From Mixed Backgrounds to NLP Skills

Libby Barak and Anna Feldman
Montclair State University
New Jersey, USA

Abstract

Student demand for NLP training now spans linguistics, computer science, data science, and applied fields, producing cohorts with uneven preparation. We report on a four-course curriculum used in an M.S. Computational Linguistics program: an undergraduate on-ramp, a two-course graduate core (classical methods and neural/LLM methods), and a rotating special-topics seminar. We describe the role of each course, the bridging strategy that keeps the core sequence focused, and assessment patterns that emphasize error analysis, experimental reasoning, and reproducible practice. The goal is a set of reusable curricular design patterns for mixed-background programs facing rapid topic turnover in NLP.

1 Motivation and Challenges

Transformer-based language modeling and widely available generative AI tools have broadened demand for NLP training beyond traditional computer science audiences (Vaswani et al., 2017; Devlin et al., 2019). Increasingly, students from linguistics, psychology, education, health, and other applied fields seek practical NLP training to support research or professional goals. However, many of these students lack the programming, mathematical, or algorithmic background assumed by standard NLP courses. At the same time, undergraduate and graduate students with stronger technical preparation often seek deeper specialization, creating heterogeneity within a single classroom.

Prior work on teaching NLP and computational linguistics has proposed curricula that balance accessibility with technical rigor, particularly in mixed-background classrooms (Baldridge and Erk, 2008; Agarwal, 2013; Manning et al., 2021; Kennington, 2021). Approaches range from courses that assume substantial computational preparation (Kennington, 2021) to low-code or non-computational offerings (Baldridge and Erk, 2008). Smaller programs also face enrollment constraints that can make multiple specialized tracks infeasible (Baldridge and Erk, 2008). One classic strategy is to form mixed teams that combine linguistic and computational strengths for project work (Manning and Schütze, 1999). We take a different tack: we separate baseline skill-building from NLP method instruction. At the undergraduate level, we create multiple tracks in a shared classroom that build upon student knowledge through assignment design and use of code generation. At the graduate level, we use a layered course sequence, so students can progress from their existing strengths toward independent, reproducible NLP practice.

Many M.S.-level CL/NLP curricula address mixed preparation either by (i) assuming strong CS prerequisites and offering optional linguistics support, or (ii) offering a single introductory NLP course that must serve as both a methods course and a programming ramp. Our curriculum separates these roles: an on-ramp course emphasizes conceptual understanding and error analysis with controlled AI-assisted coding, while the graduate core targets reproducible experimentation and ACL-style writing. We further isolate fast-changing content into CL-II and a rotating seminar to reduce curricular drift in the core.

2 Program Context and Curricular Structure

The courses described in this paper are part of an M.S. program in Computational Linguistics at a public research university and also serve undergraduate students through an on-ramp course. The M.S. program admits students from any undergraduate major, resulting in substantial variation in prior preparation. In the early years of the program, we offered a single core NLP course to a mixed class of undergraduate and graduate students, similar to (Baldridge and Erk, 2008). As graduate

Proceedings of the Seventh Workshop on Teaching Natural Language Processing (TeachNLP 2026), pages 59–68
March 29, 2026 ©2026 Association for Computational Linguistics

enrollments grew (in computational linguistics and data science), we split the curriculum into multiple layers with distinct instructional roles. We do not claim this curriculum is universal; we focus on transferable design patterns from a mixed-background program.

Contributions We describe a four-course NLP curriculum for mixed-background cohorts and identify reusable design patterns. Specifically, we present:

1. a layered course sequence with distinct roles that supports uneven preparation;
2. assessment and project designs that emphasize error analysis, experimental reasoning, and reproducibility;
3. a curriculum structure that places fast-changing topics in a rotating seminar.

