Package 2: AI and Natural Language Processing Sequence

Instructor: Utku Turk (utkuturk@umd.edu) **Format:** Quarter System (10 weeks per course) **Level Progression:** Mixed (Undergrad/Grad) → Advanced Graduate

---

Package Vision

This two-course sequence bridges machine learning, NLP, and linguistic theory. Students learn to build neural models **and** use them as scientific instruments to investigate linguistic structure. Unlike typical NLP courses prioritizing engineering performance, this sequence maintains a linguistic focus: every model is interrogated for what it reveals about language. Students build models from scratch, probe their representations, and critically evaluate claims about AI capabilities. The sequence prepares students for interdisciplinary research in computational linguistics, whether in academia or industry.

Course Summaries

Course 1: Machine Learning for Language (Mixed Level, Fall Quarter)

Prerequisites: None. No prior programming or math experience required.

This course introduces neural networks and their applications to language, balancing technical foundations with linguistic questions. Students build neural networks from scratch—implementing backpropagation by hand, training word embeddings, constructing RNNs/LSTMs, and finally working with transformers and LLMs. But critically, every model is interrogated through three questions: (1) What can it do? (2) What can't it do? (3) What does that tell us about language? Students design probing experiments to test whether models capture syntactic dependencies, semantic relations, or merely surface heuristics. Python programming taught from scratch through hands-on labs in Google Colab.

Key outputs: 4 homework assignments (coding + concepts), midterm classifier proposal, final research proposal (8-10 pages) investigating a linguistic question with neural networks.

Course 2: Advanced NLP & AI for Linguistics (Graduate Level, Winter Quarter)

Prerequisites: Background in neural networks (Course 1 or equivalent)

Neural networks are often treated as black boxes. In this advanced seminar, we open them up. Students learn cutting-edge interpretability methods: (1) **probing classifiers** to test for linguistic knowledge, (2) **attention analysis** to understand syntactic dependency encoding, and (3) **causal intervention** to establish causal relationships between model components and behavior. The course runs as a research workshop—students read 2024-2025 papers, lead critical discussions, implement state-of-the-art methods, and develop original research proposals. By the end, students can design rigorous interpretability experiments and contribute to debates about what neural models actually learn.

Key outputs: 2 paper presentations, 3 method implementations (probing, attention, causality/geometry), final research paper (10-12 pages in ACL format) with code.

Learning Philosophy

This sequence embodies three core principles. **First, linguistics first, engineering second:** these courses prioritize linguistic questions over benchmark performance. **Second, critical AI literacy:** students question claims about AI capabilities—not "can LLMs do syntax?" but "what syntactic patterns do they capture?" **Third, hands-on implementation:** students implement backpropagation by hand, code trans-

formers from scratch, and run causal interventions. Students completing both courses can build modern NLP systems, critically evaluate AI capabilities, and conduct interpretability research.

Progression: Course 1 teaches how to build and train neural models → Course 2 teaches how to probe and interpret them. Skills build cumulatively: Python/PyTorch (both courses), transformers (introduced in Course 1, analyzed in Course 2), critical evaluation (Course 1 foundations, Course 2 advanced methods).

Machine Learning for Language

Neural Networks, Linguistic Structure, and NLP Applications

LING 3XXX – Fall Quarter 2026

Time: TBD (meets 4 hours/week: 2 sessions of 2 hours each)

Location: TBD

Instructor: Utku Turk (utkuturk@umd.edu)

Office hours: By appointment

Course site: Canvas

Prerequisites: None. No prior programming or math experience required.

Format: Quarter system (10 weeks)

---

1. Course Description

How do neural networks learn to process language? Can they acquire grammatical knowledge from text alone? What can machine learning tell us about linguistic structure?

This course introduces neural networks and their applications to language, balancing technical foundations with linguistic questions. You will learn how to build text classifiers, train language models, and implement modern NLP systems. But you will also investigate what these models learn: Do they capture syntactic dependencies? Can they generalize across languages? How do their representations compare to human linguistic knowledge? I am excited to explore the intersection of these two fields with you as we build the technology of the future.

By the end of the quarter, you will be able to implement neural models in Python, design experiments to probe what they've learned, and critically evaluate claims about AI and language. This is a hands-on course that combines practical ML skills with linguistic analysis.

No prior programming experience required—we build skills from scratch.

