Machine Learning for Language
Neural Networks, Linguistic Structure, and NLP Applications

LING 2XXX – Fall 2026

Course information

Time	TBD
Location	TBD
Instructor	Utku Turk
Email	utkuturk@umd.edu
Office hours	By appointment
Course site	Canvas

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

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 semester, 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. Background in linguistics helpful but not required.

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 analysis, 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 under the hood (Tue/Thu first half).
Coding lab	Hands-on Python work: implementing models, debugging, running experiments (Tue/Thu second half).
Homework	Biweekly 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 1–5**: Build your own neural networks from scratch (no magic libraries—you'll implement backprop by hand!)
- **Weeks 6–9**: Explore what models learn about language structure (syntax, semantics, pragmatics)
- **Weeks 10–14**: Work with state-of-the-art transformers and LLMs
- **Weeks 15–17**: 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 (no local setup required)
- **Hugging Face:** pretrained models and datasets

Lab structure

Labs are designed to build your skills progressively. Each lab is self-contained and followable, with step-by-step tutorials that prepare you for homework assignments. Labs include:

Introductory sessions (Weeks 1–4):

- **Intro:** Python basics, Colab setup, running your first code
- **Text as Strings:** String manipulation, tokenization, preprocessing
- **Classification:** Binary classification concepts, train/test splits
- **Linguistic Classification I:** Classifying modal readings (epistemic vs. deontic)
- **Features:** Feature extraction from text, linguistic features
- **Weights and Matrix Representations:** Vectors, matrices, dot products

- **Weights and Bias:** Linear models, parameters, learning
- **Sigmoid:** Activation functions, probabilities
- **Linguistic Classification II:** Genericity detection (generic vs. episodic)

Advanced sessions (Weeks 5–16): Build on these foundations to implement RNNs, LSTMs, attention, transformers, and probing experiments. Includes a dedicated lab on Universal Dependencies for working with machine-readable syntactic annotations.

4 Course requirements

4.1 Grading

Item	%	What counts
Participation + in-class coding	15	Active engagement in labs; completion of in-class exercises.
Homework (5 assignments)	40	Coding + conceptual questions (8% each).
Midterm project	10	Classifier proposal: research question, dataset plan, model design, predictions.
Final project	35	Detailed research proposal: motivation, literature review, methodology, predictions, evaluation plan.

4.2 Homework policy

Homework is due on Canvas by 11:59pm on the specified date. Late submissions lose 10% per day (up to 3 days). After 3 days, maximum 50% credit without prior arrangement.

You may collaborate on homework, but you must write your own code and answers. Copying code from classmates or online sources without attribution is academic dishonesty.

4.3 Projects

Midterm project (Week 9): Propose a neural classifier for a linguistic phenomenon. Choose one: (i) modal reading classification (epistemic vs. deontic), (ii) genericity detection (generic vs. episodic), (iii) aspect classification (perfective vs. imperfective), or (iv) optionality prediction (e.g., plural marking). Your proposal should include: research question, dataset description (where to get it or how to create it), model architecture, features to use, predicted results, and evaluation metrics. 2–3 page proposal.

Final project (Finals week): Detailed research proposal investigating a linguistic question using neural networks. This should be a publishable-quality proposal including: (i) clear research question and hypothesis with linguistic motivation, (ii) literature review situating your question, (iii) detailed dataset plan (sources, annotation scheme, size), (iv) proposed model architecture and training procedure, (v) specific predictions about what the model will learn and why, (vi) evaluation plan (metrics, baselines, error analysis), (vii) expected results and their theoretical implications, and (viii) potential limitations and future directions. 8–10 page proposal in ACL format + 10-minute presentation.

Important: Your proposal must address a question from theoretical linguistics, not just applied NLP. Ground your work in existing syntactic, semantic, phonological, or sociolinguistic theory. Example topics: testing Distributed Morphology feature bundles, quantifying language contact effects, analyzing ellipsis licensing, cross-linguistic optionality patterns (following Polinsky, Preminger, etc.), or encoding semantic composition (Kratzer, Heim). See SCiL proceedings for theory-driven computational work.

