This date-parsing model supports multi-language date/time detection and transformation to a pre-defined date format. It is based on highly accurate machine learning models. Given an input date and time, the model returns a transformed date in YYYY-MM-DD format with the attached time. Supported formats can be found at the end of this document. The model can also transform relative dates, for example '3 years and 2 months ago'. Current supported languages for both ago and dates are: English, Russian, Turkish, Spanish, Romanian, French, German, Dutch, Italian, Czech, Bulgarian, Polish, Swedish. Supported range of years are from 1975 to 2050.

The model also supports date and time extraction before transformation. The extraction can be made from any stream of words that represents text, meaningful or not, the date and time pattern will be searched. Additionally, the extraction can be made from any kind of token streams, like HTML. The date and time extraction are currently supported only for English language and absolute dates.

Return:

Date in standardized format: xxxx-xx-xx (yyyy-mm-dd)

Parameters:

- *dates* (type: list of string, default: []). Used to pass the list of dates as strings.
- *isdateonly* (type: bool, default: false). When true, returns the date with annulated time portion (00:00:00)
- *noyeardate* (type: bool default: false). When true, indicates that the model accepts only dates without the “year” portion. The missing year part is replaced with the current year. For supported formats, please refer to the end of the document.
- *resetnoyearcounter* (type: bool default: false. Works only when *noyeardate* is set to true). When true, indicates that the model is working in descending no-year date formats. Each successive date is compared to the previously encoded date. The expected new date is older than previous; therefore, the year portion is decremented if required. The year portion of the first date is the current year. For example:
    - ’29 July’ -> 2020-07-29
    - ’13 April’ -> 2020-04-13
    - ’2 November’ -> 2019-11-02 (year decremented)
    - ’11 August’ -> 2019-08-11

  all dates and previous years are memorized, until the first request occurrence where *resetnoyearcounter* is set to false, when year portion is set to the current year.

- *showtime*: (default: true). If set to false, only dates are returned without time portion.
- *dateformat*: (type: list on integers, default: all zeros, accepts: {0,1,2}). Used for handling ambiguous formats, where each element in this list corresponds to elements in the *dates* list. If all dates from the *dates* list are in default format, *dateformat* parameter is not required. For more details please refer to the next section.
- *ago*: (type: bool default: false). If true, model is working in ago/relative date format mode and accepts only relative dates. Parameters *isdateonly* and *showtime* can be used together this parameter if needed.
- *extractdates*: (default: false). If true, the dates are first extracted before transformation. All other parameters can be used along when this parameter is set to true.

Ambiguous formats:

Handling ambiguous formats like 04/09/08 requires using *dateformat* parameter.

- default format for xx/xx/xx is dd/mm/yy. For example: 04/07/09 translates to 2009-07-04
- if US-style mm/dd/yy or mm/dd/yyyy format is required, *dateformat* matching this element must be set to 1. For example: 04/07/09 translates to 2009-04-07.
- if yy/mm/dd format is required *dateformat* matching this element must be set to 2. For example: 04/07/09 translates to 2004-07-09
- yyyy/mm/dd does not require any additional *dateformat* parameter and it is transformed as-is to yyyy-mm-dd.
- yyyy/dd/mm or yy/dd/mm are not supported.
- example of usage:

```
{"dates" :["08.01.06 03:58:58", "29th July of 2020", "07.02.12 1:15pm", "04.06.01"], "dateformat":[1,0,2]}

returns:
[

  [
```

```
    "08.01.06 03:58:58",
    "2006-08-01 03:58:58"
  ],
  [
    "29th July of 2020",
    "2020-07-29 00:00:00"
  ],
  [
    "07.02.12 1:15pm",
    "2007-02-12 13:15:00"
  ],
  [
    "04.06.01",
    "2001-06-04 00:00:00"
  ]
]
```


because for the last “04.06.01”, *dateformat* parameter is not passed, it is transformed as default dd/mm/yy format.

