Contemporary NLP Modeling in Six Comprehensive Programming Assignments

Greg Durrett[*] **Jifan Chen** **Shrey Desai** **Tanya Goyal**
Lucas Kabela **Yasumasa Onoe** **Jiacheng Xu**
Department of Computer Science
The University of Texas at Austin
gdurrett@cs.utexas.edu

Abstract

We present a series of programming assignments, adaptable to a range of experience levels from advanced undergraduate to PhD, to teach students design and implementation of modern NLP systems. These assignments build from the ground up and emphasize full-stack understanding of machine learning models: initially, students implement inference and gradient computation by hand, then use PyTorch to build nearly state-of-the-art neural networks using current best practices. Topics are chosen to cover a wide range of modeling and inference techniques that one might encounter, ranging from linear models suitable for industry applications to state-of-the-art deep learning models used in NLP research. The assignments are customizable, with constrained options to guide less experienced students or open-ended options giving advanced students freedom to explore. All of them can be deployed in a fully autogradable fashion, and have collectively been tested on over 300 students across several semesters.[1]

1 Introduction

This paper presents a series of assignments designed to give a survey of modern NLP through the lens of system-building. These assignments provide hands-on experience with concepts and implementation practices that we consider critical for students to master, ranging from linear feature-based models to cutting-edge deep learning approaches. The assignments are as follows:

A1. Sentiment analysis with linear models (Pang et al., 2002) on the Stanford Sentiment Treebank (Socher et al., 2013).

A2. Sentiment analysis with feedforward "deep averaging" networks (Iyyer et al., 2015) using GloVe embeddings (Pennington et al., 2014).

A3. Hidden Markov Models and linear-chain conditional random fields (CRFs) for named entity recognition (NER) (Tjong Kim Sang and De Meulder, 2003), using features similar to those from Zhang and Johnson (2003).

A4. Character-level RNN language modeling (Mikolov et al., 2010).

A5. Semantic parsing with seq2seq models (Jia and Liang, 2016) on the GeoQuery dataset (Zelle and Mooney, 1996).

A6. Reading comprehension on SQuAD (Rajpurkar et al., 2016) using a simplified version of the DrQA model (Chen et al., 2017), similar to BiDAF (Seo et al., 2016).

A1-A5 come with autograders. These train each student's model from scratch and evaluate performance on the development set of each task, verifying whether their code behaves as intended. The autograders are bundled to be deployable on Gradescope using their Docker framework.[2] These coding assignments can also be supplemented with conceptual questions for hybrid assignments, though we do not distribute those as part of this release.

Other Courses and Materials Several other widely-publicized courses like Stanford CS224N and CMU CS 11-747 are much more "neural-first" views of NLP: their assignments delve more heavily into word embeddings and low-level neural implementation like backpropagation. By contrast, this course is designed to be a survey that

[*] Corresponding author. Subsequent authors listed alphabetically.

[1] See `https://cs.utexas.edu/~gdurrett` for past offerings and static versions of these assignments; contact Greg Durrett for access to the repository with instructor solutions.

[2] For the CRF and seq2seq modeling assignments, a custom framework must be used, as Gradescope autograders cannot handle these. We grade these in a batch fashion on a single instructional machine, which poses some logistical challenges.

Assignment	Main Concepts	Output	Components				
			Linear	FFNN	Enc	Dec	Attn
A1: Sentiment (Linear)	Classification, SGD, bag-of-words	Binary	■				
A2: Sentiment (FFNNs)	PyTorch, word embeddings	Binary	■	■			
A3: HMMs and CRFs for NER	Structured prediction, dyn. prog.	Tags	■				
A4: Language Modeling	Neural sequence modeling	Token seq	■	▨	■	▨	
A5: Seq2seq Semantic Parsing	Encoder-decoder, attention	Token seq	■	▨	▨	■	■
A6: Reading Comprehension	QA, domain adaptation	Span	▨	▨	▨		▨

Table 1: Breakdown of assignments. The concepts and model components in each are designed to build on one another. A gray square indicates partial engagement with a concept, typically when students are already given the needed component or it isn't a focus of the assignment.

also covers topics like linear classification, generative modeling (HMMs), and structured inference. Other hands-on courses discussed in prior Teaching NLP papers (Klein, 2005; Madnani and Dorr, 2008; Baldridge and Erk, 2008) make some similar choices about how to blend linguistics and CS concepts, but our desire to integrate deep learning as a primary (but not the sole) focus area guides us towards a different set of assignment topics.