5 Course schedule (16-week semester)

Schedule subject to change. Canvas is the live version.

Wk	Date	Topic	Readings / Resources	Lab / due
1	–	Introduction; neural networks and language	**Read:** Newell (1973) "You can't play 20 questions with nature and win"; Marr (1982) *Vision* [Ch. 1, levels of analysis]. **Optional:** Linzen et al. (2016) "Assessing the ability of LSTMs to learn syntax-sensitive dependencies".	Python basics I: variables, lists, loops
2	–	Representing words; distributional semantics	**Read:** Jurafsky & Martin (2024) Ch. 6 [word2vec]. **Optional:** Mikolov et al. (2013) "Distributed representations of words".	Python basics II: functions, dictionaries; vector operations
3	–	Perceptrons; gradient descent; loss functions	**Read:** Goodfellow et al. (2016) Ch. 6 [§6.1–6.2]. **Optional:** 3Blue1Brown neural network series (YouTube).	Implementing a perceptron from scratch
4	–	Multi-layer perceptrons; backpropagation	**Read:** Goodfellow et al. (2016) Ch. 6 [§6.5]. **Optional:** Olah (2015) "Calculus on computational graphs".	PyTorch basics; building an MLP **Due:** HW1
5	–	Word embeddings; semantic and syntactic structure	**Read:** Jurafsky & Martin (2024) Ch. 6 [word2vec]; Ettinger (2020) "What BERT is not" [§1–3]. **Optional:** Hewitt & Manning (2019) "A structural probe for finding syntax in word representations"; Kann et al. (2019) "Verb argument structure alternations in word and sentence embeddings".	Exploring word2vec; probing for linguistic structure
6	–	RNNs and language modeling; text generation	**Read:** Jurafsky & Martin (2024) Ch. 9 [RNNs]; Karpathy (2015) "The unreasonable effectiveness of RNNs". **Optional:** Gulordava et al. (2018) "Colorless green recurrent networks dream hierarchically"; Linzen et al. (2016) "Assessing the ability of LSTMs to learn syntax-sensitive dependencies".	Building a character-level language model; generating text
7	–	Evaluating linguistic knowledge; minimal pairs	**Read:** Marvin & Linzen (2018) "Targeted syntactic evaluation of language models"; Warstadt et al. (2020) "BLiMP: A benchmark of linguistic minimal pairs". **Optional:** Futrell et al. (2019) "Neural language models as psycholinguistic subjects"; Wilcox et al. (2018) "What do RNN language models learn about filler–gap dependencies?"	Designing minimal pairs; testing agreement **Due:** HW2
8	–	LSTMs; vanishing gradients; sequence-to-sequence	**Read:** Olah (2015) "Understanding LSTMs". **Optional:** Hochreiter & Schmidhuber (1997) [original LSTM paper].	Implementing an LSTM; midterm project work session
9	–	Midterm project presentations	–	**Due:** Midterm project (text classifier)
10	–	Attention mechanisms; modeling long-distance dependencies	**Read:** Bahdanau et al. (2015) "Neural machine translation by jointly learning to align and translate"; Vig & Belinkov (2019) "Analyzing the structure of attention in a transformer language model". **Optional:** Clark et al. (2019) "What does BERT look at?"	Visualizing attention on syntactic dependencies **Due:** HW3
11	–	Transformers I: self-attention and architecture	**Read:** Vaswani et al. (2017) "Attention is all you need" [§1–3]; Alammar (2018) "The illustrated transformer". **Optional:** Htut et al. (2019) "Do attention heads in BERT track syntactic dependencies?"	Implementing self-attention; visualizing attention patterns
12	–	Transformers II: BERT, GPT, and transfer learning	**Read:** Devlin et al. (2019) "BERT" [§1–3]; Radford et al. (2019) "GPT-2" [§1–2]. **Optional:** Jawahar et al. (2019) "What does BERT learn about the structure of language?"	Fine-tuning BERT for text classification
13	–	Large language models; prompting and applications	**Read:** Brown et al. (2020) "GPT-3" [§1–3]; Wei et al. (2022) "Chain-of-thought prompting". **Optional:** Papadimitriou & Jurafsky (2020) "Learning music helps you read: Using transfer to study linguistic structure in language models".	Experimenting with GPT prompts; building applications **Due:** HW4
14	–	Counterfactual evaluation; minimal pair testing	**Read:** Kaushik et al. (2020) "Learning the difference that makes a difference with counterfactually-augmented data"; Gardner et al. (2020) "Evaluating models' local decision boundaries via contrast sets". **Optional:** Teney et al. (2020) "Learning what makes a difference from counterfactual examples and gradient supervision".	Designing minimal pair tests; counterfactual data augmentation
15	–	Model evaluation; surprisal and human behavior	**Read:** Futrell et al. (2019) "Neural language models as psycholinguistic subjects"; Wilcox et al. (2020) "On the predictive power of neural language models for human real-time comprehension behavior". **Optional:** Hale (2001) "A probabilistic Earley parser as a psycholinguistic model"; Goodkind & Bicknell (2018) "Predictive power of word surprisal for reading times is a linear function of language model quality"	Computing surprisal; comparing model and human reading times