---

2. Learning Objectives

Upon successful completion of this course, students will be able to:

1. Implement neural networks for text classification and language modeling in Python/PyTorch.
2. Understand word embeddings and how meaning is represented in vector space.
3. Build and train RNNs, LSTMs, and transformer models.
4. Design probing experiments to test what models learn about linguistic structure.
5. Analyze attention patterns and model representations.
6. Evaluate whether models learn syntax, semantics, or surface heuristics.
7. Apply neural models to practical NLP tasks (sentiment, translation, generation).
8. Critically evaluate claims about AI capabilities and limitations.

---

3. Course Structure

Weekly Rhythm

Component	What it looks like
Lecture	Concepts, math intuition, and how things work (Session 1).
Coding lab	Hands-on Python work: implementing models, debugging (Session 2).
Homework	Four problem sets with coding + conceptual questions.
Model of the Week	Each week, we spotlight one influential model or paper. You'll interact with it hands-on—not just read about it, but run it, break it, and understand what makes it tick.

The Three Questions

Every model we study will be interrogated through three lenses:

1. **What can it do?** (Capabilities: tasks it solves, benchmarks it passes)
2. **What can't it do?** (Limitations: where it fails, what it doesn't learn)
3. **What does that tell us about language?** (Implications: what this reveals about linguistic structure and cognition)

This isn't just a coding class or just a linguistics class—it's both. You'll build the models AND use them as scientific instruments to investigate language.

Your Learning Arc

- **Weeks 1-3**: Build your own neural networks from scratch (no magic libraries—you'll implement backprop by hand!)
- **Weeks 4-6**: Explore what models learn about language structure (syntax, semantics, pragmatics)
- **Weeks 7-9**: Work with state-of-the-art transformers and LLMs
- **Week 10**: Critical evaluation—when do models succeed for the wrong reasons?

Tools We Use

- **Python:** primary programming language.
- **PyTorch:** neural network framework.
- **Google Colab:** cloud-based coding environment.
- **Hugging Face:** pretrained models and datasets.

---

4. Course Requirements

4.1 Grading

Item	%	What counts
Participation	15	Active engagement in labs.
Homework (4x)	48	Coding + concepts (12% each).
Midterm presentation	7	Classifier proposal (Week 5).
Final project	30	Detailed research proposal (8-10 pages).

4.2 Projects

- **Midterm (Week 5):** Propose a neural classifier for a linguistic phenomenon (e.g., modal reading, genericity). 2-page proposal + 5-minute presentation.

- **Final (Finals week):** Detailed research proposal investigating a linguistic question using neural networks. 8-10 page proposal in ACL format + presentation.

---

5. Course Schedule (10 Weeks)

Wk	Topic	Readings (Selected)	Lab / Due
1a	Intro; NNs and Language	Newell (1973); Marr (1982)	Python basics I
1b	Representing words	Jurafsky & Martin Ch. 6	Python basics II
2a	Perceptrons; Gradient Descent	Goodfellow et al. Ch. 6	Perceptron from scratch
2b	MLP; Backprop	Goodfellow et al. Ch. 6; Olah	PyTorch basics
3a	Word Embeddings	Jurafsky & Martin Ch. 6	Training embeddings
3b	Probing Structure	Ettinger (2020)	**Due:** HW1; Probing lab
4a	RNNs; Language Modeling	Jurafsky & Martin Ch. 9; Karpathy	Character LM
4b	Evaluating Linguistic Knowledge	Marvin & Linzen (2018); BLiMP	Minimal pairs testing
5a	LSTMs	Olah (2015); Hochreiter & Schmidhuber	**Midterm presentations**
5b	LSTM implementation		LSTM coding; **Due:** HW2
6a	Attention Mechanisms	Bahdanau et al. (2015)	Visualizing attention
6b	Attention & Syntax	Vig & Belinkov (2019)	Attention analysis
7a	Transformers I	Vaswani et al. (2017); Illustrated Transformer	**Due:** HW3; Self-attention
7b	Transformers II (BERT/GPT)	Devlin et al.; Radford et al.	Fine-tuning BERT
8a	LLMs & Prompting	GPT-3 paper; Chain-of-thought	GPT prompts
8b	Counterfactual Evaluation	Kaushik et al. (2020); Gardner et al.	Minimal pair testing
9a	Model Evaluation (Surprisal)	Futrell et al. (2019); Wilcox et al.	**Due:** HW4; Computing surprisal
9b	Limitations & Ethics	Bender & Koller (2020); Stochastic Parrots	Bias detection
10a	Final Presentations I		Peer feedback
10b	Final Presentations II		Peer feedback; Wrap-up
Finals			**Due:** Final Report

Note: Each week has two sessions (a/b). Session (a) focuses on concepts; session (b) focuses on hands-on coding.

