

AllenNLP: A Deep Semantic Natural Language Processing Platform

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi,
Nelson F. Liu, Matthew Peters, Michael Schmitz, Luke Zettlemoyer
Allen Institute for Artificial Intelligence

Abstract

Modern natural language processing (NLP) research requires writing code. Ideally this code would provide a precise definition of the approach, easy repeatability of results, and a basis for extending the research. However, many research codebases bury high-level parameters under implementation details, are challenging to run and debug, and are difficult enough to extend that they are more likely to be rewritten. This paper describes AllenNLP, a library for applying deep learning methods to NLP research, which addresses these issues with easy-to-use command-line tools, declarative configuration-driven experiments, and modular NLP abstractions. AllenNLP has already increased the rate of research experimentation and the sharing of NLP components at the Allen Institute for Artificial Intelligence, and we are working to have the same impact across the field.

1 Introduction

Neural network models are now the state-of-the-art for a wide range of tasks such as text classification (Howard and Ruder, 2018), machine translation (Vaswani et al., 2017), semantic role labeling (Zhou and Xu, 2015; He et al., 2017), coreference resolution (Lee et al., 2017a), and semantic parsing (Krishnamurthy et al., 2017). However it can be surprisingly difficult to tune new models or replicate existing results. State-of-the-art deep learning models often take over a week to train on modern GPUs and are sensitive to initialization and hyperparameter settings. Furthermore, reference implementations often re-implement NLP components from scratch and make it difficult to

reproduce results, creating a barrier to entry for research on many problems.

AllenNLP, a platform for research on deep learning methods in natural language processing, is designed to address these problems and to significantly lower barriers to high quality NLP research by

- implementing useful NLP abstractions that make it easy to write higher-level model code for a broad range of NLP tasks, swap out components, and re-use implementations,
- handling common NLP deep learning problems, such as masking and padding, and keeping these low-level details separate from the high-level model and experiment definitions,
- defining experiments using declarative configuration files, which provide a high-level summary of a model and its training, and make it easy to change the deep learning architecture and tune hyper-parameters, and
- sharing models through live demos, making complex NLP accessible and debug-able.

The [AllenNLP website](http://allennlp.org/)¹ provides tutorials, API documentation, pretrained models, and [source code](https://github.com/allenai/allennlp)². The AllenNLP platform has a permissive Apache 2.0 license and is easy to download and install via pip, a Docker image, or cloning the GitHub repository. It includes reference implementations for recent state-of-the-art models (see Section 3) that can be easily run (to make predictions on arbitrary new inputs) and retrained with different parameters or on new data. These pretrained models have [interactive online demos](http://demo.allennlp.org/)³

¹<http://allennlp.org/>

²[http://github.com/allenai/allennlp](https://github.com/allenai/allennlp)

³<http://demo.allennlp.org/>

with visualizations to help interpret model decisions and make predictions accessible to others. The reference implementations provide examples of the framework functionality (Section 2) and also serve as baselines for future research.

AllenNLP is an ongoing open-source effort maintained by several full-time engineers and researchers at the Allen Institute for Artificial Intelligence, as well as interns from top PhD programs and contributors from the broader NLP community. It is used widespread internally for research on common sense, logical reasoning, and state-of-the-art NLP components such as: constituency parsers, semantic parsing, and word representations. AllenNLP is gaining traction externally and we want to invest to make it the standard for advancing NLP research using PyTorch.

2 Library Design

AllenNLP is a platform designed specifically for deep learning and NLP research. AllenNLP is built on PyTorch (Paszke et al., 2017), which provides many attractive features for NLP research. PyTorch supports dynamic networks, has a clean “Pythonic” syntax, and is easy to use.

The AllenNLP library provides (1) a flexible data API that handles intelligent batching and padding, (2) high-level abstractions for common operations in working with text, and (3) a modular and extensible experiment framework that makes doing good science easy.

AllenNLP maintains a high test coverage of over 90%⁴ to ensure its components and models are working as intended. Library features are built with testability in mind so new components can maintain a similar test coverage.