3 Bridging Skills Across Linguistics, Mathematics, and Computer Science

The on-ramp has no formal CS or mathematics prerequisites but assumes either introductory linguistics or basic Python. Students lacking both may enroll with instructor approval. We scaffold linguistic analysis, Python practice, basic statistics, and linear-algebra intuitions as we progress in the sequence. As described in Section 4, we use controlled, documented AI-assisted coding to reduce time spent on boilerplate implementation and to keep the on-ramp focused on NLP concepts, analysis, and debugging. We still teach essential programming skills (reading code, debugging generated code, testing, and error diagnosis, alongside using open-source packages and using published NLP modules for reproduction). This choice reflects student feedback that a domain course can stall if too much time is diverted to introductory programming.

At the graduate-level, introductory foundation courses provide initial normalization. Rather than embedding remedial instruction inside NLP courses, we rely on explicit bridging courses in linguistics, mathematics, and computer science. This keeps the NLP sequence focused while giving students clear paths to fill gaps. First, *Language and Linguistics* supports students without prior linguistic training, while *Python Programming* introduces basic programming skills for students without a computational background. These courses allow subsequent NLP instruction to proceed without assuming uniform undergraduate preparation.

In addition to in-class assignments and labs, students in the M.S. program acquire the necessary foundational knowledge through the following set of required courses. Quantitative and algorithmic competence is reinforced through required coursework in applied mathematics and computer science. *Applied Mathematics for Computational Science* introduces linear algebra, probability, and optimization concepts that later underpin vector representations, statistical inference, and neural models. These connections reflect long-standing foundations of statistical NLP, where linguistic modeling, probability, and linear algebra are tightly coupled (Manning and Schütze, 1999; Manning et al., 2008). *Data Structures with Python* emphasizes algorithmic thinking and software practices relevant to corpus processing and feature extraction, while *Machine Learning* provides a formal introduction to supervised and unsupervised modeling techniques used throughout the NLP curriculum. These courses are not treated as stand-alone prerequisites: later NLP assignments explicitly reuse their concepts (e.g., vector similarity in embeddings; complexity and data handling in corpus work). These bridging courses are program requirements outside of the four NLP anchor courses; we describe them here to illustrate the NLP sequence can assume.

4 Learning Objectives and Assessment Alignment

Learning objectives progress from introductory competence to independent research. We summarize the four courses included in our programs in Table 1. At the undergraduate level, the primary goals are technical confidence, familiarity with core NLP concepts, and the ability to interpret model behavior rather than to optimize performance. At the graduate level, objectives expand to include experimental design, reproducibility, critical evaluation of models, and formulation of research questions.

These objectives are aligned explicitly with assessment formats, with an emphasis on applied and project-based work at all levels. In the introductory course, learning is assessed through guided notebooks, short written analyses, and guided projects that emphasize interpretation and error analysis.

In the first graduate core course *Computational Linguistics I* (CL-I), assessment is more structured and exam-based, combining problem sets and lab assignments with a midterm and final exam to en-

Course	Pedagogical role and typical assessment
On-ramp	Guided notebooks; error analysis; small project
CL-I	Classical NLP core; problem sets + exams
CL-II	Neural/LLM methods; mini-projects + short reports
Seminar	Reading responses; research project

Table 1: Four anchor courses and their roles in the layered curriculum.

sure mastery of foundational concepts and methods. In the subsequent course, *Computational Linguistics II* (CL-II), assessments shift toward more open-ended mini-projects and short technical reports that require students to justify modeling choices, evaluation metrics, and experimental results.

In the seminar, *Special Topics in NLP*, assessment is centered on research-oriented writing and staged project development. Weekly reading responses assess critical engagement, and staged project milestones assess literature synthesis, planning, implementation, and analysis. Across all courses, grading criteria prioritize clarity of reasoning, transparency of methods, and reflection on model limitations over absolute performance scores, consistent with recent work emphasizing behavioral testing and interpretability in NLP (Ribeiro et al., 2020; Belinkov and Glass, 2019).

In this curriculum, we use "interpretability" primarily in the behavioral sense: targeted tests, error analysis, counterfactual/perturbation analysis, and controlled comparisons across datasets and prompts. We do not aim for deep mechanistic interpretability of model internals in the core sequence, though some seminar rotations may cover internal analysis tools.