---

6. Policies

6.1 What you might struggle with (and how to succeed)

- **Time management:** Expect 10-12 hours/week outside class. Start homework early.

- **Python learning curve:** Week 1-2 will be challenging if you're new to programming. Use office hours!
- **Math anxiety:** We explain through intuition and visualization, not proofs. Focus on understanding, not memorization.
- **Reading papers:** Read strategically. Focus on key contributions, not every detail.
- **Coding:** Debug systematically. Pair program. Use Google Colab so everyone has the same environment.

6.2 Quarter System Considerations

This course moves **fast**:

- **Week 3 (quarter) = Week 5 (semester):** Already implementing backprop by hand
- **Week 7 (quarter) = Week 11 (semester):** Already working with transformers
- **No catching up later:** Missing one week = missing 10% of content
- **Use labs effectively:** Don't skip session (b)—that's where you learn to code

6.3 Academic Integrity

Do your own work. You may collaborate on homework, but must write your own code and answers. Cite sources.

6.4 Use of LLMs

LLMs may be used for **support** (debugging, learning syntax) but not to generate solutions. Document any use.

Pedagogical note: Part of this course is understanding how LLMs work. Using ChatGPT to solve homework defeats the purpose—you're trying to build the thing that ChatGPT is!

6.5 Accessibility & Wellness

If you need accommodations, please contact the relevant campus office and talk to me early in the quarter. If you are struggling—academically or personally—please reach out. I really appreciate when students communicate with me, and I'm happy to work with you to make a plan together.

---

7. Resources

Textbooks (Free Online)

- **Jurafsky & Martin (3rd ed.).** *Speech and Language Processing.* https://web.stanford.edu/~jurafsky/slp3/
- **Goodfellow et al.** *Deep Learning.* https://www.deeplearningbook.org/

Online Tutorials

- **PyTorch Tutorials:** https://pytorch.org/tutorials/
- **The Illustrated Transformer:** https://jalammar.github.io/illustrated-transformer/
- **Karpathy's Neural Networks:** https://karpathy.github.io/

Frameworks

- **PyTorch:** Neural network implementation
- **Hugging Face:** Pre-trained models (https://huggingface.co/)
- **Google Colab:** Free GPU access (https://colab.research.google.com/)

---

9. Connection to Course Sequence

This course is **Course 1** in the AI/NLP sequence: - **Course 1 (this course):** Machine Learning for Language (mixed) - Fall Quarter - **Course 2:** Advanced NLP & AI for Linguistics (grad) - Winter Quarter

Skills developed here that transfer to Course 2: - **PyTorch:** Used for implementing interpretability methods - **Transformers:** Course 2 probes transformer representations - **Attention:** Course 2 analyzes attention patterns in depth - **Critical thinking:** Course 2 evaluates interpretability claims

Can be taken standalone if you just want ML/NLP skills without the interpretability focus.

---

10. Who Should Take This Course

Good fit if you:

- Curious about how neural networks process language
- Want to learn Python and PyTorch
- Like hands-on coding (not just theory)
- Interested in linguistics AND computation
- Comfortable with mathematical intuition (no proofs required)

May struggle if you:

- Uncomfortable with programming (we teach from scratch, but it's fast-paced)
- Prefer lecture-only courses (half the time is coding labs)
- Find 10-week quarters overwhelming

No Prerequisites

Seriously—we assume zero programming background. Week 1 starts with "what is a variable?"

For Different Student Populations

- **Linguistics undergrads:** Learn computational skills for NLP career paths
- **CS undergrads:** Gain linguistic perspective on NLP (beyond engineering)
- **Linguistics grad students:** Prepare for computational linguistics research
- **CS grad students:** Understand what NLP systems actually learn about language

---

11. What You'll Build

By the end of the quarter, you'll have:

1. **Perceptron from scratch** (Week 2) - no libraries, just NumPy
2. **Multi-layer network with backprop** (Week 3) - understand gradient descent
3. **Word embeddings** (Week 3) - train your own Word2Vec-style model
4. **Character-level language model** (Week 4) - generate text with RNNs
5. **LSTM classifier** (Week 5) - sentiment analysis
6. **Attention mechanism** (Week 6) - visualize what models attend to
7. **Transformer** (Week 7) - self-attention and multi-head attention
8. **BERT fine-tuning** (Week 7) - use pre-trained models
9. **GPT prompting** (Week 8) - in-context learning experiments
10. **Probing classifier** (Week 9) - test what models learn about syntax

Plus: A **final research proposal** investigating a linguistic question with neural networks.