2.1 Text Data Processing

AllenNLP’s data processing API is built around the notion of `Fields`. Each `Field` represents a single input array to a model. `Fields` are grouped together in `Instances` that represent the examples for training or prediction.

The `Field` API is flexible and easy to extend, allowing for a unified data API for tasks as diverse as tagging, semantic role labeling, question answering, and textual entailment. To represent the SQuAD dataset (Rajpurkar et al., 2016), for example, which has a question and a passage as inputs

and a span from the passage as output, each training `Instance` comprises a `TextField` for the question, a `TextField` for the passage, and a `SpanField` representing the start and end positions of the answer in the passage.

The user need only read data into a set of `Instance` objects with the desired fields, and the library can automatically sort them into batches with similar sequence lengths, pad all sequences in each batch to the same length, and randomly shuffle the batches for input to a model.

2.2 NLP-Focused Abstractions

AllenNLP provides a high-level API for building models, with abstractions designed specifically for NLP research. By design, the code for a model actually specifies a *class* of related models. The researcher can then experiment with various architectures within this class by simply changing a configuration file, without having to change any code.

The library has many abstractions that encapsulate common decision points in NLP models. Key examples are: (1) how text is represented as vectors, (2) how vector sequences are modified to produce new vector sequences, (3) how vector sequences are merged into a single vector.

TokenEmbedder: This abstraction takes input arrays generated by e.g. a `TextField` and returns a sequence of vector embeddings. Through the use of polymorphism and AllenNLP’s experiment framework (see Section 2.3), researchers can easily switch between a wide variety of possible word representations. Simply by changing a configuration file, an experimenter can choose between pre-trained word embeddings, word embeddings concatenated with a character-level CNN encoding, or even pre-trained model token-in-context embeddings (Peters et al., 2017), which allows for easy controlled experimentation.

Seq2SeqEncoder: A common operation in deep NLP models is to take a sequence of word vectors and pass them through a recurrent network to encode contextual information, producing a new sequence of vectors as output. There is a large number of ways to do this, including LSTMs (Hochreiter and Schmidhuber, 1997), GRUs (Cho et al., 2014), intra-sentence attention (Cheng et al., 2016), recurrent additive networks (Lee et al., 2017b), and many more. AllenNLP’s `Seq2SeqEncoder` abstracts away the

⁴<https://codecov.io/gh/allenai/allennlp>

decision of which particular encoder to use, allowing the user to build an encoder-agnostic model and specify the encoder via configuration. In this way, a researcher can easily explore new recurrent architectures; for example, they can replace the LSTMs in *any model* that uses this abstraction with any other encoder, measuring the impact across a wide range of models and tasks.

Seq2VecEncoder: Another common operation in NLP models is to merge a sequence of vectors into a single vector, using either a recurrent network with some kind of averaging or pooling, or using a convolutional network. This operation is encapsulated in AllenNLP by a `Seq2VecEncoder`. This abstraction again allows the model code to only describe a *class* of similar models, with particular instantiations of that model class being determined by a configuration file.

SpanExtractor: A recent trend in NLP is to build models that operate on *spans* of text, instead of on *tokens*. State-of-the-art models for coreference resolution (Lee et al., 2017a), constituency parsing (Stern et al., 2017), and semantic role labeling (He et al., 2017) all operate in this way. Support for building this kind of model is built into AllenNLP, including a `SpanExtractor` abstraction that determines how span vectors get computed from sequences of token vectors.

2.3 Experimental Framework

The primary design goal of AllenNLP is to make it easy to do good science with controlled experiments. Because of the abstractions described in Section 2.2, large parts of the model architecture and training-related hyper-parameters can be configured outside of model code. This makes it easy to clearly specify the important decisions that define a new model in configuration, and frees the researcher from needing to code all of the implementation details from scratch.

