CS 639: Foundation Models
Course Overview

Fred Sala

University of Wisconsin-Madison

Jan. 20, 2026

Logistics: Lecture Location

- In-person in **Chemistry S413**
  - Will have slides
  - Occasionally whiteboard for derivations.

- Planning to record---final decision TBD.

Logistics: Enrollment

- Currently at capacity, approx. 200 students
  - Some folks on waitlist may not make it in
  - Decent chance many of the waitlist folks will

  - **Sorry** ☹ … will be offered again

Logistics: Teaching Team

Instructor: **Fred Sala**

- Location: 5514 Morgridge Hall
- Office Hours: TBD

TAs: **Dyah Adila, Sonia Cromp, Samuel Guo, Akshat Singhal, Yichen Wang**

- Location: Morgridge Huddle Room B2576
- Office Hours: Calendar

- Note: times possibly **subject to change**

Logistics: Content

Three locations:

- 1. **Course website**: https://pages.cs.wisc.edu/~fredsala/cs639/
- 2. **Piazza**. https://piazza.com/wisc/spring2026/d11c
  - access code: *introtofm*
  - **Preferred for questions!**
- 3. **Canvas**

Flickr Photo by Jacob Surland - creative commons licensed (BY-NC)

Course Content / Schedule

Tuesday Jan. 20	**Lecture 1:** Introduction and Class Overview	• On the Opportunities and Risks of Foundation Models
Thursday Jan. 22		
Tuesday Jan. 27		
Thursday Jan. 29		
Tuesday Feb. 3		
Thursday Feb. 5		
Tuesday Feb. 10		
Thursday Feb. 12		
Tuesday Feb. 17		
Thursday Feb. 19		
Tuesday Feb. 24		
Thursday Feb. 26		

Tuesday Mar. 3	**Lecture 13:** Multimodal Architectures II	• SAM 2 • MMMU
Thursday Mar. 5		
Tuesday Mar. 10		
Thursday Mar. 12		
Tuesday Mar. 17		
Thursday Mar. 19		
Tuesday Mar. 24		
Thursday Mar. 26		• DeepSeek-R1

Tuesday Apr. 7	**Lecture 21:** Evaluation I: Metrics and Benchmarks	• HELM • MMLU • MMLU-Pro
Thursday Apr. 9	**Lecture 22:** Evaluation II	• LLM-as-a-Judge: MT-Bench & Chatbot Arena • G-Eval: NLG Evaluation using LLMs
Tuesday Apr. 14	**Lecture 23:** Scaling I: Laws, MoEs and More	• Scaling Laws for Neural Language Models • Switch Transformers • GLaM: Efficient Scaling of Language Models
Thursday Apr. 16	**Lecture 24:** Scaling II: Test-time Scaling	• Compute-Optimal Large Language Models
Tuesday Apr. 21	**Lecture 25:** Agents I	• ReAct: Reasoning + Acting • Toolformer
Thursday Apr. 23	**Lecture 26:** Agents II	• AgentBench • WebArena • Voyager
Tuesday Apr. 28	**Lecture 27:** Applications: FMs for Science & Medicine	• Med-PaLM 2 • ClimaX
Thursday Apr. 30	**Lecture 28:** Future Areas	• AI Index Report

Logistics: Lecture Formats

Most class sessions:

- **Type 1: Lectures**
  - Mostly slides, some whiteboard
  - Will take some breaks, 1-2 during the lecture
  - Can ask questions---during lecture and breaks

- **Type 2: Guest Lectures**
  - More info later

- Combination of these two.

Logistics: Assignments & Grades

Homeworks:

- 6 or so, worth 50% total
- Posted after class; due when class starts on due date. About 2 weeks given for each one
- Combination of conceptual, implementation, calculation

In-class quizzes:

- Using Top Hat, for bonus points

Midterm

- Worth 20%. More info coming soon. (Note: no final exam).

Final Project:

