MultiProcessing Implementation in Texthero

August 24, 2020

0.1 Overview

In this notebook, we're showcasing the parallelization we implemented in Texthero, see PR #162. Short explanation of our reasoning and our approach:

We spent a lot of time looking at every possible options to speed up the library. Main findings:

1. `str.replace` and `s.apply(re.sub)` are equally fast (see #124) -> don't need to change this
2. dask requires users to use different data structures themselves, so it reduces complexity significantly for users. We believe only few users will profit from this when they're using datasets that do not fit into RAM, so it does not make sense to change all of texthero for this.
3. modin as drop-in replacement for pandas performed very bad for small to medium sized DFs
4. pandarallel did not work for `pd.apply` inside the library
5. Implementing multithreading through a decorator will not work in python; see every result when googling `python multiprocessing decorator`

The only way we see to parallelize things while still keeping a great User Experience and not slowing down users with small and medium sized datasets is the following:

Implement a function `helper.parallel` that handles parallelization for Pandas Series and use that to wrap our functions. Example:

```
# Example old approach:
@InputSeries(TextSeries)
def remove_round_brackets(s: TextSeries) -> TextSeries:
    """
    some nice docstring
    """
    return s.str.replace(r"\([^()]*\)", "")

# Example new approach: (same for user from the outside,
# a little more complicated (but fully parallelized!) on the inside)

def _remove_round_brackets(s: TextSeries) -> TextSeries:
    return s.str.replace(r"\([^()]*\)", "")

@InputSeries(TextSeries)
def remove_round_brackets(s: TextSeries) -> TextSeries:
    """
    some nice docstring
    """
    return parallel(s, _remove_round_brackets)
```

So we keep all functionality that is *not* parallelizable (like initializing patterns, downloading a spacy model, …) in a function `f` with a nice docstring that the user can see. This function at the end calls `helper.parallel(s, _f, other arguments)` where `_f` houses all the parallelizable functionality.

Here's the `helper.parallel` implementation, should be rather self-explanatory:

```
import pandas as pd
import multiprocessing as mp

cores = mp.cpu_count()
partitions = cores

MIN_LINES_FOR_PARALLELIZATION = 10000  # Set min. number of lines in Series to parallelize ->
PARALLELIZE = True  # Allows users to fully turn off parallelization for all dataset sizes


def parallel(s, func, *args, **kwargs):

    if len(s) < MIN_LINES_FOR_PARALLELIZATION or not PARALLELIZE:
        # Execute as usual.
        return func(s, *args, **kwargs)

    else:
        # Execute in parallel.

        # Split the data up into batches.
        s_split = np.array_split(s, partitions)

        # Open threadpool.
        pool = mp.Pool(cores)
        # Execute in parallel and concat results (order is kept).
        s_result = pd.concat(
            pool.map(functools.partial(func, *args, **kwargs), s_split)
        )

        pool.close()
        pool.join()

        return s_result
```

We know that this does introduce some added complexity for developers in the `nlp` and `preprocessing` modules, but we believe that it's the best solution for users (automatic parallelization and same user experience as before) and with some more developer documentation, other contributors should not have any problems.

Of course, lots of tests are added to `test_helper.py`.

0.2 For Developers

As described above, in the `preprocessing` and `nlp` modules, there are slight changes that have to be made when implementing a new function. New approach should be:

1. write the new function as usual, e.g. `find_keywords`
2. at the end of the function body, there is probably something you `apply` to the whole input series (e.g. with `s.str.replace` or `pd.apply(s, ...)`, just some part that takes as input a series and returns a series, and where the output for row x does not depend on row y. Move this part to a new `_find_keywords` and call `parallel(s, _find_keywords, all other arguments that _find_keywords needs)` at the end of `find_keywords`. Keep all non-parallel parts (e.g. loading a spaCy model) in `find_keywords`.
3. add a test for multiprocessing in `test_helpers`
4. that's it, your function will now automatically be parallelized when being run on big datasets.

All of this should be easy after having a look at some `preprocessing` source code.