AI-assisted coding policy (all courses). Students may use generative AI tools to draft code and explanations, but they must document prompts and iterations, cite AI assistance in submissions, and verify correctness with tests and sanity checks. Grading emphasizes (i) experimental reasoning, (ii) error analysis, and (iii) reproducibility; unverified or opaque AI-generated outputs receive little credit. When assessments require individual understanding (e.g., exams or oral checkoffs), AI tools are not permitted.

5 An Undergraduate On-Ramp to NLP

The upper-level undergraduate course introduces NLP to students with little or no prior programming experience and often serves as a first point of contact with computational linguistics. Accordingly, the course balances linguistic analysis with practical computing: non-linguistics students need linguistic structure, while non-computing students need executable workflows. Thus, the main goal is to build technical confidence without losing sight of linguistic insight.

The syllabus advances through NLP topics aligned with increasing levels of linguistic and modeling abstraction. The course alternates lectures and lab sessions; each topic is introduced in lecture and practiced in guided notebooks. Students begin with hands-on Python labs covering tokenization, lemmatization, frequency analysis, and corpus queries, progressing to POS tagging, and simple text classification. We demonstrate how to use generative AI tools to draft code, run it, and evaluate outputs, while keeping students responsible for verification and interpretation. Students with stronger computing backgrounds are pushed toward linguistically motivated analysis (e.g., how tokenization and tagging decisions affect downstream errors). Students with stronger linguistics backgrounds leverage their prior knowledge of structure and meaning to formulate hypotheses and test them with small computational experiments. All students complete at several manual walks-through of a core algorithm (e.g., BPE tokenization) on a small dataset to build an explicit mental model of how the algorithm behaves. The course covers Naive Bayes, logistic regression, and neural-network fundamentals, and ends with an introduction to LLMs.

In the first weeks of the semester, assignments emphasize interpretation and evaluation rather than raw implementation, for example, through error analysis of tagging outputs or comparison of modeling assumptions across datasets. We offer multiple lab paths within the same assignment so students reach the same learning goals. Students with computational background complete guided model analysis to introduce core linguistic concepts such as parsing and part-of-speech tagging, while students from linguistics and humanities backgrounds are guided through short, targeted code-generation exercises to learn how to prompt generative AI to generate starter code for standard NLP workflows. For example, one lab pairs a controlled code-generation workflow (with verification and tests) with a manual annotation baseline; students complete one path and then compare outputs, errors, and explanations across methods. Using generative AI in this controlled way emphasizes inspection

and critique rather than automation, and requires behavioral tests and targeted checks rather than trusting fluent outputs (Ribeiro et al., 2020).

The final project invites students to apply the learned concepts to a topic of their choice. Students are provided with three notebooks applying NLP models to a sample task, (1) Sentiment analysis with Logistic Regression using tf-idf features, (2) evaluating bias in job application data using word2vec similarity (Mikolov et al., 2013), and (3) fake news identification using LLM prompting. Students can familiarize themselves with each option using the sample file. Students can choose between applying one of the options to new data of their choice, comparing the performance of multiple options on data of their choice, or extending the seed code to evaluate additional models or features. The project structure guides students through an exploratory research process:

1. Reading brief: Students analyze one paper chosen from an instructor-provided list, identifying its goal, method, key results, and limitations, and posing two questions (e.g., linguistic change (Hamilton et al., 2016), bias (Levy et al., 2024), second-language learning (Barak et al., 2020)).
2. Data plan: students submit a short description of their project goal and proposed dataset, with lab support for data sourcing and preprocessing.
3. Work-in-progress poster: students present project goal, data, method, and preliminary or expected results, followed by guided class discussion.
4. Final abstract: students submit a one-page report emphasizing evaluation, analysis, and limitations.

6 Layered Graduate Training

Computational Linguistics I (CL-I) and *Computational Linguistics II* (CL-II) (see Table 1) form the technical core of graduate-level training. *Computational Linguistics I* enrolls students from Computational Linguistics, Computer Science, and Data Science programs, leading to sharp asymmetries in prior experience. The course emphasizes shared foundations in NLP, including classical models, evaluation practices, and reproducible experimentation. The course draws on widely used introductory texts in NLP and computational linguistics (Jurafsky and Martin, 2025; Bird et al., 2009; Eisenstein, 2019), emphasizing hands-on exploration and interpretive analysis rather than mathematical formalism.