6 Policies

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

This course is challenging. Here's what students typically struggle with and evidence-based advice:

Time management:

- Expect 8–10 hours/week outside class: 2–3 hours reading, 4–5 hours coding, 1–2 hours reviewing notes
- Don't cram: Distributed practice works better than massed practice (Dunlosky et al., 2013)
- Start homework early: Debugging takes longer than you think

Note-taking:

- Take notes by hand, not laptop: Handwriting improves retention (Mueller & Oppenheimer, 2014)
- Why handwriting works: You can't transcribe verbatim, forcing you to process and synthesize
- Avoid digital note-taking apps during class: They encourage procrastination and shallow processing
- After class: Transfer key concepts to digital notes for organization

Computer use in class:

- Labs only: Laptops/tablets for coding labs only, not lectures
- Why: Multitasking hurts learning (Sana et al., 2013); students with laptops score lower on tests
- Exception: Accommodations for students with documented needs

Reading papers:

- Don't read linearly: Start with abstract → conclusions → figures → introduction → methods
- Take notes while reading: Summarize each section in your own words
- Read actively: Ask "what's the claim?", "what's the evidence?", "what are the limitations?"

Coding:

- Debug systematically: Print intermediate values, test small pieces, read error messages carefully
- Use LLMs wisely: For syntax help and debugging, not for generating solutions
- Pair program: Explaining your code to someone else helps you understand it better

Resources:

- How to Read a Paper (Keshav, 2007): `https://web.stanford.edu/class/ee384m/Handouts/HowtoReadPaper.pdf`
- Learning How to Learn (free Coursera course): `https://www.coursera.org/learn/learning-how-to-learn`
- The Science of Effective Learning (Dunlosky et al., 2013): `https://doi.org/10.1177/1529100612453266`

6.2 Attendance

Attendance is expected. Labs are hands-on and difficult to replicate remotely. If you must miss class, notify me in advance and complete the lab exercises on your own.

6.2 Academic integrity

Do your own work; cite sources. You may collaborate on homework, but submitted code and answers must be your own. Copying code from classmates or online sources without attribution is plagiarism.

When using code from Stack Overflow, tutorials, or documentation, cite the source with a comment and demonstrate understanding by explaining what it does.

6.3 Use of ChatGPT and LLMs

LLMs may be used for **support** (debugging, learning syntax, explaining error messages) but not to generate solutions to homework problems or projects. Any use must be documented with a brief statement and chat log.

Why this matters: The goal is for *you* to learn how these systems work. Using them to do your homework defeats the purpose. LLMs also make mistakes, and you need to be able to catch them.