0.3 For Users

Changes for users: - when they have a Series with more than `hero.config.MIN_LINES_FOR_PARALLELIZATION`, executing is parallelized for almost all preprocessing and nlp functions - they can change `hero.config.MIN_LINES_FOR_PARALLELIZATION` (if they want to also parallelize when they have fewer rows for example), and set `hero.config.PARALLELIZE = False` if they want to turn of parallelization completely.

We'll explain this in the relevant documentation parts.

[1]:
```
!pip install git+https://github.com/SummerOfCode-NoHate/
↪texthero@decorator_for_parallelization
```

```
Collecting git+https://github.com/SummerOfCode-
NoHate/texthero@decorator_for_parallelization
  Cloning https://github.com/SummerOfCode-NoHate/texthero (to revision
decorator_for_parallelization) to
/private/var/folders/ff/v8q71qfn4hbdkzbpmf28ymsr0000gn/T/pip-req-build-42_k0x_2
  Running command git clone -q https://github.com/SummerOfCode-NoHate/texthero
/private/var/folders/ff/v8q71qfn4hbdkzbpmf28ymsr0000gn/T/pip-req-build-42_k0x_2
  Running command git checkout -b decorator_for_parallelization --track
origin/decorator_for_parallelization
  Switched to a new branch 'decorator_for_parallelization'
  Branch 'decorator_for_parallelization' set up to track remote branch
'decorator_for_parallelization' from 'origin'.
Requirement already satisfied (use --upgrade to upgrade): texthero==1.0.9 from
git+https://github.com/SummerOfCode-
NoHate/texthero@decorator_for_parallelization in
/opt/anaconda3/lib/python3.7/site-packages
Requirement already satisfied: numpy>=1.17 in /opt/anaconda3/lib/python3.7/site-
packages (from texthero==1.0.9) (1.18.1)
Requirement already satisfied: scikit-learn>=0.22 in
/opt/anaconda3/lib/python3.7/site-packages (from texthero==1.0.9) (0.22.1)
```

```
Requirement already satisfied: spacy>=2.2.2 in
/opt/anaconda3/lib/python3.7/site-packages (from texthero==1.0.9) (2.3.2)
Requirement already satisfied: tqdm>=4.3 in /opt/anaconda3/lib/python3.7/site-
packages (from texthero==1.0.9) (4.42.1)
Requirement already satisfied: nltk>=3.3 in /opt/anaconda3/lib/python3.7/site-
packages (from texthero==1.0.9) (3.4.5)
Requirement already satisfied: plotly>=4.2.0 in
/opt/anaconda3/lib/python3.7/site-packages (from texthero==1.0.9) (4.9.0)
Requirement already satisfied: pandas>=1.0.2 in
/opt/anaconda3/lib/python3.7/site-packages (from texthero==1.0.9) (1.1.1)
Requirement already satisfied: wordcloud>=1.5.0 in
/opt/anaconda3/lib/python3.7/site-packages (from texthero==1.0.9) (1.7.0)
Requirement already satisfied: unidecode>=1.1.1 in
/opt/anaconda3/lib/python3.7/site-packages (from texthero==1.0.9) (1.1.1)
Requirement already satisfied: gensim>=3.6.0 in
/opt/anaconda3/lib/python3.7/site-packages (from texthero==1.0.9) (3.8.3)
Requirement already satisfied: matplotlib>=3.1.0 in
/opt/anaconda3/lib/python3.7/site-packages (from texthero==1.0.9) (3.1.3)
Requirement already satisfied: scipy>=0.17.0 in
/opt/anaconda3/lib/python3.7/site-packages (from scikit-
learn>=0.22->texthero==1.0.9) (1.4.1)
Requirement already satisfied: joblib>=0.11 in
/opt/anaconda3/lib/python3.7/site-packages (from scikit-
learn>=0.22->texthero==1.0.9) (0.14.1)
Requirement already satisfied: srsly<1.1.0,>=1.0.2 in
/opt/anaconda3/lib/python3.7/site-packages (from spacy>=2.2.2->texthero==1.0.9)
(1.0.2)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in