- 30% total, groups of 5-10; proposal midway. **More info soon!**

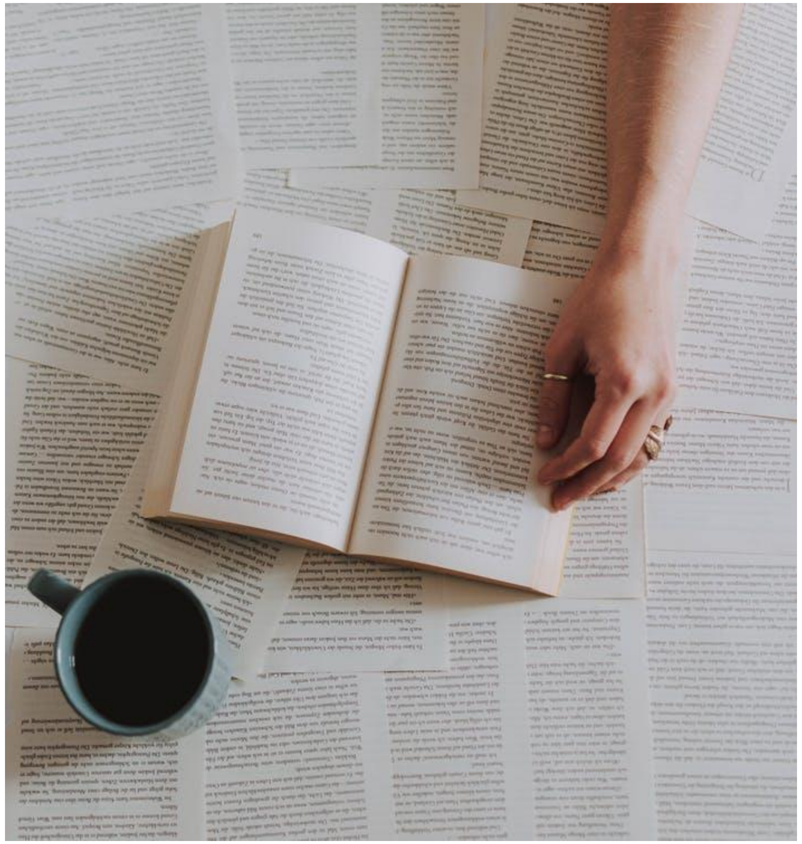
Class Setup: Reading

No overall textbook

- I will post useful notes, primers, papers
  - See schedule page

- We don't expect you to read everything---many of the posted content is long
  - But, it's often useful to find relevant pieces and ask questions about them

- Expect **new papers** (submitted during the timeframe of the class)



Class Setup: Background

More on this at the end of class, but

- **Basic ML**
  - A few review lectures coming up soon
- **Technical components:**
  - Linear Algebra
  - Calculus
  - Probability

Note: this class is partially **conceptual** and partially **technical**

Class Setup: Goals

Two goals:

- Become acquainted with **how to use** large pretrained/language/foundation models
- Understanding the technical underpinnings of these models and ***why*** they work

Note: if you are only interested in a very broad overview of ML, then CS 540 or 760 might be a better choice.

Class Setup: Goals II

Mini-goals:

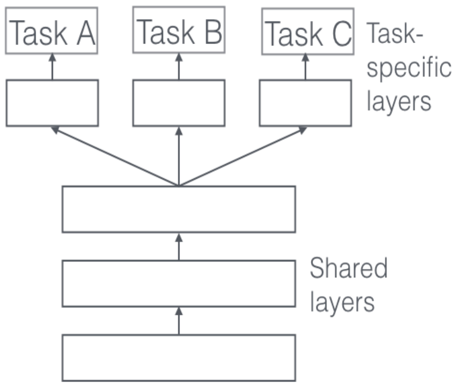
- **Understanding** research
- **Big picture/**ML ecosystem
- **Intuition** around modern ML paradigms

Break & Questions

What Is a Foundation Model?

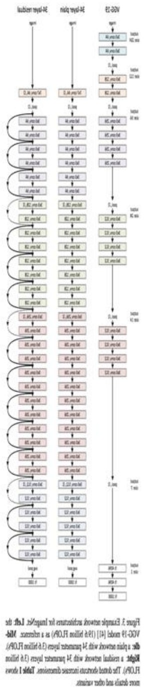
Three Historical Trends

- Brief introduction, more to come, but can explain some of **why** and **what**
- **FMs** start being developed in ~2018-2020



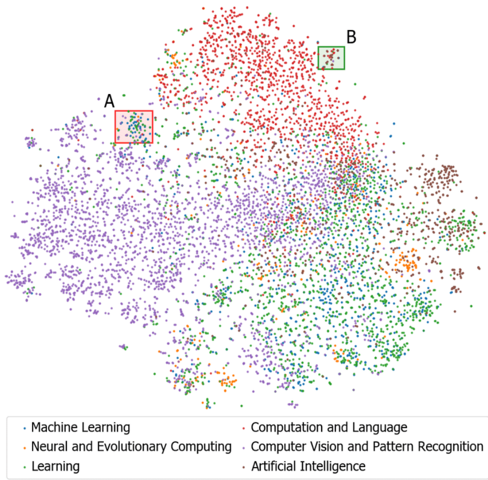
J. Ray

1. **Multitask** models (old!)
2. **Pretrained models** and fine-tuning (2015 onwards)
3. **Word embeddings** and language models (2013 onwards)



He et al ‘16.

Embeddings for arXiv papers (6 ML categories)



Lo et al ‘19.

1. Multitask Models: What's a Task?

A little bit of terminology: in ML we build a model **f** to solve a task ***T***. We train **f** on data pertaining to the task

- Example: **mushroom safety classification.**
  - **f** must take in a mushroom image and predict {safe, poisonous}
  - Training data:

$$(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(n)}, y^{(n)})$$

safe

poisonous

safe

- We can train different models to do different tasks

1. Multitask Models: Handling Multiple Tasks

A little bit of terminology: in ML we build a model **f** to solve a task ***T***. We train **f** on data pertaining to the task

- We can train different models to do different tasks
- Example:
  - T_1 is a mushroom safety classification task; train $\mathbf{f}_1$
  - T_2 is a potato safety classification task; train $\mathbf{f}_2$