---

12. Instructor Note

Welcome! This course moves fast, but we'll move together. My goal is to demystify neural networks—show you that they're not magic, just math and code. By Week 10, you'll be able to read an NLP paper and think "I could implement that."

The linguistics perspective is what makes this course special. We don't just ask "does it work?"—we ask "what does it learn?" and "what does that tell us about language?" This is the future of linguistics: using computational models as scientific instruments.

I'm excited to build models and test theories with you. Let's discover what neural networks can teach us about language.

Best, Utku Turk

Advanced NLP & AI for Linguistics

Attention Probing, Causality, and Interpretability

LING 7XXX – Winter Quarter 2027

Time: Wednesday 1:30pm–4:20pm (3 hours/week)

Location: TBD

Instructor: Utku Turk (utkuturk@umd.edu)

Office hours: By appointment

Course site: Canvas

Prerequisites: Background in neural networks (intro ML for language) or permission.

Format: Quarter system (10 weeks)

---

1. Course Description

What do neural language models actually learn? How can we probe their internal representations? Can we establish causal relationships between model components and linguistic behavior?

Neural networks are often treated as black boxes. In this seminar, we will open them up together. This advanced seminar explores cutting-edge methods for interpreting and analyzing neural networks through a linguistic lens. We focus on three core areas: (i) **attention probing**—analyzing attention patterns to understand how models process syntactic dependencies, (ii) **causality and intervention**—using counterfactual methods to establish causal relationships, and (iii) **representation analysis**—extracting and interpreting hidden representations for linguistic knowledge.

Students will learn to design rigorous probing experiments, implement causal intervention methods, analyze attention entropy and head specialization, and critically evaluate interpretability claims. This is a research-oriented seminar where students develop original proposals for investigating what neural models learn about language.

---

2. Learning Objectives

Upon successful completion of this course, students will be able to: 1. Design and implement probing classifiers to test for linguistic knowledge. 2. Analyze attention patterns for syntactic dependencies and semantic roles. 3. Apply causal intervention methods (counterfactuals, interchange interventions). 4. Compute and interpret attention entropy and head specialization metrics. 5. Extract and visualize hidden representations using dimensionality reduction. 6. Critically evaluate interpretability claims and methodological choices. 7. Propose original research investigating model representations.

---

3. Course Structure

Weekly Rhythm

Component	What it looks like
Paper discussion	Student-led presentation + group discussion of 1–2 papers (45 min).

Component	What it looks like
Methods tutorial	Hands-on coding session implementing the week's method (90 min).
Research clinic	Workshopping student project ideas, designs, and analyses (30 min).

Research Seminar Format

This is a **research-level** course. You'll read cutting-edge papers (many from 2024-2025), implement state-of-the-art methods, and develop original research proposals. I expect you to engage critically, propose extensions, and identify flaws in published work.

Paper Presentation Rotation: You will lead discussion twice over the quarter. Your job: 1. **Situate the work** (2 min): What problem does this solve? Why does it matter? 2. **Explain the method** (3 min): Walk us through the key technical contribution 3. **Evaluate the claims** (3 min): What did they show? What did they NOT show? 4. **Propose extensions** (2 min): How would you improve or extend this work?

Prepare 3-5 discussion questions that get at the core tensions in the paper.

Method Implementation Showcase: After implementing each method, you'll demonstrate it on a linguistic phenomenon of your choice: - Week 3: "Here's what my probing classifier found about verb transitivity…" - Week 5: "I discovered that attention head 7-3 specializes in…" - Week 7: "When I intervene on this representation, the model's predictions change because…"

This builds your portfolio of interpretability tools AND helps you discover research questions.

Tools and Methods

- **Probing:** Linear probes, control tasks, diagnostic classifiers.
- **Attention analysis:** Attention weights, entropy, rollout, head pruning.
- **Causality:** Counterfactual data, causal mediation, interchange interventions.
- **Representation:** PCA, t-SNE, UMAP, CKA similarity.
- **Frameworks:** Hugging Face Transformers, AllenNLP Interpret, Captum.