The progression of assignments balances computational skills while keeping the focus on core computational linguistics concepts. Thus, the labs over the first weeks use guided notebooks in which students complete small parts of missing code, modify sections using detailed instructions, or execute provided code to analyze the behavior of the model. For example, students are provided with an implementation of a subword unit tokenization algorithm that learns tokens from text of varying length. Instructions guide the students to modify the code to control which text is used to learn the tokens, and evaluate the relationship between text length, vocabulary size, and learning outcome.

As described in Section 3, graduate students complete bridging coursework in mathematics, programming, and data structures. Even with this preparation, students report that CL-I is often their first structured exposure to cloud-based computational environments and collaborative version control. Students also report that manually stepping through algorithms on simplified data helps them connect linguistic properties to model behavior, which in turn makes later neural and LLM material easier to interpret. This motivated our decision to keep CL-I focused on stable foundations while using CL-II and the rotating seminar to cover emerging methods. CL-II extends this foundation to modern neural architectures, including transformers, pretrained language models, multilingual modeling, interpretability, and ethical considerations. In this course, students complete mini-projects using pretrained models and synthesize their findings in short ACL-style research reports.

While CL-I follows a more rigid syllabus that aims to cover core topics in the field, CL-II remains more flexible. Each year, the course is updated to cover current topics such as retrieval-augmented generation (RAG) and other emerging methods for model integration.

7 Special Topics in NLP

Special Topics in NLP is a rotating graduate seminar designed to develop research literacy and independent problem formulation in Natural Language Processing. Unlike the core graduate sequence, which emphasizes methodological foundations and model implementation, this course centers on close

reading of recent research, critical discussion, and staged project development. The seminar serves two purposes: it gives students structured practice reading and critiquing recent NLP research, and it provides an on-ramp to independent projects that can later become capstone work. Because each rotation is led by a domain specialist, the course also creates a natural pathway for motivated students to join ongoing research collaborations. Weekly meetings are organized around recent conference and journal papers, selected to reflect current directions in NLP. Each week, students are required to submit written questions and comments on the readings prior to class. These submissions are used to structure in-class discussion and encourage analytical engagement rather than summary. This component constitutes a regular, low-stakes writing requirement and accounts for 10% of the final grade.

The course is designed to give instructors flexibility in how they structure each rotation. At the same time, students are guaranteed a consistent set of expectations and an evaluation scheme, with 45% of the grade remaining flexible across rotations. For example, students may be asked to use publicly available code to replicate published results. Alternatively, each student may be assigned a journal paper and asked to create a class guiding peers through the topic. While the grading scheme remains constant, this portion of the grade allows each instructor to construct curricula that aligns with the target skills.

A substantial portion of the course is devoted to an individual or small-group final project, which accounts for 45% of the total grade. The project is structured as a staged research process with fixed milestones, including a literature review, project proposal, two intermediate reports, and a final paper and presentation. The literature review requires students to synthesize multiple recent papers, situate their project within existing work, and articulate a concrete research question or implementation goal.

Projects may take the form of exploratory research studies or implementation-focused replications. In both cases, students are expected to define a task, implement or adapt an NLP method, evaluate outcomes, and reflect on limitations and future directions. Final submissions follow ACL formatting guidelines and include code, data, and documentation in a public repository, reinforcing reproducibility and professional research practices.

The seminar topics vary from year to year and have included neural network models, transformer-based architectures, prompting and instruction tuning, machine translation, natural language generation, model interpretability, and the social impact of NLP systems. By isolating fast-evolving topics in a dedicated seminar, this course enables the curriculum to remain responsive to developments in the field without destabilizing the technical core courses.