This architecture design is accomplished in AllenNLP using a HOCON⁵ configuration file that specifies, e.g., which text representations and encoders to use in an experiment. The mapping from strings in the configuration file to instantiated objects in code is done through the use of a *registry*, which allows users of the library to add new im-

⁵We use it as JSON with comments. See <https://github.com/lightbend/config/blob/master/HOCON.md> for the full spec.

plementations of any of the provided abstractions, or even to create their own new abstractions.

While some entries in the configuration file are optional, many are required and if unspecified AllenNLP will raise a `ConfigurationError` when reading the configuration. Additionally, when a configuration file is loaded, AllenNLP logs the configuration values, providing a record of both specified and default parameters for your model.

3 Reference Models

AllenNLP includes reference implementations of widely used language understanding models. These models demonstrate how to use the framework functionality presented in Section 2. They also have verified performance levels that closely match the original results, and can serve as comparison baselines for future research.

AllenNLP includes reference implementations for several tasks, including:

- **Semantic Role Labeling (SRL)** models recover the latent predicate argument structure of a sentence (Palmer et al., 2005). SRL builds representations that answer basic questions about sentence meaning; for example, “who” did “what” to “whom.” The AllenNLP SRL model is a re-implementation of a deep BiLSTM model (He et al., 2017). The implemented model closely matches the published model which was state of the art in 2017, achieving a F1 of 78.9% on English Ontonotes 5.0 dataset using the CoNLL 2011/12 shared task format.
- **Machine Comprehension (MC)** systems take an evidence text and a question as input, and predict a span within the evidence that answers the question. AllenNLP includes a reference implementation of the BiDAF MC model (Seo et al., 2017) which was state of the art for the SQuAD benchmark (Rajpurkar et al., 2016) in early 2017.
- **Textual Entailment (TE)** models take a pair of sentences and predict whether the facts in the first necessarily imply the facts in the second. The AllenNLP TE model is a re-implementation of the decomposable attention model (Parikh et al., 2016), a widely used TE baseline that was state-of-the-art on the SNLI dataset (Bowman et al., 2015) in

late 2016. The AllenNLP TE model achieves an accuracy of 86.4% on the SNLI 1.0 test dataset, a 2% improvement on most publicly available implementations and a similar score as the original paper. Rather than pre-trained Glove vectors, this model uses ELMo embeddings (Peters et al., 2018), which are completely character based and account for the 2% improvement.

- A **Constituency Parser** breaks a text into sub-phrases, or constituents. Non-terminals in the tree are types of phrases and the terminals are the words in the sentence. The AllenNLP constituency parser is an implementation of a minimal neural model for constituency parsing based on an independent scoring of labels and spans (Stern et al., 2017). This model uses ELMo embeddings (Peters et al., 2018), which are completely character based and improves single model performance from 92.6 F1 to 94.11 F1 on the Penn Tree bank, a 20% relative error reduction.

AllenNLP also includes a token embedder that uses pre-trained ELMo (Peters et al., 2018) representations. ELMo is a deep contextualized word representation that models both complex characteristics of word use (e.g., syntax and semantics) and how these uses vary across linguistic contexts (in order to model polysemy). ELMo embeddings significantly improve the state of the art across a broad range of challenging NLP problems, including question answering, textual entailment, and sentiment analysis.

Additional models are currently under development and are regularly released, including semantic parsing (Krishnamurthy et al., 2017) and multi-paragraph reading comprehension (Clark and Gardner, 2017). We expect the number of tasks and reference implementations to grow steadily over time. The most up-to-date list of reference models is maintained at <http://allennlp.org/models>.

4 Related Work

Many existing NLP pipelines, such as Stanford CoreNLP (Manning et al., 2014) and spaCy⁶, focus on predicting linguistic structures rather

⁶<https://spacy.io/>

than modeling NLP architectures. While AllenNLP supports making predictions using pre-trained models, its core focus is on enabling novel research. This emphasis on configuring parameters, training, and evaluating is similar to Weka (Witten and Frank, 1999) or Scikit-learn (Pedregosa et al., 2011), but AllenNLP focuses on cutting-edge research in deep learning and is designed around declarative configuration of model architectures in addition to model parameters.