---

4. Course Requirements

4.1 Grading

Item	%	What counts
Participation	15	Active discussion; constructive feedback.
Paper presentations (2x)	20	Lead discussion on 2 papers (10% each).
Method implementations (3x)	30	Implement and document 3 methods (10% each).
Final project	35	Full implementation: paper, code, presentation.

4.2 Paper Presentations

Lead discussion on 2 papers over the quarter. Summarize question/method/findings, critique methodology, propose extensions, and prepare discussion questions.

4.3 Method Implementations

Choose 3 methods to implement (one from each core area: probing, attention, causality/representation). Includes working code (Colab), brief write-up (2-3 pages), and in-class demonstration.

4.4 Final Project

Conduct an original research project investigating a linguistic question. * Research question grounded in linguistic theory (syntax, semantics, phonology, etc.). * Implementation of interpretability method (probing, intervention, analysis). * Systematic evaluation. * **Format:** 10–12 page paper (ACL style) + code + presentation.

---

5. Course Schedule (10 Weeks)

Wk	Topic	Readings (Selected)	Methods / Due
1	Intro & Probing I	Newell (1973); Marr (1982); Belinkov & Glass (2019)	Setup Hugging Face; Linear probes
2	Probing II: Methodology & Controls	Hewitt & Liang (2019); Hewitt & Manning (2019)	Control tasks
3	Probing III: Implementation	Conneau et al. (2018); Pimentel et al. (2020)	**Due:** Method 1 (Probing); Showcase
4	Attention I: Patterns & Visualization	Clark et al. (2019); Vig & Belinkov (2019)	Visualizing attention
5	Attention II: Entropy & Specialization	Voita et al. (2019); Michel et al. (2019)	**Due:** Method 2 (Attention); Head pruning
6	Causal I: Counterfactuals	Kaushik et al. (2020); Gardner et al. (2021)	Data augmentation
7	Causal II & Geometry	Vig et al. (2020); Kornblith et al. (2019)	**Due:** Method 3 (Causality/Geometry); CKA
8	Cross-linguistic Probing & Limitations	Pires et al. (2019); Belinkov & Glass (2019)	Multilingual models; Critique workshop
9	Final Project Workshop & Emergent Abilities	Wei et al. (2022); Schaeffer et al. (2023)	Individual consultations
10	Final Presentations		Peer feedback; Wrap-up
Finals			**Due:** Final Project Paper

Note: Each week is a single 3-hour session. Structure: 45 min paper discussion → 90 min methods tutorial → 30 min research clinic.

---

6. Policies

6.1 What you might struggle with (and how to succeed)

- **Time management:** This is advanced. Budget 10–12 hours/week outside class.
- **Reading papers:** Read strategically. Focus on methods and results. Take notes by hand.
- **Implementing methods:** Start with toy examples. Document as you go. Use Hugging Face extensively.
- **Quarter pace:** Week 3 (quarter) = Week 4-5 (semester). Very fast progression.

- **Debugging:** Interpretability code can be tricky. Use office hours. Share code snippets on Slack/Discord.

6.2 Academic Integrity

You may collaborate on understanding methods, but must implement your own code and write your own analyses. Cite all sources.

6.3 Use of LLMs

LLMs may be used for **debugging** and **understanding** code, but not for generating implementations. You must understand every line. Document any use.

6.4 Accessibility & Wellness

If you need accommodations, please contact the relevant campus office and talk to me early in the quarter. If you are struggling—academically or personally—please reach out. I really appreciate when students communicate with me, and I'm happy to work with you to make a plan together.

---

7. Resources

Key Surveys

- **Belinkov & Glass (2019).** "Analysis Methods in Neural Language Processing: A Survey." *TACL.*
- **Rogers et al. (2020).** "A Primer on Neural Network Architectures for Natural Language Processing." *JAIR.*

Tools

- **Hugging Face:** Pre-trained models and datasets
- **Captum:** Model interpretability for PyTorch
- **BertViz:** Attention visualization

Papers (Available on Course Site)

All assigned papers available through library access or as preprints on arXiv.