T_1


safe

poisonous

safe

T_2

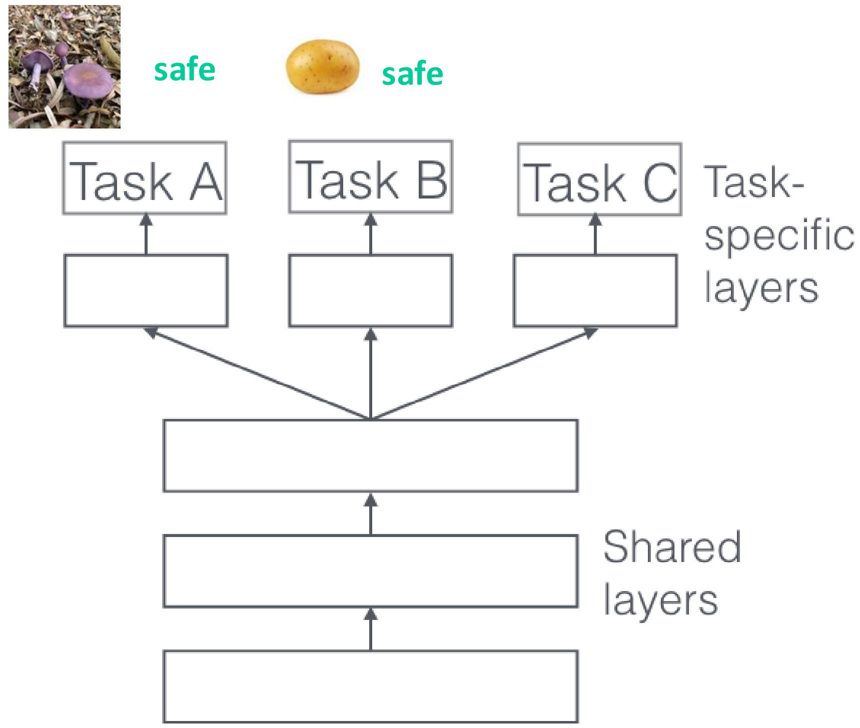

safe

poisonous

1. Multitask Models: An Alternative

Idea: Given tasks $T_1, \ldots, T_k$, rather than training k separate models, train a common base and task-specific "heads"

- Related to ***transfer learning***
- *Why?* If tasks are related, there's
  - Common information
  - Equivalent to more data



J. Ray

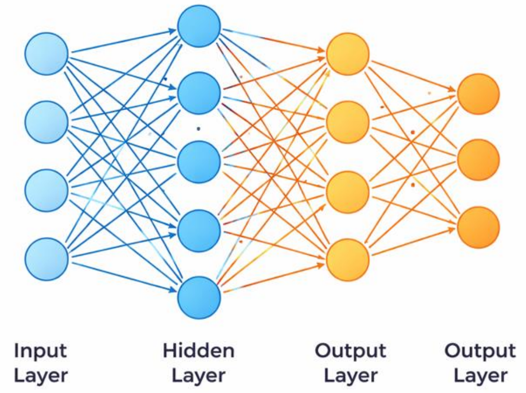
Differences (vs. modern FMs)

- Usually **fixed tasks**
- Train on **data from all tasks** (limited)

2. Pretraining and Fine-tuning

Motivation: Training from scratch is expensive. ***Why***?

- What are the ingredients for a model? We need

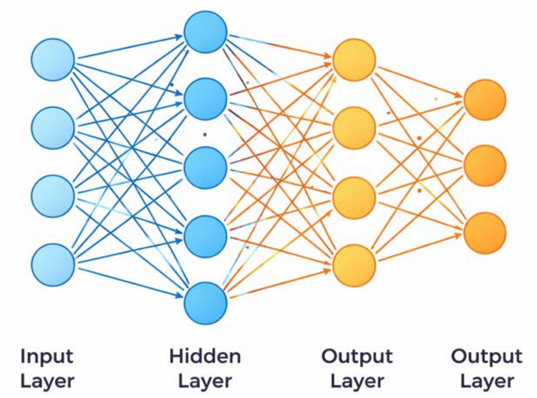


+

+

Model **Data** **Hardware**

2. Pretraining and Fine-tuning



Motivation: Training from scratch is expensive.

- Deep learning revolution (2010-). Each factor changes…
  - Larger datasets (for example, ImageNet)
  - Larger hardware resources (GPUs, multiple GPUs)
  - Produces larger models
    - LeNet: 60 thousand. AlexNet: 60 **million**.

- Much of 2010-2015 CV research builds larger and larger CNNs, so training costs ↑

2. Pretraining and Fine-tuning

Motivation: Training from scratch is expensive.

- Much of 2010-2015 CV research builds larger and larger CNNs, so training costs ↑
- Example: very deep residual networks (ResNets)