Ago/Relative Mode:

Examples:

- 3 year’
- ‘year and 2 months ago’
- ‘1 week ago’
- ‘3 m 6 s’
- ‘3 months ago’

The “Ago” model has some specific behavior for ambiguous dates. In English, ‘m’ can sometimes represent ‘minute’ or ‘month’. When only one ‘m’ stands, for example ‘2 m ago’ it means ‘2 months ago’ by default. If we have ‘2 m and 30 sec ago’ this means ‘2 minutes and 30 seconds ago’, because seconds are much closer to minutes than months. On the other hand, if we have ‘3 m and 2 w ago’ this means ‘3 months and 2 weeks ago’, because in this case weeks are closer to months than minutes.

Ago mode also works with ‘Today’ and ‘Yesterday’ terms. For example, ‘Today @ 12:13 pm’ or ‘Yesterday at 11:13:45’.

Some general guidelines and rules of behavior:

- “number of something” in ago format must be a digit. Input cannot be ‘three months ago’; it must be ‘3 months ago’.
- ago date parts must be in descending or ascending order. Mixture is not allowed. For example, proper usage is:
  “2 months, 3 weeks and hour ago” OR “1 second and 3 minutes ago”
  Improper usage is something like:
  “1 second, 2 years and 2 months ago”, is not supported.
- having only ‘time’ portion, translates to ‘today @ time’.

example of usage on 2020-05-11:

```
{"dates" :["3 year", "year and 2 months ago", "1 week ago","3 m 6 s","3 months ago" ], "ago":true}
```

returns:


```
[
  [
    "3 year",
    "2017-05-12 13:52:16"
  ],
  [
    "year and 2 months ago",
    "2019-03-13 13:52:16"
  ],
  [
    "1 week ago",
```

```
      "2020-05-04 13:52:16"
    ],
    [
      "3 m 6 s",
      "2020-05-11 13:49:10"
    ],
    [
      "3 months ago",
      "2020-02-11 13:52:16"
    ]
]
```


Some examples of usage:

request: {"dates" :["23 April", "11/23", "Thu Jan 17", "October 23rd"], "noyeardate" : true}

returns:


```
[
    [
      "23 April",
      "2020-04-23 00:00:00"
    ],
    [
      "11/23",
      "2020-11-23 00:00:00"
    ],
    [
      "Thu Jan 17",
      "2020-01-17 00:00:00"
    ],
    [
      "October 23rd",
      "2020-10-23 00:00:00"
    ]
]
```


request: {"dates" :["29th July 2019", "Oct-1-03 1:13pm", "March 23, 2001", "September 1 03 23:12:11", "2004Jul30", "Jan Thu 17th 02"]}

returns:


```
[

    [
      "29th July 2019",
      "2019-07-29 00:00:00"
    ],
    [
      "Oct-1-03 1:13pm",
      "2003-10-01 13:13:00"
    ],
    [
      "March 23, 2001",
      "2001-03-23 00:00:00"
    ],
    [
      "September 1 03 23:12:11",
      "2003-09-01 23:12:11"
    ],
    [
```

```
    "2004Jul30",
    "2004-07-30 00:00:00"
  ],
  [
    "Jan Thu 17th 02",
    "2002-01-17 00:00:00"
  ]
]
```


request: {"dates" :["July 1 2020 @ 17:20:13", "13th Oct 2010 12:20am"], "isdateonly":true}

returns:


```
[
  [
    "July 1 2020 @ 17:20:13",
    "2020-07-01 00:00:00"
  ],
  [
    "13th Oct 2010 12:20am",
    "2010-10-13 00:00:00"
  ]
]
```


request: {"dates" :["1999 september 19th (Mon) @ 1:25pm"], "showtime":false}

returns:


```
[
  [
    "1999 september 19th (Mon) @ 1:25pm",
    "1999-09-19"
  ]
]
```


Extraction mode:

Dates along with the time portion will be extracted. If the time is the part of the complete date, then date and time concatenated will returned as on entity. Otherwise, both date and time are returned as separate entities. For example:

- *User registration date: 2019-12-31 @ 1:30pm*. Returns one entity [‘2019-12-31 @ 1:30pm’]
- *The time is 11:20am and the date is 29th June of 2017*. Returns two entities [’11:20am’ , ’ 29th June of 2017’]
- *22.04.2020 |15.03*. Returns two entities [’ 22.04.2020’ , ’ 15.03’]

request:

{"dates":["<tbody datetime=\"22-04-2020\" pubdate=\"\" onmouseover=\"\">3 weeks ago by EllasHoplite</tbody>"], "extractdates":true}

response:


```
[
  [
    "22-04-2020",
    "2020-04-22 00:00:00"
  ]
]
```

request:

{"dates" :["<section class=\"single\"> <article> <time class=\"pub-date\" datetime=\"2020-4-22 15:3\" pubdate=\"\"> <span class=\"date\">22.04.2020"], "extractdates":true}

response:


```
[
  [
    "2020-4-22 15:3",
    "2020-04-22 15:03:00"
  ],
  [
    "22.04.2020",
    "2020-04-22 00:00:00"
  ]
]
```


Some of the supported formats:

1	yyyy\mm\dd or yyyy/mm/dd or yyyy-mm-dd or yyyy.mm.dd or yyyy_mm_dd
2	mm-dd-yyyy or mm/dd/yyyy or mm\dd\yyyy or mm.dd.yyyy where yyyy could be yy
3	dd-mm-yyyy or dd/mm/yyyy or dd\mm\yyyy or dd.mm.yyyy where yyyy could be yy
4	mm/dd or mm\dd or mm-dd or mm.dd
5	March 23, 2001 or Jul 19, 2001
6	23rdXMarchX2001 or 2nd May, 02 or 5thXSeptX2003 where X is: '\' or '-' or '.' or '/' or ' ' or ','
7	01XAprX2002 or 23XMarchX2001 where X is: '\' or '-' or '.' or '/' or ' ' or ','
8	Tuesday, 01 January YYYY (YYYY could be 2 or 4 year digits)
9	ThuXJanX17 where X is: '\' or '-' or '.' or '/' or ' ' or ','
10	OctoberX23rd where X is: '\' or '-' or '.' or '/' or ' ' or ','
11	JulyX6thX1999 or JulyX6X1999 or SeptX6X1999 where X is: '\' or '-' or '.' or '/' or ' ' or ','
12	ThuXJanX17X2002 or ThuXJanX17X02 where X is: '\' or '-' or '.' or '/' or ' ' or ',' and Year could be 2 or 4 digits.
13	Jun. 19, YYYY OR Jun. 19th YYYY (YYYY could be 2 or 4 year digits)
14	AprilX30th, 2003 OR Wednesday, AprilX30th, 2003
15	2003-Oct-23 OR 2003-October-23
16	September 1 '03 or September 01 '03
17	Oct-01-2003 or Oct-1-03
18	2004Jul30 (No Separators)
19	Aug. 11/04 Where "." is optional

20	ddXmmXyyyy where X coud be \,-,\,.,/ and Might occur twice and yyyy could be yy. Ex: 29-08.-2004
21	JanXThuX17thX02 where X is: '\' or '-' or '.' or '/' or ' ' or ',' and Year could be 2 or 4 digits.
22	01XApr or 23XMarch where X is: '\' or '-' or '.' or '/' or ' ' or ',' (No Year)
23	yy\mm\dd or yy/mm/dd or yy-mm-dd or yy.mm.dd or yy_mm_dd
24	ThuX17XJan where X is: '\' or '-' or '.' or '/' or ' ' or ','
25	Jul 24th, '08
26	17th of September 2007 (Mon)
27	2008 August 29th
28	Thu 19th Mar '09