2 Design Principles

This set of assignments was designed after we asked ourselves, *what should a student taking NLP know how to build?* NLP draws on principles from machine learning, statistics, linguistics, algorithms, and more, and we set out to expose students to a range of ideas from these disciplines through the lens of implementation. This choice follows the "text processing first" (Bird, 2008) or "core tools" (Klein, 2005) views of the field, with the idea that students can study undertake additional study of particular topic areas and quickly get up to speed on modeling approaches given the building blocks presented here.

2.1 Covering Model Types

There are far too many NLP tasks and models to cover in a single course. Rather than focus on exposing students to the most important applications, we instead designed these assignments to feature a range of models along the following typological dimensions.

Output space The prediction spaces of models considered here include binary/multiclass (A1, A2), structured (sequence in A3, span in A6), and natural language (sequence of words in A4, executable query in A5). While structured models have fallen out of favor with the advent of neural networks, we view tagging and parsing as fundamental pedagogical tools for getting students to think about linguistic structure and ambiguity, and these are emphasized in our courses.

Modeling framework We cover generative models with categorical distributions (A3), linear feature-based models including logistic regression (A1) and CRFs (A3), and neural networks (A2, A4, A5, A6). These particularly highlight differences in training, optimization, and inference required for these different techniques.

Neural architectures We cover feedforward networks (A2), recurrent neural encoders (A4, A5, A6), seq2seq models (A5), and attention (A5, A6). From these, Transformers (Vaswani et al., 2017) naturally emerge even though they are not explicitly implemented in an assignment.

2.2 Other Desiderata

A major consideration in designing these assignments was to **enable understanding without large-scale computational resources.** Maintaining simplicity and tractability is the major reason we do not feature more exploration of pre-trained models (Devlin et al., 2019). These factors are also why we choose character-level language modeling (rather than word-level) and seq2seq semantic parsing (rather than translation): training large autoregressive models to perform well when output vocabularies are in the tens of thousands requires significant engineering expertise. While we teach students skills like debugging and testing models on simplified settings, we still found it less painful to build our projects around these more tractable tasks where students can iterate quickly.

Another core goal was to allow students to **build systems from the ground-up using simple, understandable code**. We build on PyTorch primitives (Paszke et al., 2019), but otherwise avoid using frameworks like Keras, Huggingface, or AllenNLP. The code is also somewhat "underengineered:" we avoid an overly heavy reliance on Pythonic constructs like list comprehensions or generators as not all students come in with a high level of familiarity with Python.

What's missing **Parsing** is notably absent from these assignments; we judged that both chart parsers and transition-based parsers involved too many engineering details specific to these settings. All of our classes do cover parsing and in some cases have other hands-on components that engage with parsing, but students do not actually build a parser. Instead, sequence models are taken as an example of structured inference, and other classification tasks are used instead of transition systems.

From a system-building perspective, the biggest omissions are **pre-training and Transformers**. These can be explored in the context of final projects, as we describe in the next section.

Finally, our courses integrate additional discussion around **ethics**, with specific discussions surrounding bias in word embeddings (Bolukbasi et al., 2016; Gonen and Goldberg, 2019) and ethical considerations of pre-trained models (Bender et al., 2021), as well as an open-ended discussion surrounding social impact and ethical considerations of NLP, deep learning, and machine learning. These are not formally assessed at present, but we are considering this for future iterations of the course given these topics' importance.

3 Deployment

These assignments have been used in four different versions of an NLP survey course: an upper-level undergraduate course, a masters level course (delivered online), and two PhD-level courses. In the online MS course, these constitute the only assessment. **For courses delivered in a traditional classroom format, we recommend choosing a subset of the assignments and supplementing with additional written assignments testing conceptual understanding.**

Our undergrad courses use A1, A2, A4, and a final project based on A6. We use additional written assignments covering word embedding techniques, syntactic parsing, machine translation, and pre-trained models. Our PhD-level courses use A1, A2, A3, A5, and an independent final project. The assignments also support further "extension" options: for example, in A3, beam search is presented as optional and students can also explore parallel decoding for the CRF or features for NER to work better on German. For the seq2seq model, they could experiment with Transformers or implement constrained decoding to always produce valid logical forms.

Assignment	Eisenstein	Jurafsky + Martin
A1	2, 4	4, 5
A2	3	7
A3	7	8
A4	6	7, 9
A5	12, 18	11, 15
A6	17.5	23.2

Table 2: Book chapters associated with each assignment; gray indicates an imperfect match. Our courses use a combination of Eisenstein, ad hoc lecture notes on certain topics, and academic papers.