To accommodate different research trajectories and to reflect the evolving role of NLP across disciplines, the *Special Topics in NLP* course has been offered with different thematic emphases. While the course mechanics (weekly readings, discussion-based seminars, and a staged final project) remain constant, the intellectual focus and types of research questions vary across offerings. Below we outline three representative variants.

7.1 Variant 1: Computational Modeling of Human Cognition

This variant focuses on computational modeling of linguistic and cognitive phenomena. This broad topic highlights interdisciplinary research and challenges students to an understanding of experimental work in psycholinguistics and social sciences. On the one hand, students learn how computational designs align with assumptions on learner characteristics, e.g., can the learner break sound into words? does the learner have perfect memory? how is meaning represented by the learner? On the other hand, students learn to map the experimental psycholinguistic design to a computational model. For example, various studies on the acquisition of word learning may differ in the stimulus, participants, and evaluation, which should be mapped to design choices of training data, computational model, and test data.

The flexible portion of the grade in this variant centers on three guided replication experiments. These assignments enhance theoretical discussions on model alignment with human cognition. Students compare Bayesian and data-driven learners, vary training data selection, and test synthetic data generation.

Projects in this class are often exploratory, since obtaining data to replicate psycholinguistic experiments is often constrained by practical and logistical limitations. For example, many psycholinguistic publications do not include the full stimulus used to test human subjects, but rather focus

on qualitative analysis. Yet, students are able to pursue a wide range of topic inspired by the assigned readings. Some representative final projects include:

- AI generation of child-directed speech - comparing child-directed speech to AI generated text using the size of vocabulary as a measure of learning trajectory by prompting LLMs to generate data similar to child-directed speech.
- Grammatical morphology acquisition - modeling the acquisition process of number system in languages with dual and plural marking such as Slovenian.
- Cross-lingual semantic mapping - modeling the relationship between representation and comprehension by comparing semantic representation of words across languages.

7.2 Variant 2: Computational Social Science

In some offerings, *Special Topics in NLP* adopts a thematic focus on computational social science, using NLP methods to study socially grounded language phenomena such as euphemism, stance, framing, bias, and power relations in text. Within this variant, the seminar foregrounds language as social action and treats NLP models both as analytical tools and as objects of critical examination.

Readings in these offerings are drawn from NLP, computational linguistics, and adjacent fields such as sociolinguistics and political science, and emphasize methodological choices, data bias, annotation decisions, and evaluation challenges in socially situated tasks. Class discussions focus on how modeling assumptions interact with social meaning, representation, and ethical considerations, rather than on model performance alone.

Typical projects involve corpus construction or adaptation, classification or sequence labeling for social variables, analysis of model errors and decision boundaries, and reflection on fairness, interpretability, or downstream impact. Students are encouraged to treat replication, careful error analysis, and ablation-style studies as valid contributions, particularly when working with sensitive or noisy data.

This seminar emphasis allows students to apply technical NLP methods to socially grounded research questions while reinforcing best practices in experimental design, evaluation, and reproducibility.

Example final projects have included:

- identification and categorization of euphemistic expressions in online discussion forums, with analysis of how model predictions vary across contexts;
- stance detection in political or public health discourse, comparing supervised classifiers and prompt-based methods;
- evaluation of social bias in pretrained language models through controlled prompt and corpus-based experiments;
- replication and ablation of recently published NLP models to assess robustness under domain shift;
- construction and annotation of small, domain-specific corpora followed by baseline modeling and error analysis.

7.3 Variant 3: Figurative Language Processing and Pragmatic Competence

In this variant, *Special Topics in NLP* is organized around figurative language as a case study in pragmatic competence and computational social meaning. The seminar uses phenomena such as euphemism, metaphor, indirectness, and pragmatic inference to examine how meaning is constructed, negotiated, and obscured in real-world text, and how current NLP models succeed or fail in capturing these aspects of language use.

Readings span NLP, computational linguistics, and linguistics, including work on figurative language processing, stance and framing, social meaning, bias, and interpretability. Rather than treating figurative language as a narrowly defined task, the seminar frames it as a stress test for NLP models, revealing assumptions about literal meaning, contextual reasoning, and commonsense knowledge. Discussion emphasizes annotation practices, dataset design, evaluation beyond accuracy, and the relationship between linguistic theory and computational representations.