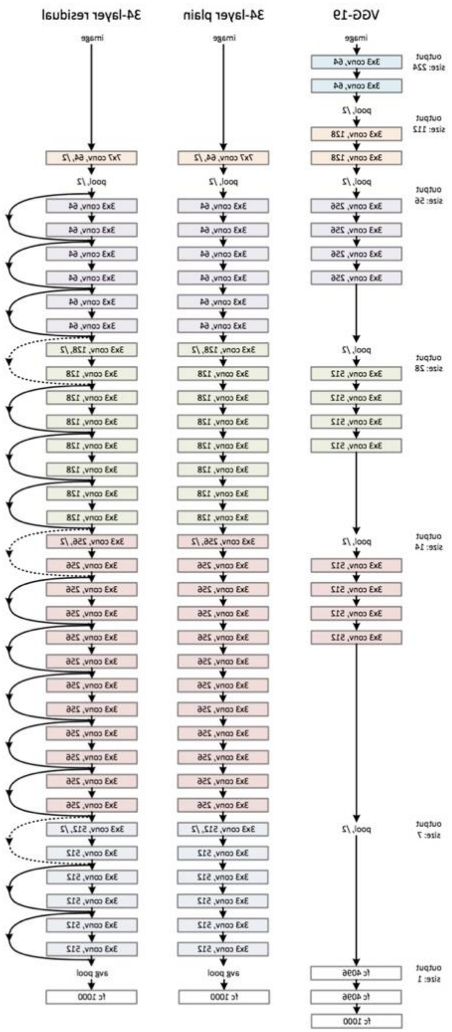


Figure 3. Example network architectures for ImageNet. **Left:** the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle:** a plain network with 34 parameter layers (3.6 billion FLOPs). **Right:** a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

He et al '16.

2. Pretraining and Fine-tuning

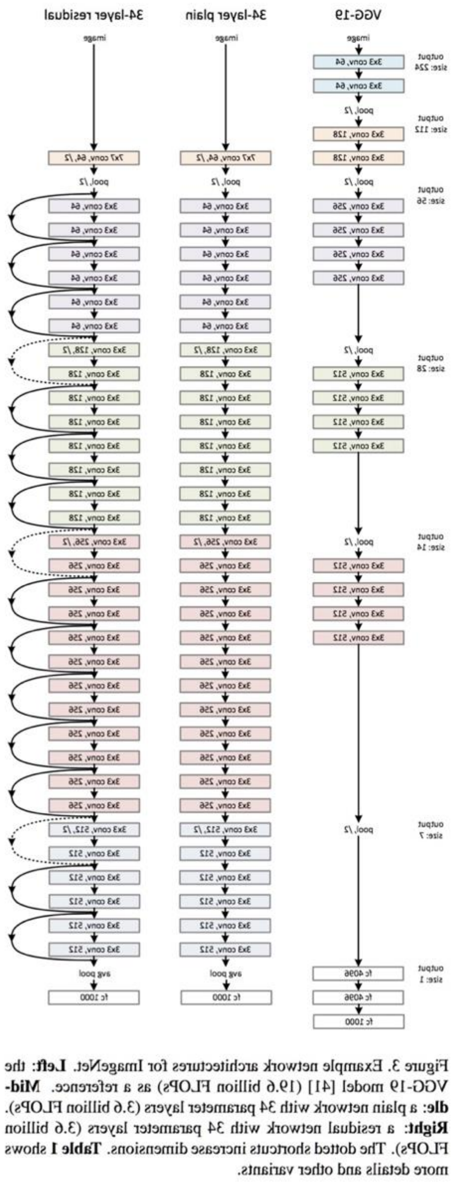
Motivation: Training from scratch is expensive.

Idea: ***pretrain*** a single model on a dataset

- Then ***fine-tune*** to adapt to downstream task
- Ex: pretrained ResNets on ImageNet (2015-)

Issues:

- Other data modalities/domains? Could build ImageNet analogue, but expensive
- Leads to **self-supervised training** (2016-)
  - No labels needed! Ex: SimCLR, DINO, lots more



He et al '16.

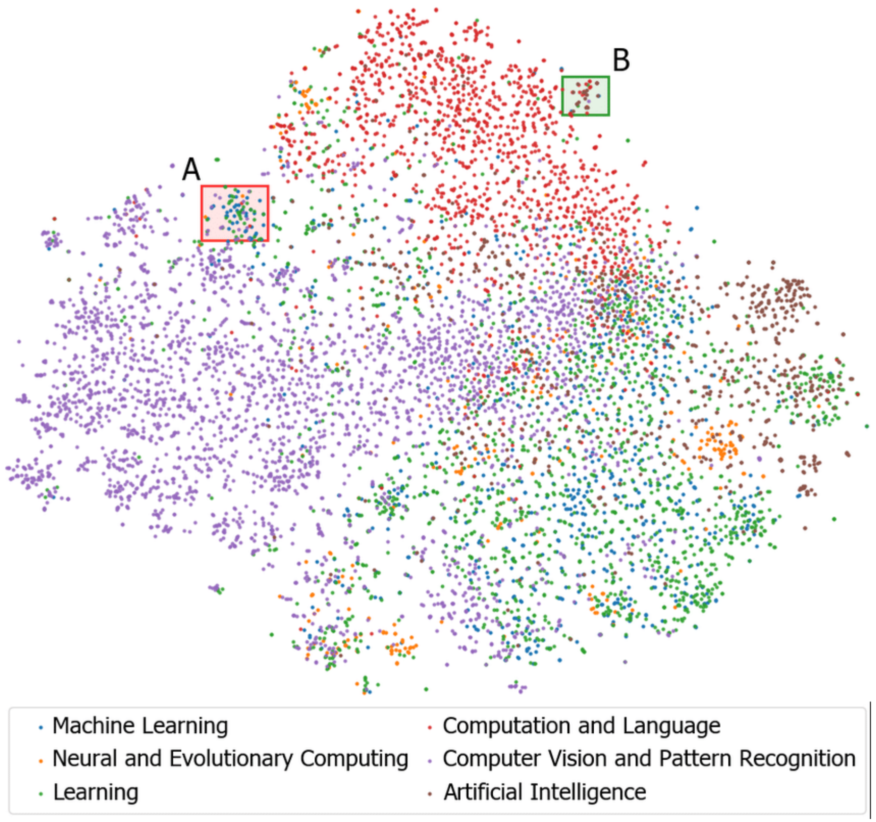
3. Word Embeddings and Language Models

Motivation: Deep learning advances – can they be applied to NLP?