6.4 Accessibility & wellness

If you need accommodations, please contact the relevant campus office and talk to me early in the semester. 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 (all free online)

- Jurafsky & Martin (2024). *Speech and Language Processing* (3rd ed. draft). `https://web.stanford.edu/~jurafsky/slp3/`
- Goodfellow, Bengio, & Courville (2016). *Deep Learning*. `https://www.deeplearningbook.org/`
- Eisenstein (2019). *Introduction to Natural Language Processing*. `https://github.com/jacobeisenstein/gt-nlp-class/blob/master/notes/eisenstein-nlp-notes.pdf`

Technical resources

Category	Links
Python	Official tutorial `https://docs.python.org/3/tutorial/`; Colab `https://colab.research.google.com/`
PyTorch	Tutorials `https://pytorch.org/tutorials/`; Documentation `https://pytorch.org/docs/`
Hugging Face	Transformers `https://huggingface.co/docs/transformers/`; Datasets `https://huggingface.co/docs/datasets/`
Visualization	Alammar's blog `https://jalammar.github.io/`; Distill.pub `https://distill.pub/`

This syllabus is a living document and may be updated during the semester. Major changes will be announced in class and on Canvas. Last updated: January 3, 2026

Appendix: Quarter-system (UChicago-style) version (10 weeks)

A compact quarter adaptation with the same linguistics focus.

Wk	Topic	Readings	Lab / due
1	Intro; can ANNs learn syntax? + distributional semantics	**Req:** Newell (1973) "You can't play 20 questions"; Marr (1982) [Ch. 1]; Jurafsky & Martin Ch. 6.	Python basics; vector operations
2	Perceptrons + MLPs; probing for linguistic structure	**Req:** Goodfellow et al. Ch. 6 [§6.1–6.2, 6.5]; Ettinger (2020) "What BERT is not".	Implementing MLP; probing experiments
3	Language models; grammatical structure	**Req:** Gulordava et al. (2018) "Colorless green recurrent networks"; Marvin & Linzen (2018) "Targeted syntactic evaluation".	Building LM; grammaticality tests **Due:** HW1
4	Testing syntactic knowledge; minimal pairs	**Req:** Warstadt et al. (2020) "BLiMP"; Wilcox et al. (2018) "What do RNNs learn about filler–gap dependencies?"	Designing minimal pairs; testing agreement
5	LSTMs; long-distance dependencies	**Req:** Olah (2015) "Understanding LSTMs"; Futrell et al. (2019) "Neural LMs as psycholinguistic subjects".	LSTM implementation **Due:** HW2 (midterm probing project)
6	Attention; modeling syntactic dependencies	**Req:** Bahdanau et al. (2015); Vig & Belinkov (2019) "Analyzing attention structure".	Visualizing attention on dependencies
7	Transformers; what linguistic knowledge emerges?	**Req:** Vaswani et al. (2017) [§1–3]; Jawahar et al. (2019) "What does BERT learn about language structure?"	Probing BERT for syntax/semantics **Due:** HW3
8	Cross-linguistic syntax; typological generalization	**Req:** Ravfogel et al. (2019) "Inductive biases of RNNs"; Papadimitriou & Jurafsky (2020) "Learning music helps you read".	Testing on artificial languages
9	Incremental processing; surprisal; limitations	**Req:** Wilcox et al. (2020) "Predictive power of neural LMs"; McCoy et al. (2019) "Right for the wrong reasons".	Computing surprisal; adversarial tests **Due:** HW4
10	Final project presentations	–	5-min presentations; final project work session
Finals	–	–	**Due:** Final project (ACL-style paper + code)

Quarter grading:

Item	%	What counts
Participation + in-class coding	15	Active engagement in labs.
Homework 1–2	20	Coding + conceptual (10% each).
Midterm project (HW2)	10	Classifier proposal.
Homework 3–4	20	Coding + conceptual (10% each).
Final project	35	Detailed research proposal in ACL format + presentation.