Example final projects in this variant have included:

- detection and categorization of euphemistic expressions in online health or political forums, with analysis of model sensitivity to context;
- metaphor identification in news or social media text, comparing supervised, embedding-based, and prompt-based approaches;

- analysis of indirect requests and implicatures in dialogue data, focusing on pragmatic inference beyond surface meaning;
- controlled error analysis showing where models interpret figurative language literally or conflate distinct pragmatic functions;
- replication of recent figurative language studies with alternative datasets or evaluation metrics.

7.4 Illustrative Assignments and Use of LLMs

To ground the curriculum design in concrete practice, we briefly describe representative assignments that illustrate how learning objectives are operationalized across levels and how large language models are incorporated pedagogically.

Undergraduate error analysis with pretrained models. In the undergraduate on-ramp course, students are given the output of a pretrained part-of-speech tagger or classifier applied to a small corpus. Rather than training models from scratch, students analyze systematic errors, relate them to linguistic properties of the data, and propose hypotheses about model behavior. Submissions consist of a short notebook and written analysis; grading emphasizes interpretive reasoning and clarity rather than performance.

Graduate comparison of classical and LLM-based methods. In the graduate core sequence, a typical assignment asks students to compare a classical machine-learning approach (e.g., feature-based classification) with a pretrained language model on the same task. Students must justify preprocessing choices, evaluation metrics, and experimental controls, and reflect on differences in error profiles. LLMs are treated explicitly as objects of analysis rather than as black-box solutions. Comparing feature-based models with pretrained transformers highlights trade-offs between inductive bias, data efficiency, and interpretability (Goldberg, 2017; Devlin et al., 2019).

Because LLM prompting can produce a working baseline quickly, we explicitly teach classical methods as instruments for reasoning: they make feature/assumption choices visible, support small-data settings, and provide interpretable baselines for diagnosing failures. LLM-based solutions are taught alongside them, but students must compare error profiles, stability, and reproducibility rather than treating prompt outputs as the endpoint.

Seminar projects using LLMs as analytical tools. In *Special Topics in NLP*, students may use LLMs for tasks such as exploratory annotation, prompt-based baselines, or hypothesis generation. Assignments require students to document prompts, assess stability across runs, and critically evaluate limitations. The focus remains on methodological transparency, reproducibility, and the relationship between model behavior and linguistic or social phenomena.

Across levels, AI-assisted work must be documented (e.g., prompts, revisions, and verification steps), and grading emphasizes reasoning, evaluation, and reproducibility rather than output fluency.

8 Skill Outcomes and Reproducible Practice

Across courses, students learn to construct and explore corpora, preprocess and analyze linguistic data, build classical machine-learning models for text, and work with pretrained language models through fine-tuning, prompting, and systematic evaluation. Emphasis is placed on understanding model behavior through targeted experiments and careful error analysis.

Reproducibility is treated as a core learning objective rather than an optional add-on. All assignments and projects require students to submit code and documentation in a public or course-hosted GitHub repository, including instructions for data acquisition, environment configuration, and experiment replication. Students are expected to version-control their work, use fixed random seeds where applicable, and articulate the assumptions behind preprocessing, hyperparameter choices, and evaluation metrics.

Tool use is framed not as tool mastery but as a means for transparent experimentation. Students work with open-source NLP ecosystems (e.g., spaCy, scikit-learn, Hugging Face transformers) to focus on modeling decisions, data design, and interpretation rather than low-level engineering. This emphasis prepares students to critique existing models, adapt methods to new domains, and conduct small but meaningful empirical studies with clear methodological justification. In the on-ramp, we teach prompting as a debugging aid: students must (i) verify AI-generated code with tests and sanity checks and (ii) inspect whether the solution appropriately uses standard libraries (e.g., spaCy, scikit-learn) rather than reinventing them.