Three areas of application:

1. General: ***word embeddings***
2. Specific: ***translation tasks***
3. Specific: ***language modeling tasks***

Embeddings for arXiv papers (6 ML categories)



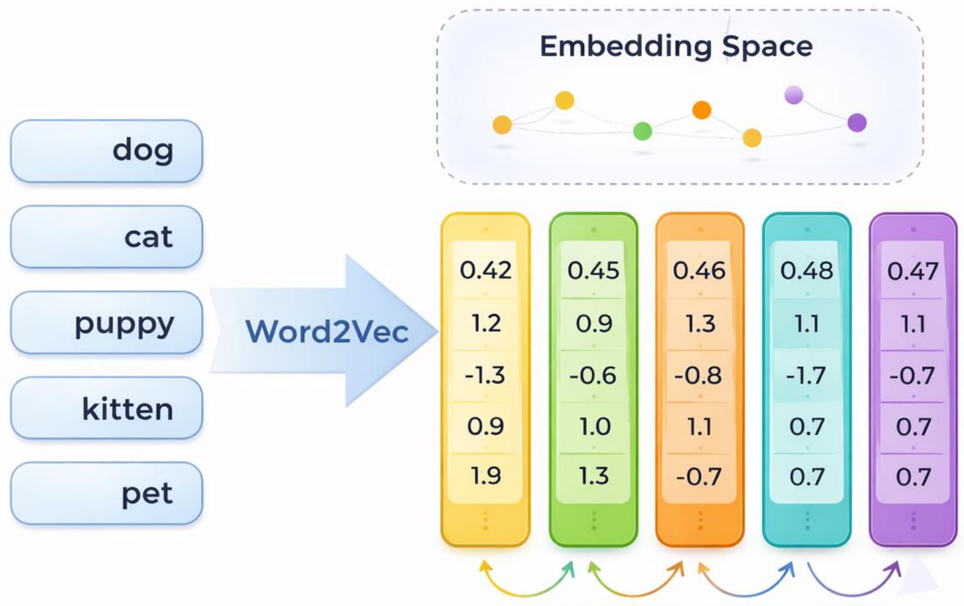
Lo et al '19.

3. Word Embeddings and Language Models

Motivation: Can we learn, in advance, ***structured representations*** of words?

- Then plug into language-specific neural networks (LSTMs, etc)

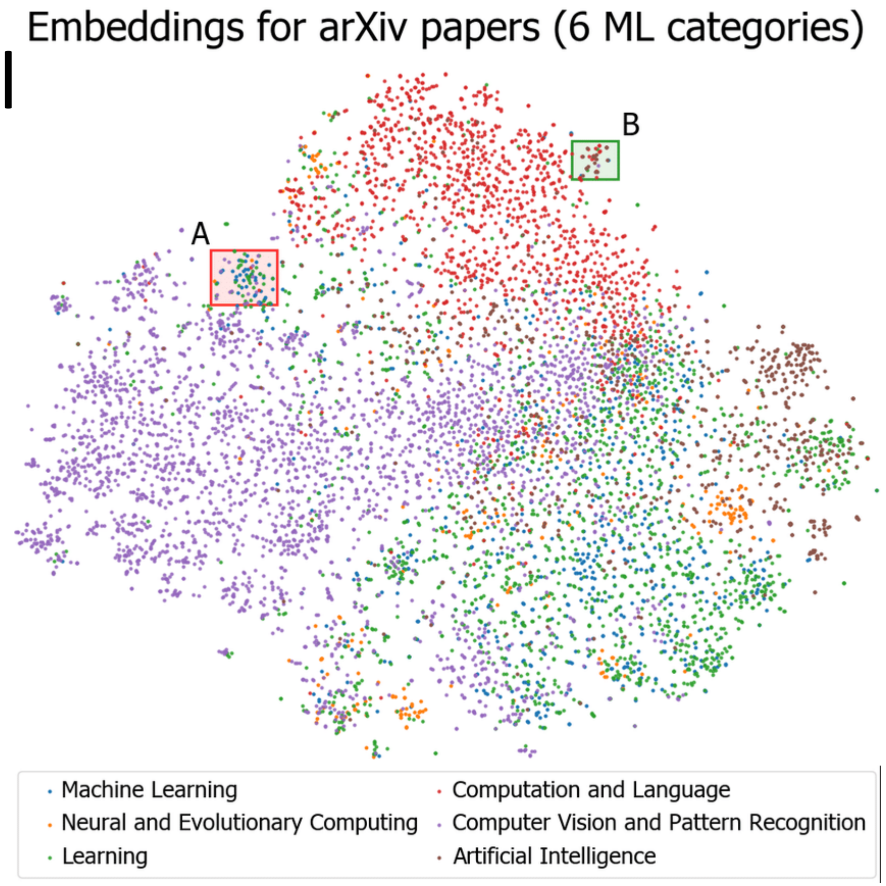
- First step: **word embeddings** (2013-2016): Glove, Word2Vec, etc.
  - Transform words into vectors
  - Can use as input to a neural network
  - A form of ***representation learning***



3. Word Embeddings and Language Models

Motivation: Can we learn, in advance, ***structured representations*** of words?