We believe that A1 and A2 could be adapted to use in a wide range of courses, but A3-A6 are most appropriate for advanced undergraduates or graduate students.

Syllabus Table 2 pairs these assignments with readings in texts by Jurafsky and Martin (2021) and Eisenstein (2019). See Greg Durrett's course pages for complete sets of readings.

Logistics We typically provide students around 2 weeks per assignment. Their submission either consists of just the code or a code with a brief report, depending on the course format. Students collaborate on assignments through a discussion board on Piazza as well as in person. We have relatively low incidence of students copying code, assessed using Moss over several semesters.

Pain Points Especially on A3, A4, and A5, we come across students who find **debugging** to be a major challenge. In the assignments, we suggest strategies to verify parts of inference code independently of training, as well as simplified tasks to test models on, but some students find it challenging or are unwilling to pursue these avenues. On a similar note, students often **do not have a prior on what the system should do.** It might not raise a red flag that their code takes an hour per epoch, or gets 3% accuracy on the development set, and they end up getting stuck as a result. Understanding what these failures mean is something we emphasize. Finally, students sometimes have (real or perceived) **lack of background** on either coding or the mathematical fundamentals of the course; however, many such students end up doing well in these courses as their first ML/NLP courses.

Acknowledgments

We would like to acknowledge the additional graduate and undergraduate TAs for various offerings of these courses: Christopher Crabtree, Uday Kusupati, Shivangi Mahto, Abhilash Potluri, Shivang Singh, Xi Ye, and Didi Zhou. Our thanks also go out to all of the students who have taken these courses, whose comments and experiences have helped make them stronger. Thanks as well to the anonymous reviewers for their helpful comments.

In the development of these materials, we consulted courses and teaching materials by Emily Bender, Sam Bowman, Chris Dyer, Mohit Iyyer, Vivek Srikumar, and many others. We would also like to thank Jacob Eisenstein, Dan Jurafsky, James H. Martin, and Yoav Goldberg for their helpful textbooks.

References

Jason Baldridge and Katrin Erk. 2008. Teaching computational linguistics to a large, diverse student body: Courses, tools, and interdepartmental interaction. In *Proceedings of the Third Workshop on Issues in Teaching Computational Linguistics*, pages 1–9, Columbus, Ohio. Association for Computational Linguistics.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA. Association for Computing Machinery.

Steven Bird. 2008. Defining a core body of knowledge for the introductory computational linguistics curriculum. In *Proceedings of the Third Workshop on Issues in Teaching Computational Linguistics*, pages 27–35, Columbus, Ohio. Association for Computational Linguistics.

Tolga Bolukbasi, Kai-Wei Chang, James Zou, Venkatesh Saligrama, and Adam Kalai. 2016. Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 4356–4364, Red Hook, NY, USA. Curran Associates Inc.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to Answer Open-Domain Questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jacob Eisenstein. 2019. *Introduction to Natural Language Processing*. MIT Press.

Hila Gonen and Yoav Goldberg. 2019. Lipstick on a pig: Debiasing methods cover up systematic gender biases in word embeddings but do not remove them. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 609–614, Minneapolis, Minnesota. Association for Computational Linguistics.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep Unordered Composition Rivals Syntactic Methods for Text Classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691, Beijing, China. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2016. Data Recombination for Neural Semantic Parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22, Berlin, Germany. Association for Computational Linguistics.

Dan Jurafsky and James H. Martin. 2021. *Speech and Language Processing, 3rd Ed.* Online.

Dan Klein. 2005. A core-tools statistical NLP course. In *Proceedings of the Second ACL Workshop on Effective Tools and Methodologies for Teaching NLP and CL*, pages 23–27, Ann Arbor, Michigan. Association for Computational Linguistics.

Nitin Madnani and Bonnie J. Dorr. 2008. Combining open-source with research to re-engineer a hands-on introductory NLP course. In *Proceedings of the Third Workshop on Issues in Teaching Computational Linguistics*, pages 71–79, Columbus, Ohio. Association for Computational Linguistics.

Tomas Mikolov, M. Karafiát, L. Burget, J. Černocký, and S. Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 79–86. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *arXiv cs.CL 1912.01703*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv cs.CL 1611.01603*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

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 *Advances in Neural Information Processing Systems (NeurIPS)*.

John M. Zelle and Raymond J. Mooney. 1996. Learning to Parse Database Queries Using Inductive Logic Programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2 (AAAI)*.

Tong Zhang and David Johnson. 2003. A Robust Risk Minimization based Named Entity Recognition System. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 204–207.