9 Discussion

We presented a layered NLP curriculum spanning an undergraduate on-ramp, a two-course graduate core, and a rotating seminar. Our main design choice is to separate baseline skill-building from NLP method instruction. The undergraduate on-ramp emphasizes conceptual understanding, error analysis, and reproducible workflows while using controlled AI-assisted coding to reduce time spent on boilerplate implementation. The graduate core then builds toward open-ended experimentation and ACL-style writing, while a rotating seminar absorbs fast-moving topics. This structure is intended to help mixed-background programs maintain both accessibility and depth without destabilizing the core syllabus each year. Although grounded in one program, the approach targets common constraints in NLP teaching: uneven preparation and rapid model turnover. The course roles and assessment patterns are intended as reusable templates.

Our proposal stems from an internal review process of previous course offerings, student feedback, and open discussion with graduate students and faculty. Our observations from previous offerings of the undergraduate course indicated that integrating programming instruction into a domain-specific course can create an imbalance, with disproportionate time devoted to coding at the expense of broader skill development. Moreover, our observations from graduate students showed that NLP modeling offers a good introduction to programming tools that are valuable to professionals in Computer Science, Data Science, and Information technology, but leave the students wanting more nuanced experience with advanced NLP topics.

Peer M.S. programs in computational linguistics commonly include (i) prerequisite leveling courses in linguistics and programming, (ii) one or two core NLP/CL methods courses, and (iii) a capstone or thesis. Our contribution is not the existence of these components, but an explicit division of responsibilities across four anchor NLP courses and an assessment strategy that treats error analysis and reproducibility as first-class outcomes across the sequence.

Students without prior programming experience struggle most with debugging and abstraction, while technically advanced students may initially underestimate the importance of linguistic analysis and evaluation. Explicitly separating conceptual understanding from implementation early in the curriculum mitigates both issues. Students with computational background are often familiar with concepts of model evaluation such as correlation tests and accuracy, but lack knowledge in qualitative analysis and linguistic theory. The guided analysis over the introductory modules, such as tokenization and part-of-speech tagging, allow students to draw the connection between model design, training data, and parametrization. Students with linguistics majors report that the course increased their confidence using NLP and programming in other classes and in applied projects. For example, students report using course skills to automate text-heavy workflows (e.g., resume screening or document triage) and to analyze domain datasets from their home disciplines. Several students from sociology and psychology backgrounds applied NLP methods to text from mental-health and emotional-support settings. On the other hand, undergraduate students from data science major felt that this course offered a more accessible way to learn model analysis with deeper understanding of data and input structure. Several students continued to work in research labs (e.g., euphemism annotation in Spanish, and applying LLMs to Native American Languages). In the graduate level, a student with cognitive science background secured a computational NLP internship using the skills learned in CL-I. Several students published their work in ACL workshops based on projects initiated in CL-II and Special Topics in NLP (Ducret et al., 2020; Aharodnik et al., 2013; Ng et al., 2020; Shode et al., 2023; Lee et al., 2024; Mulholland and Quinn, 2013; Poh et al., 2025).

A second challenge is curricular drift driven by rapid advances in models and tooling. Concentrating fast-changing material in a rotating seminar allows core courses to remain stable while still exposing students to current research. We addressed this challenge by creating CL-II with a more flexible syllabus that is continuously updated with recent publications and advances. In addition, Special Topics in NLP covers a different topic yearly to provide students with the oportunity to engage with experts in NLP.

Finally, reliance on pretrained models lowers barriers to entry but risks obscuring modeling assumptions; structured error analysis assignments partially address this trade-off. A related instructional challenge is managing workload and assessment consistency in courses with open-ended assignments. Clear rubrics focused on reasoning,

documentation, and evaluation, rather than raw performance, help control grading complexity while maintaining fairness across students with different technical starting points. Conversely, approaches that rely on uniform prerequisites, performance-only grading, or rapidly changing tool-specific content tend to amplify background differences and reduce opportunities for meaningful analysis.