- Then plug into language-specific neural networks (LSTMs, etc)?
- First step: **word embeddings** (2013-2016): Glove, Word2Vec, etc.
- **Issues:** static. No context used for words like "bank" that have **multiple meanings**

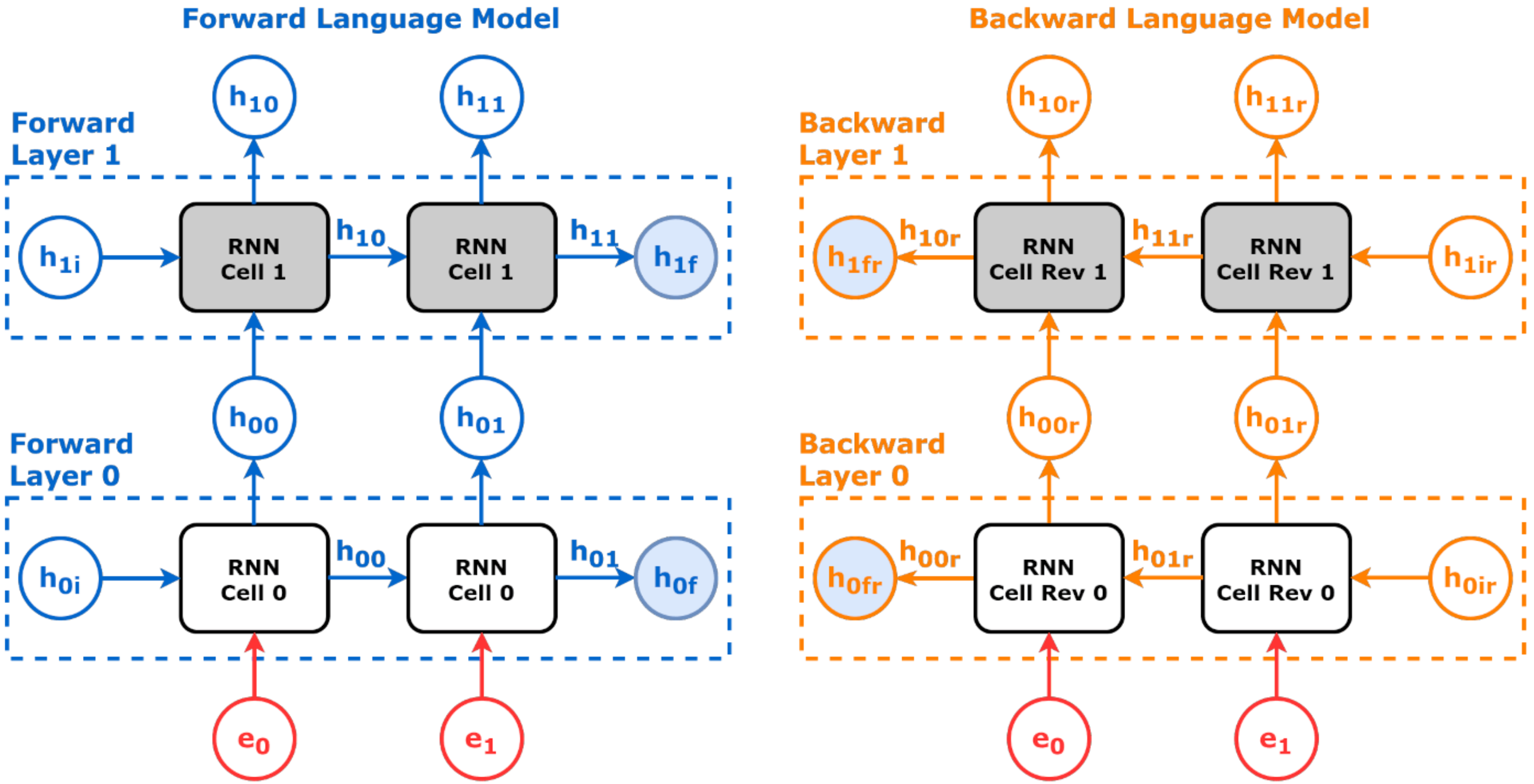


Lo et al '19.