Most existing deep-learning toolkits are designed for general machine learning (Bergstra et al., 2010; Yu et al., 2014; Chen et al., 2015; Abadi et al., 2016; Neubig et al., 2017), and can require significant effort to develop research infrastructure for particular model classes. Some, such as Keras (Chollet et al., 2015), do aim to make it easy to build deep learning models. Similar to how AllenNLP is an abstraction layer on top of PyTorch, Keras provides high-level abstractions on top of static graph frameworks such as TensorFlow. While Keras' abstractions and functionality are useful for general machine learning, they are somewhat lacking for NLP, where input data types can be very complex and dynamic graph frameworks are more often necessary.

Finally, AllenNLP is related to toolkits for deep learning research in dialog (Miller et al., 2017) and machine translation (Klein et al., 2017). Those toolkits support learning general functions that map strings (e.g. foreign language text or user utterances) to strings (e.g. English text or system responses). AllenNLP, in contrast, is a more general library for building models for any kind of NLP task, including text classification, constituency parsing, textual entailment, question answering, and more.

5 Conclusion

The design of AllenNLP allows researchers to focus on the high-level summary of their models rather than the details, and to do careful, reproducible research. Internally at the Allen Institute for Artificial Intelligence the library is widely adopted and has improved the quality of our research code, spread knowledge about deep learning, and made it easier to share discoveries between teams. AllenNLP is gaining traction externally and is growing an open-source community

of contributors ⁷. The AllenNLP team is committed to continuing work on this library in order to enable better research practices throughout the NLP community and to build a community of researchers who maintain a collection of the best models in natural language processing.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR abs/1603.04467*.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: A cpu and gpu math compiler in python. In *Proc. 9th Python in Science Conf*, pages 1–7.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. 2015. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *CoRR abs/1512.01274*.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Kyunghyun Cho, Bart van Merriënboer, Çağlar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

François Chollet et al. 2015. Keras. <https://keras.io>.

Christopher T Clark and Matthew Gardner. 2017. Simple and effective multi-paragraph reading comprehension. *CoRR*, abs/1710.10723.

⁷See GitHub stars and issues on <https://github.com/allenai/allennlp> and mentions from publications at <https://www.semanticscholar.org/search?q=allennlp>.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Jeremy Howard and Sebastian Ruder. 2018. Fine-tuned language models for text classification. *CoRR*, abs/1801.06146.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL*, pages 67–72.

Jayant Krishnamurthy, Pradeep Dasigi, and Matthew Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *EMNLP*.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017a. End-to-end neural coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2017b. Recurrent additive networks. *CoRR abs/1705.07393*.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of the Association for Computational Linguistics (ACL) (System Demonstrations)*.

Alexander H. Miller, Will Feng, Dhruv Batra, Antoine Bordes, Adam Fisch, Jiasen Lu, Devi Parikh, and Jason Weston. 2017. Parlai: A dialog research software platform. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 79–84.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. Dynet: The dynamic neural network toolkit. *CoRR abs/1701.03980*.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

- Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.
- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.
- Matthew E Peters, Waleed Ammar, Chandra Bhagavathula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the Association for Computational Linguistics (ACL)*.
- Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matthew Gardner, Christopher T Clark, Kenton Lee, and Luke S. Zettlemoyer. 2018. Deep contextualized word representations. *CoRR*, abs/1802.05365.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension.
- Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *ACL*.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.
- Ian H. Witten and Eibe Frank. 1999. Data mining: Practical machine learning tools and techniques with java implementations.
- Dong Yu, Adam Eversole, Mike Seltzer, Kaisheng Yao, Zhiheng Huang, Brian Guenter, Oleksii Kuchaiev, Yu Zhang, Frank Seide, Huaming Wang, et al. 2014. An introduction to computational networks and the computational network toolkit. *Microsoft Technical Report MSR-TR-2014-112*.
- Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the Association for Computational Linguistics (ACL)*.