CSCI 334:
Principles of Programming Languages

Lecture 3: ML

Instructor: Dan Barowy

Williams

Topics

ML family of languages

F#

Your to-dos

1. Lab 1, **due Sunday 9/17** (partner lab)
2. Read *Advanced F#* (for Monday)

Announcements

- CS Colloquium **tomorrow @ 2:35pm in Wege Auditorium (TCL 123)**

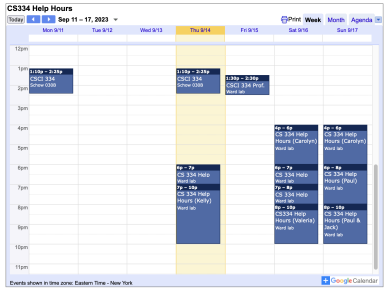
David Mimno (Cornell)

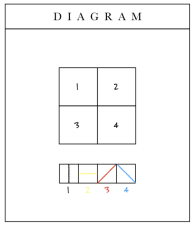
The data in data science: measuring the impact of data curation on large language model pretraining

Large language models like BERT and ChatGPT are fundamentally a reflection of the data used to train them. Putting together millions of documents from diverse sources requires innumerable choices. But because of the time and expense of the initial, general-purpose "pretraining" phase of model training, many of these choices are made heuristically without any systematic evidence-based justification. We train models to measure the effects of three common curation decisions: document age, quality and toxicity filtering, and data sources. We find that these choices have significant, noticeable effects that cannot be fully overcome by additional training.

Announcements

- TA Hours start tonight (don't let them get lonely)



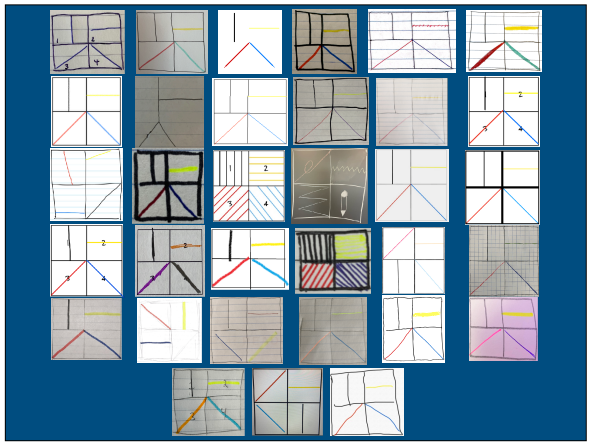


A square divided horizontally and vertically into four equal parts, each with a different color and line direction.

Red, yellow, blue, black pencil

Is this a program?

Recall the prompt I gave you before you did this assignment. It asked you to think about what makes a program a program. Is this a program?



Here are all of the renderings of those instructions that you produced. They're all similar but definitely not identical.

The one on the right is a "certified" Sol LeWitt.

So is our specification a program? Why or why not?

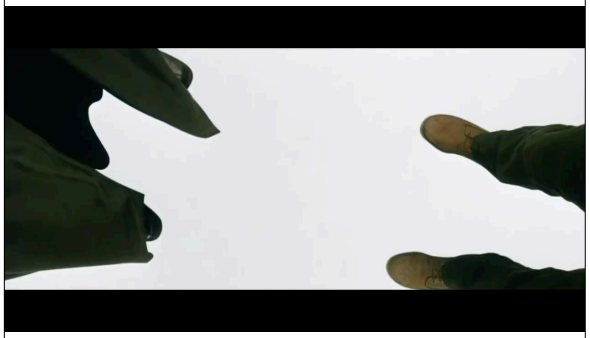
Activity

Today we are going to talk about a family of programming languages, called "ML." Note that this is a different "ML" than the term that refers to machine learning.

ML

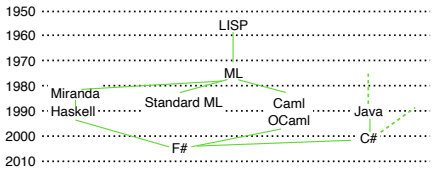
Today we are going to talk about a family of programming languages, called “ML.” Note that this is a different “ML” than the term that refers to machine learning.

“Free your mind”



Before we start, I want you to free your mind. Learning ML requires you to do some mind bending things sometimes. Be prepared not to get it right the first time. Be like Neo.

ML



Originally, ML was just a language. It was strongly influenced by LISP, which we will also touch on this semester. But many others were inspired by ML, and created new languages that added many new features. We will primarily spend our time learning F#, which is most directly influenced by Haskell, OCaml, and C#. I really love F#, and I hope you enjoy it too.

ML

- Dana Scott
- Logic of Computable Functions
- Can we automate proofs?
- Yes. Theorem proving is essentially a “search problem”!
- But proof search is “hard.” Many problems are NP-Complete.
- Works “in practice” with the right “tactics”

So where did ML come from? It was not born in a vacuum.

ML

- Robin Milner
- How to program tactics?
- A “meta language” is needed
- ML is born (1973)
- First impression upon encountering a computer: "Programming was not a very beautiful thing. I resolved I would never go near a computer in my life."

ML stands for “meta language.”

F#

- Don Syme
- ML is “more fun” than Java or C#.
- Can we use ML instead?
- F# is born (2010).

F# is a modern reinvention of ML for the .NET runtime produced by Microsoft.

Logical operators

Logical operators

operation	syntax
and	`&&`
not	`not`
equals	`=`
not equals	`<>`
inequalities	`<, >, <=, >=`

```
unit
```

Because in F# everything is an expression, we need a way to express the idea that a function may return nothing. For that, we have a special value called “unit.”

`unit` datatype

```
public static void main(String[] args) { … }
```

```
let main args = …
```

`unit` datatype

```
public static void main(String[] args) { … }
```

```
let main(args: string[]) = …
```

Remember: every expression must **return a value**.

A function **can't** return nothing.

`unit` datatype

```
public static void main(String[] args) { … }
```

```
let main(args: string[]) : unit = …
```

Therefore, "nothing" is a thing… called `unit`.

`unit` datatype

```
$ dotnet fsi

Microsoft (R) F# Interactive version 10.2.3 for F# 4.5
Copyright (c) Microsoft Corporation. All Rights Reserved.

For help type #help;;

> unit;;

  unit;;
  ^^^^

stdin(1,1): error FS0039: The value or constructor 'unit' is
not defined.

> ();;
val it : unit = ()

>
```

How does one obtain a value of `unit`? `()`

You can also `ignore`…

```
> let foo() = 2;;
val foo : unit -> int

> foo();;
val it : int = 2

> ignore (foo());;
val it : unit = ()

> foo() |> ignore;;
val it : unit = ()

>
```

"forward pipe" operator

```
<expr> |> <expr>

foo()  |> ignore
```

Another function called "ignore" allows you to "throw away" a value returned by a function. It replaces that value with unit. I am also showing my favorite F# operator here, which is called "forward pipe." If you've ever used pipes in the unix shell, forward pipe should be familiar.

By the way…

```
let main(args: string[]) : unit = …
```

I used this example before, but…

By the way…

```
let main(args: string[]) : int = …
```

… to be more precise, F# requires that main methods return int.

Primitives

Primitives

bool	sbyte
byte	int16
int	uint16
single	uint
double	int64
char	uint64
unit	nativeint
	unativeint
	decimal

† actually defined by the CLR

Recap & Next Class

Today:

History of ML

F#

Next class:

More F#