3. Word Embeddings and Language Models

Solution: Contextual word embeddings

- **Idea:** Plug into a model to obtain the embedding, and include the context
- **ELMO embeddings:**



Godoy

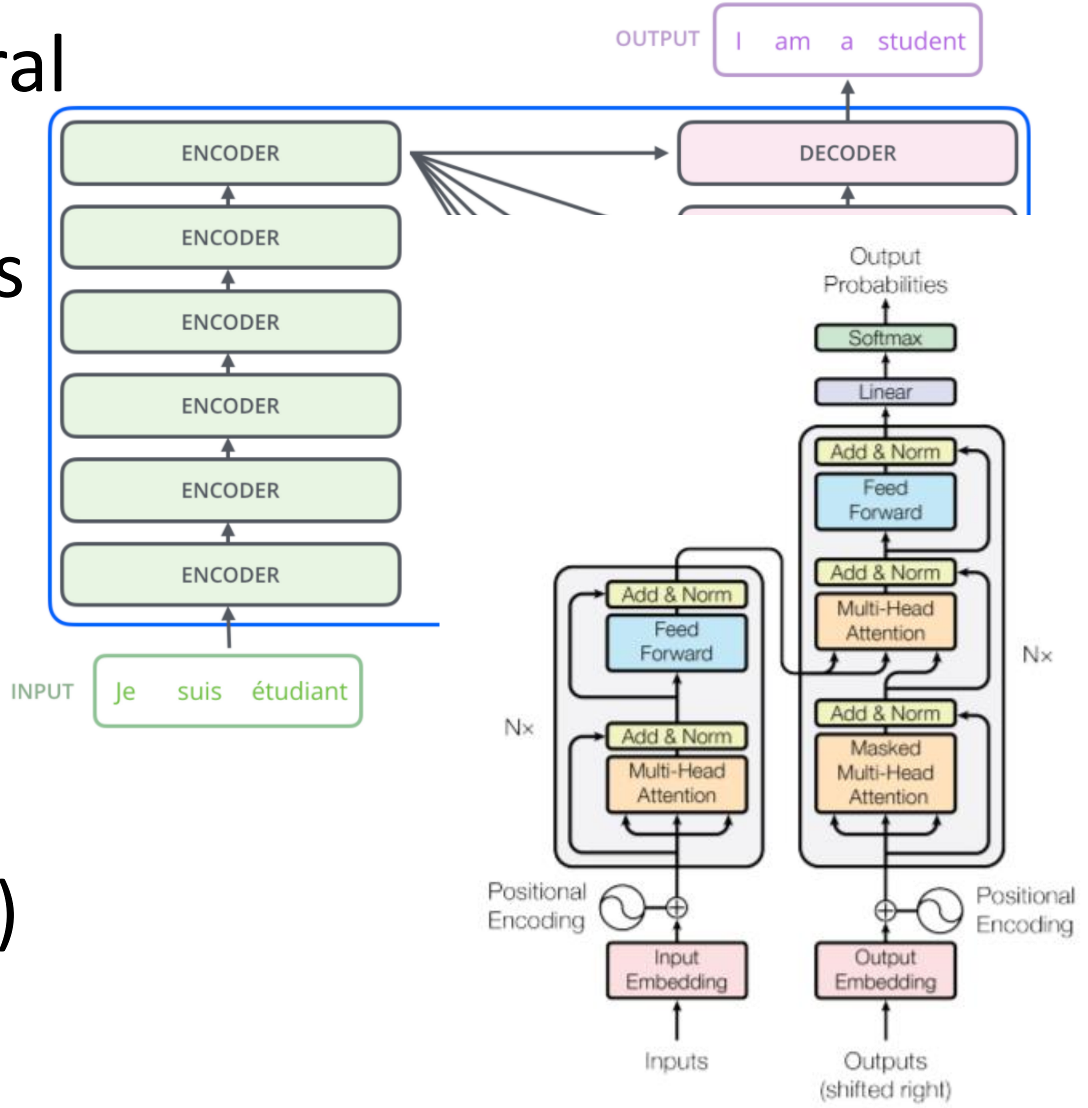
3. Word Embeddings and Language Models

So far: embeddings, which are general (whether static or contextual)

- What about deep learning advances for specific tasks?

Translation: critical task

- New architecture: ***Transformers*** (2017)
- Uses ideas around attention (2014-)



