeping with our goals to allow users to adapt our tools to their needs while maintaining high performance.

Another important feature of Baseline is the support for contextual embeddings. Until recently, DNN models for NLP have focused primarily on pre-trained word embeddings trained on large unsupervised datasets [19, 20] to map a word to a continuous dense representation. However, recent work in NLP has focused heavily on transfer learning and contextual embeddings, where a language model is pre-trained on external data and used within a deep model to provide a better context representation [17, 21, 22, 23, 24]. These pre-trained contextual embeddings can yield large gains in performance, making their use appealing, but they represent a more complex sub-graph than the single look-up table layer which previously dominated deep models. Often these representations are concatenated with pre-trained word embeddings and treated as model inputs. A challenge for

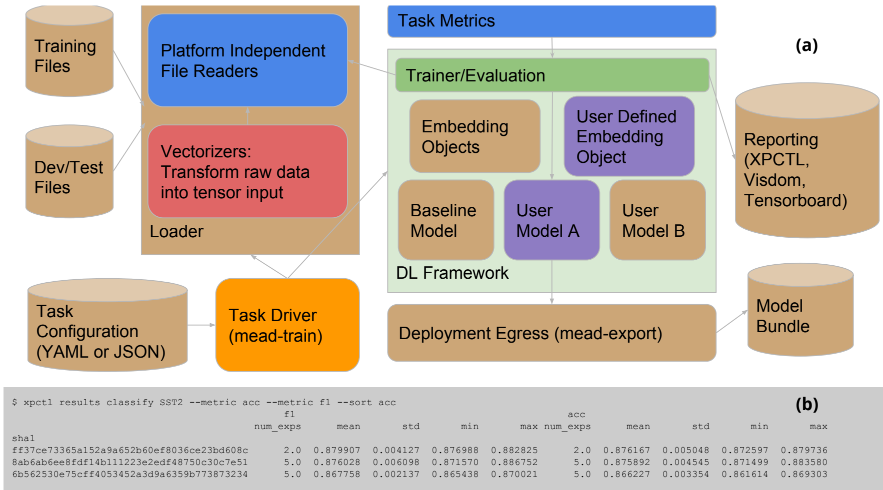

Figure 1: a. Architecture of Baseline. b. Results from a sample XPCTL query.

effective software is to represent such arbitrary embeddings sub-graphs, and use them in a way that makes the network easy to extend, generalize, and augment for research experiments.

Baseline supports common formats for word embeddings, such as those used by Word2Vec and GloVe, which yield a framework-specific look-up table sub-graph in the model. It also supports extensions such as LSTMs, transformers or convolutions on the input sub-graph, making contextual embeddings straightforward and idiomatic. The `feature` section of the `MEAD` configuration describes how the text tokens are converted to inputs necessary for the underlying backend framework. First, a `vectorizer` is used to build initial dictionaries over tokens and ultimately do the feature extraction, using a vocabulary provided to them. These `vectorizers` can be serialized during training and reused for test. The vocabularies are automatically saved by `MEAD`. For tagging, dictionary-based `vectorizers` are also supported, allowing features to be composed from multiple columns of a CoNLL format file.

In addition to its value in a research environment, Baseline also provides functionality for easy deployment of DNN models. In production, DNNs are often deployed within a model serving framework such as `TensorFlow Serving`. Baseline provides exporter utilities for all implementations. Some customization may be required for user-defined models, for which we provide interfaces.

The design patterns used in Baseline make it easy to reuse components and simplify the process of new model development. This makes it simple for a researcher to build custom models and complex training regimes. The readers can re-use the same `vectorizers` for reading files for different tasks. The embeddings sub-graph concept allows models to delegate complex word representations, simplifying model inputs and data feeding. The base models provide a set of virtual functions that can be individually overridden, minimizing the amount of code needed to develop new models.

3 Conclusion and Future Work

Deep learning has changed the landscape of NLP research, and is still evolving quickly. The goal of Baseline is to provide strong baselines, facilitate reproducible research, and allow fast model development and deployment. It follows recent best practices in deep learning for NLP and makes it easier to iterate new ideas.

Future efforts on Baseline will focus on solidifying the infrastructure developed to make publishing models and results simpler across the community as well as unifying the flow of model development with push-button tooling. We will continually update the baselines and underlying building blocks to reflect the community direction over time. We also anticipate addition of more tasks, metrics, and hope to support more frameworks.

References

[1] Timothy G. Armstrong, Justin Zobel, William Webber, and Alistair Moffat. Relative significance is insufficient: Baselines matter too. 2009.

[2] Gábor Melis, Chris Dyer, and Phil Blunsom. On the state of the art of evaluation in neural language models. *ICLR*, 2018.

[3] Daniel Pressel, Sagnik Ray Choudhury, Brian Lester, Yanjie Zhao, and Matt Barta. Baseline: A library for rapid modeling, experimentation and development of deep learning algorithms targeting nlp. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 34–40. Association for Computational Linguistics, 2018.

[4] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Gordon Murray, Benoit Steiner, Paul A. Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016, Savannah, GA, USA, November 2-4, 2016.*, pages 265–283, 2016.

[5] Adam Paszke, Sam Gross, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[6] Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. Dynet: The dynamic neural network toolkit. *CoRR*, abs/1701.03980, 2017.

[7] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[8] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 2267–2273. AAAI Press, 2015.

[9] Tao Chen, Ruifeng Xu, Yulan He, and Xuan Wang. Improving sentiment analysis via sentence type classification using bilstm-crf and cnn. *Expert Systems with Applications*, 72:221 – 230, 2017.

[10] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751, 2014.

[11] Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany, August 2016. Association for Computational Linguistics.

[12] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *CoRR*, abs/1409.2329, 2014.

[13] Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *CoRR*, abs/1602.02410, 2016.

[14] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2741–2749, 2016.

[15] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3104–3112, Cambridge, MA, USA, 2014. MIT Press.

[16] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[17] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018.

[18] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

[19] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119, 2013.

[20] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[21] Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1756–1765, 2017.

[22] Sebastian Ruder and Jeremy Howard. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 328–339, 2018.

[23] Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. Learning to generate reviews and discovering sentiment, 2018.

[24] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training, 2018.