---

9. Connection to Course Sequence

This course is **Course 2** (capstone) in the AI/NLP sequence: - **Course 1:** Machine Learning for Language (mixed) - Fall Quarter - **Course 2 (this course):** Advanced NLP & AI for Linguistics (grad) - Winter Quarter

Skills from Course 1 that are essential here: - **PyTorch:** Used for all implementations - **Transformers:** We probe transformer models extensively - **Attention mechanisms:** Now analyzed in detail - **Python programming:** Need to be comfortable debugging - **Hugging Face:** Primary framework for loading models

Integration in Course 2: - **Course 1 question:** "Can transformers learn syntax?" - **Course 2 answer:** "Here's how to test that with probing classifiers, and here's what we find…"

Can be taken standalone if you have: - Strong Python/PyTorch background - Familiarity with transformers (BERT, GPT) - Graduate-level research skills

---

10. Who Should Take This Course

Good fit if you:

- Want to conduct interpretability research
- Comfortable with Python, PyTorch, and transformers
- Interested in what models learn (not just engineering performance)
- Like reading cutting-edge papers (2024-2025 research)
- Enjoy debugging and troubleshooting code

May struggle if you:

- No neural network background (take Course 1 first)
- Uncomfortable with advanced Python (need strong programming skills)
- Prefer lecture-style seminars to research workshops
- Find 10-week quarters overwhelming

For Graduate Students

This seminar is ideal if you're: - Planning dissertation/research on interpretability or computational linguistics - Preparing for academic jobs requiring computational skills - Aiming to publish in ACL, EMNLP, *CL*, *Computational Linguistics* - Interested in AI safety, robustness, and responsible AI development

---

11. Expected Background

Required Knowledge

You should be comfortable with: - **Neural networks:** MLPs, RNNs, LSTMs, transformers (from Course 1 or equivalent) - **PyTorch:** Implementing models, loading pre-trained weights, debugging - **Python:** NumPy, pandas, matplotlib, seaborn - **Hugging Face:** Loading models, tokenizers, datasets - **Linguistics:** Basic syntax, semantics, morphology (so you can formulate research questions)

Will Learn in This Course

- **Probing:** Designing diagnostic classifiers, control tasks
- **Attention analysis:** Head specialization, entropy, pruning
- **Causal methods:** Counterfactuals, interchange interventions
- **Representation geometry:** CKA, SVCCA, PWCCA
- **Critical evaluation:** Methodological pitfalls in interpretability research

---

12. Final Project Expectations

Your final project should: 1. **Motivate a linguistic question** (1-2 pages): e.g., "Do models learn binding constraints?" 2. **Review relevant work** (1-2 pages): Existing probing studies, what's unresolved 3. **Implement an interpretability method** (core of paper): - Probing classifiers, attention analysis, or causal intervention - Clear methodology, controls, evaluation metrics 4. **Analyze results systematically** (2-3 pages): What did you find? What does it mean? 5. **Discuss implications** (1-2 pages): For linguistic theory, for NLP systems, for future work

Format: ACL 2-column style (10-12 pages)

Code: Publicly available GitHub repository with README

Outcome: Should be submittable to *ACL, **BlackboxNLP, *SCiL, or similar venues with minor revisions.

---

13. Relationship to Package 1 (Psycholinguistics)

Students who complete both Package 1 (Psycholinguistics) and Package 2 (AI/NLP) can conduct cutting-edge research at their intersection:

Research Questions You Can Answer

- **Compare neural models with human processing:**
  - Do BERT's surprisal predictions match human reading times?
  - Do neural models show agreement attraction like humans?
  - Do attention patterns align with human garden-path effects?
- **Use models to generate psycholinguistic stimuli:**
  - LLMs generate sentences → human experiments validate → probing checks what models learned
- **Model human errors:**
  - Build Bayesian models (Package 1, Course 2) of human agreement errors
  - Compare with neural model errors (Package 2, Course 2)

Combined Skills

- **Experiments + Models:** Design human experiments (Package 1) and neural model probes (Package 2) to test the same theory
- **Bayesian + Neural:** Hierarchical Bayesian models (Package 1) compared with neural network models (Package 2)
- **Production + NLP:** Morphological planning (Package 1) tested in both humans and language models

---

14. Instructor Note

This seminar is intense but rewarding. By Week 10, you'll have implemented multiple interpretability methods and proposed original research. Many students turn these projects into first-author publications.

The quarter format means we move fast—Week 3 you're already showcasing probing classifiers. But the research clinic format ensures you get individualized feedback and support.

I'm committed to helping you succeed. Use office hours, ask questions, share code on our class Slack. This is a collaborative research environment.

Looking forward to discovering what models learn about language—together.

Best, Utku Turk