Datetime parsing

ONNX Roadmap presentation

Christian Bourjau

April 5, 2023

QuantCo

Outline

Use case

Use case

- Datetime strings are very common
- Feature engineering requires parsing to timestamps

What is the goal?

What is the goal?

Goal

- Introduce minimal datetime capabilities
- Parse a datetime string into a unix timestamp*
- Output is always UTC
- Customizable input format
- Well-defined error handling

Future expansions

Non-goals

What is the goal?

Goal

- Introduce minimal datetime capabilities
- Parse a datetime string into a unix timestamp*
- Output is always UTC
- Customizable input format
- Well-defined error handling

Future expansions

- Utility functions around timestamps. E.g. weekdays, year/month/day etc.

Non-goals

What is the goal?

Goal

- Introduce minimal datetime capabilities
- Parse a datetime string into a unix timestamp*
- Output is always UTC
- Customizable input format
- Well-defined error handling

Future expansions

- Utility functions around timestamps. E.g. weekdays, year/month/day etc.

Non-goals

- New "Datetime" data type
- Time zones

Existing standards

Existing standards

1998 C standard format codes

- C (i.e. `strptime`)
- Python
- Rust (defacto standard “chrono”)

Result types

- Common to use dedicated struct / type
- `strptime` returns `NULL` on failure
- Python raises an exception
- Chrono returns an error variant

Proposal

Proposal

`ParseDatetime(s, format, unit) -> timestamp`

Inputs

s	`tensor(string)`

Attributes

format	Format string like "`%Y %b %d`". May default to one of the iso standards.
unit	Unit of the returned time stamp. May be "second" (default), "millisecond", "microsecond", or "nanosecond".

Outputs

timestamp	`tensor(double)`. Unix timestamp (UTC) in the specified unit. NaN is returned if parsing failed.