10 Limitations

This paper does not report controlled evaluations of learning outcomes or cross-institutional comparisons. Our goal is to document a curriculum architecture and assignment/assessment patterns that have been iteratively refined across multiple offerings. A natural next step is lightweight, comparable outcome measurement (e.g., rubric-based artifact evaluation or longitudinal tracking across cohorts).

We do not analyze infrastructure constraints (compute, storage, and software support) in depth. In practice, we mitigate these constraints by alternating the four courses across fall and spring, constraining dataset sizes and model scales in projects, and teaching cloud-based environments and collaborative version control. Future work should document cost-aware teaching designs that make compute limitations explicit.

References

Apoorv Agarwal. 2013. Teaching the basics of NLP and ML in an introductory course to information science. In *Proceedings of the Fourth Workshop on Teaching NLP and CL*, pages 77–84, Sofia, Bulgaria. Association for Computational Linguistics.

Katsiaryna Aharodnik, Marco Chang, Anna Feldman, and Jirka Hana. 2013. Automatic identification of learners' language background based on their writing in Czech. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1428–1436, Nagoya, Japan. Asian Federation of Natural Language Processing.

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.

Libby Barak, Scott Cheng-Hsin Yang, Chirag Rank, and Patrick Shafto. 2020. Modeling second language preposition learning. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 42.

Yonatan Belinkov and James Glass. 2019. Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL*.

Martina Ducret, Lauren Kruse, Carlos Martinez, Anna Feldman, and Jing Peng. 2020. Linguistic features in automatic sarcasm detection. In *Proceedings of the Seventh Italian Conference on Computational Linguistics (CLiC-it 2020)*, pages 132–138, Bologna, Italy. CEUR Workshop Proceedings.

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

Yoav Goldberg. 2017. *Neural Network Methods for Natural Language Processing*. Morgan & Claypool.

William L Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Cultural shift or linguistic drift? comparing two computational measures of semantic change. In *Proceedings of the conference on empirical methods in natural language processing. Conference on empirical methods in natural language processing*, volume 2016, page 2116.

Dan Jurafsky and James H. Martin. 2025. *Speech and Language Processing*, 3 edition. Draft.

Casey Kennington. 2021. Natural language processing for computer scientists and data scientists at a large state university. In *Proceedings of the Fifth Workshop on Teaching NLP*, pages 115–124.

Patrick Lee, Alain Chirino Trujillo, Diana Cuevas Plancarte, Olumide Ojo, Xinyi Liu, Iyanuoluwa Shode, Yuan Zhao, Anna Feldman, and Jing Peng. 2024. MEDs for PETs: Multilingual euphemism disambiguation for potentially euphemistic terms. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 875–881, St. Julian's, Malta. Association for Computational Linguistics.

Sharon Levy, Tahilin Sanchez Karver, William D Adler, Michelle R Kaufman, and Mark Dredze. 2024. Evaluating biases in context-dependent health questions. *arXiv preprint arXiv:2403.04858*.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

Emma Manning, Nathan Schneider, and Amir Zeldes. 2021. A balanced and broadly targeted computational linguistics curriculum. In *Proceedings of the Fifth Workshop on Teaching NLP*, pages 65–69.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

Matthew Mulholland and Joanne Quinn. 2013. Suicidal tendencies: The automatic classification of suicidal and non-suicidal lyricists using NLP. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 680–684, Nagoya, Japan. Asian Federation of Natural Language Processing.

Kei Yin Ng, Anna Feldman, and Jing Peng. 2020. Linguistic fingerprints of internet censorship: The case of sina weibo. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(01):446–453.

Whitney Poh, Michael Tombolini, and Libby Barak. 2025. What did you say? generating child-directed speech questions to train LLMs. In *Proceedings of the First BabyLM Workshop*, pages 237–245, Suzhou, China. Association for Computational Linguistics.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of nlp models with CHECKLIST. In *Proceedings of ACL*.

Iyanuoluwa Shode, David Ifeoluwa Adelani, JIng Peng, and Anna Feldman. 2023. NollySenti: Leveraging transfer learning and machine translation for Nigerian movie sentiment classification. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 986–998, Toronto, Canada. Association for Computational Linguistics.

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 *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010.