DeepZensols: A Deep Learning NLP Framework for Reproducibility

Paul Landes, Barbara Di Eugenio, and Cornelia Caragea

Department of Computer Science, University of Illinois Chicago

{plande2,bdieugen,cornelia}@uic.edu

Introduction

A deep learning NLP framework that:

- Allows hot-swapping features and embeddings without further processing and re-vectorizing the dataset (Table 1).
- Reproduces results, a issue that persists in the machine learning (ML) community (Olorisade et al. 2017).
- Includes easily configurable standard models allowing quick experimentation and without any coding.
- Thorough API and overview documentation with examples.

Technology Stack

Each library is a `pip` installable Python packages.

mednlp

- Medical natural language parsing and utility library.
- Integration with MedCAT and cTAKES for medical concept NER and entity linking with cui2vec embeddings.
- Integration with Unified Medical Language System (UMLS).

deepnlp

- Natural language deep learning specific layers: 1D CNN, Word Embedding with and without CRF.
- Integration with HuggingFace transformers for token and text classification and masked language models.
- Easily configureable large range of word embeddings including GloVE, word2vec, fastText.

deeplearn

- Easy to configure deep learning framework.
- Built on PyTorch.
- Layers: CNN, CRF, BiLSTM-CRF.
- Real-time performance, metrics benchmarking.

nlparse

- Parse natural language with spaCy, normalize text and create features.
- Fast feature saves.
- Scoring: ROUGE, BLEU, SemEval-2013 Task 9.1.

datdesc

- Annotate data frames as explainable information.
- Format results in LaTeX tables.
- Bayesian optimization.
- Excel output data annotation.

util (core)

- Inversion of control application support like Hydra/Java Spring.
- Command line interface bindings to Python dataclasses and methods.
- File system persistence API as Python decorators.
- Juypter Notebook integration.

Figure 1: The Python library stack in order of dependency.

Batch

Batch decoding groups data on the file system in mini-batches:

1. Configure the desirable features for this training run.
2. Read features per batch as a two level directory structure.
3. Reassemble features by batch.
4. Decode each batch as a tensor from the file system.
5. Preemptively copy tensors to GPU for each epoch.

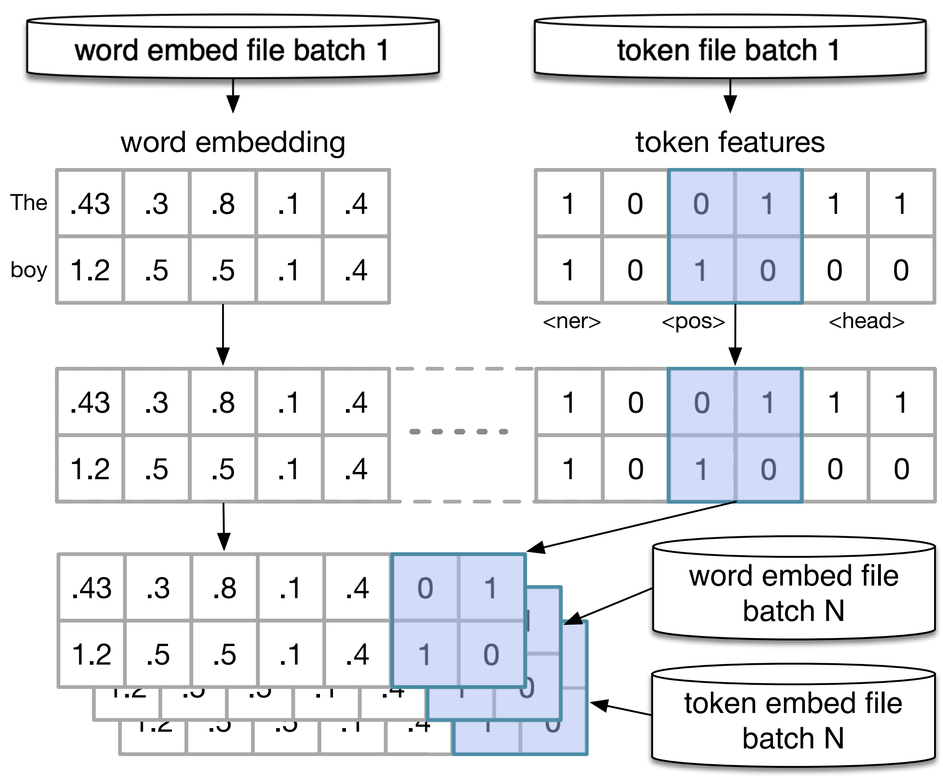

Figure 2: Batch decoding "stitches" mini-batches together from files containing features for the current run.

Runtime

Runtime analysis on three model types across three datasets:

- CoNNL 2003 (Tjong Kim Sang et al. 2003) for NER.
- Movie review corpus (Socher et al. 2013) for sentiment.
- Clickbate corpus (Chakraborty et al. 2016) for classification.

Data	Model	Duration	Batch	GPU	Both
NER	BERT	1:06:04	04:23	00:12	04:35
	GloVe	34:08	04:19	05:41	10:00
Mov	BERT	21:19	02:04	-00:26	01:38
	GloVe	05:03	03:07	01:20	04:27
CB	BERT	05:48	01:50	-00:01	01:49
	GloVe	05:45	01:51	03:03	04:54

Table 1: Efficiency benchmarks showing the **N**amed **E**ntity **R**ecognition, **Mov**ie review sentiment, and **C**lick**B**ate datasets. The "Duration" column lists processing latency with no batch or GPU caching in hours, minutes and seconds. The "Batch" and "GPU" columns have the caching speedup times in minutes and seconds. The "Both" column is the speedup with both batch and GPU caching are enabled.

Acknowledgments

Supported by R01 CA225446 from National Institutes of Health.