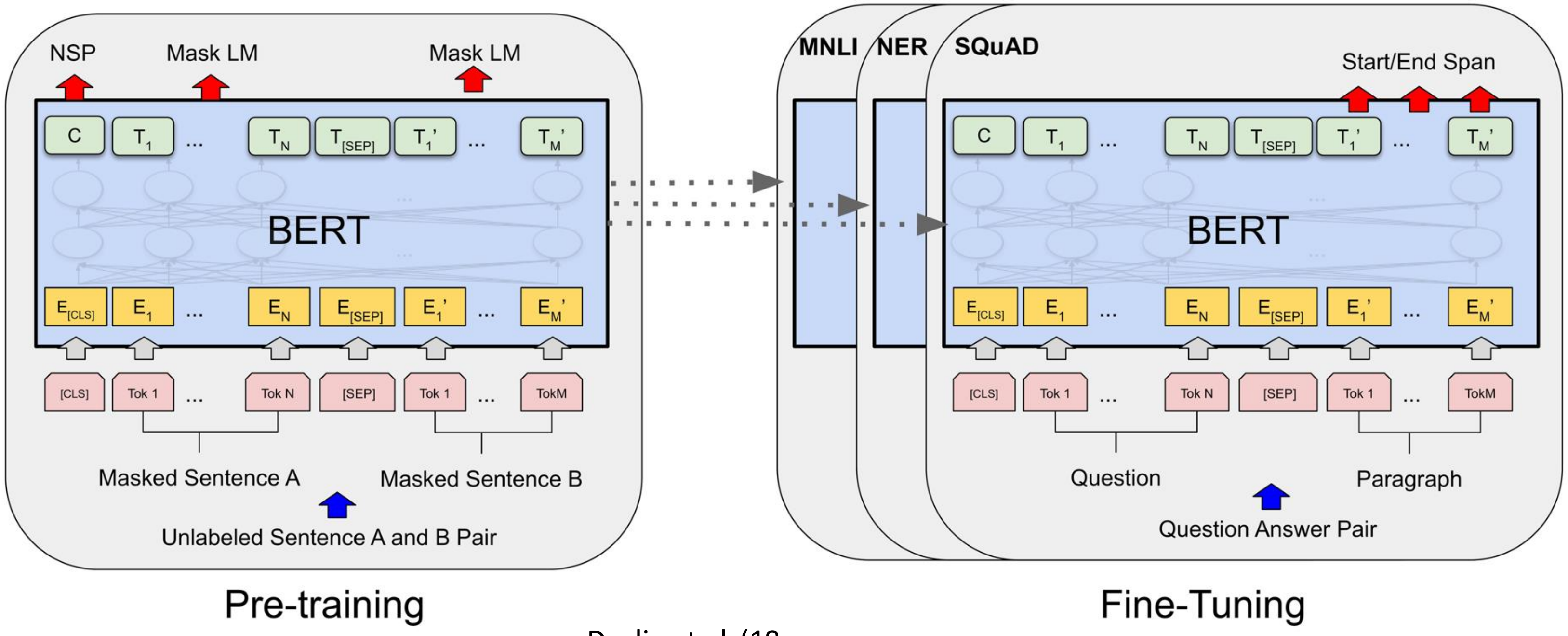
Vaswani et al. '17

3. Word Embeddings and Language Models

So far:

- Contextual embeddings (ELMO)
- Translation via Transformers architecture

Combine to ***BERT***, perhaps the first modern foundation model



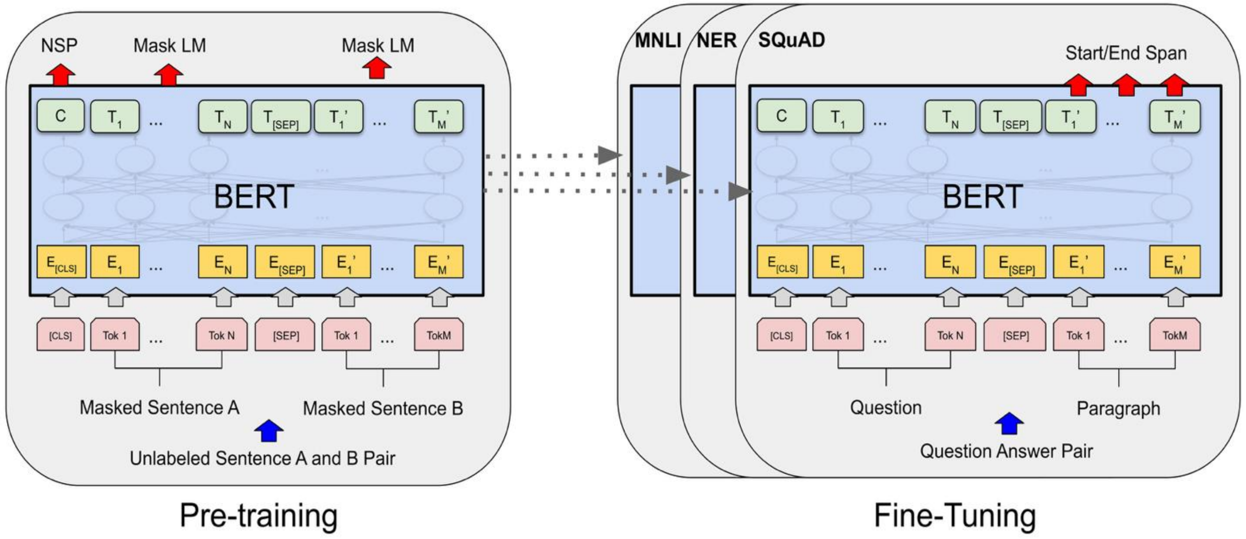
Devlin et al. '18

3. Word Embeddings and Language Models

Combine to ***BERT***, perhaps the first modern foundation model

- (1) **multitask**: same model can do QA, named entity recognition, etc.
- (2) **pretrained** (on Wikipedia!) and **fine-tuned** per task
- (3) works by producing **word embeddings**

Combines all **three trends**!



Devlin et al. '18

3. Word Embeddings and Language Models

What about language models?

- Similar idea: replace older architecture language models with new Transformer architecture
- Ex: ***GPT*** (***G***enerative ***P***retrained ***T***ransformer)
  - **Generative**: produces rich outputs (sentences and more, not just predictions)
  - **Pretrained**: as we've seen
  - **Transformer**: uses the Transformer architecture
- In all cases, pretrain on massive text corpora
  - All the way back to static embeddings, use all of Wikipedia!

Summary

Modern foundation models

- Build on old ideas about multitask learning,
- Are large-scale and pretrained on massive data, then specialized
  - Dating back to vision models from mid 2010s
- First heavily scaled for NLP applications, building on ideas on
  - Powerful contextual word embeddings
  - New architectures suitable for text (and beyond)