/opt/anaconda3/lib/python3.7/site-packages (from spacy>=2.2.2->texthero==1.0.9)
(2.0.3)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in
/opt/anaconda3/lib/python3.7/site-packages (from spacy>=2.2.2->texthero==1.0.9)
(1.0.2)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in
/opt/anaconda3/lib/python3.7/site-packages (from spacy>=2.2.2->texthero==1.0.9)
(3.0.2)
Requirement already satisfied: wasabi<1.1.0,>=0.4.0 in
/opt/anaconda3/lib/python3.7/site-packages (from spacy>=2.2.2->texthero==1.0.9)
(0.7.1)
Requirement already satisfied: blis<0.5.0,>=0.4.0 in
/opt/anaconda3/lib/python3.7/site-packages (from spacy>=2.2.2->texthero==1.0.9)
(0.4.1)
Requirement already satisfied: catalogue<1.1.0,>=0.0.7 in
/opt/anaconda3/lib/python3.7/site-packages (from spacy>=2.2.2->texthero==1.0.9)
(1.0.0)
Requirement already satisfied: plac<1.2.0,>=0.9.6 in
/opt/anaconda3/lib/python3.7/site-packages (from spacy>=2.2.2->texthero==1.0.9)
(1.1.3)
```

```
Requirement already satisfied: setuptools in /opt/anaconda3/lib/python3.7/site-
packages (from spacy>=2.2.2->texthero==1.0.9) (46.0.0.post20200309)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in
/opt/anaconda3/lib/python3.7/site-packages (from spacy>=2.2.2->texthero==1.0.9)
(2.22.0)
Requirement already satisfied: thinc==7.4.1 in
/opt/anaconda3/lib/python3.7/site-packages (from spacy>=2.2.2->texthero==1.0.9)
(7.4.1)
Requirement already satisfied: six in /opt/anaconda3/lib/python3.7/site-packages
(from nltk>=3.3->texthero==1.0.9) (1.14.0)
Requirement already satisfied: retrying>=1.3.3 in
/opt/anaconda3/lib/python3.7/site-packages (from plotly>=4.2.0->texthero==1.0.9)
(1.3.3)
Requirement already satisfied: python-dateutil>=2.7.3 in
/opt/anaconda3/lib/python3.7/site-packages (from pandas>=1.0.2->texthero==1.0.9)
(2.8.1)
Requirement already satisfied: pytz>=2017.2 in
/opt/anaconda3/lib/python3.7/site-packages (from pandas>=1.0.2->texthero==1.0.9)
(2019.3)
Requirement already satisfied: pillow in /opt/anaconda3/lib/python3.7/site-
packages (from wordcloud>=1.5.0->texthero==1.0.9) (7.0.0)
Requirement already satisfied: smart-open>=1.8.1 in
/opt/anaconda3/lib/python3.7/site-packages (from gensim>=3.6.0->texthero==1.0.9)
(2.1.0)
Requirement already satisfied: cycler>=0.10 in
/opt/anaconda3/lib/python3.7/site-packages (from
matplotlib>=3.1.0->texthero==1.0.9) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/opt/anaconda3/lib/python3.7/site-packages (from
matplotlib>=3.1.0->texthero==1.0.9) (1.1.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
/opt/anaconda3/lib/python3.7/site-packages (from
matplotlib>=3.1.0->texthero==1.0.9) (2.4.6)
Requirement already satisfied: importlib-metadata>=0.20; python_version < "3.8"
in /opt/anaconda3/lib/python3.7/site-packages (from
catalogue<1.1.0,>=0.0.7->spacy>=2.2.2->texthero==1.0.9) (1.5.0)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in
/opt/anaconda3/lib/python3.7/site-packages (from
requests<3.0.0,>=2.13.0->spacy>=2.2.2->texthero==1.0.9) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in
/opt/anaconda3/lib/python3.7/site-packages (from
requests<3.0.0,>=2.13.0->spacy>=2.2.2->texthero==1.0.9) (2019.11.28)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/opt/anaconda3/lib/python3.7/site-packages (from
requests<3.0.0,>=2.13.0->spacy>=2.2.2->texthero==1.0.9) (1.25.8)
Requirement already satisfied: idna<2.9,>=2.5 in
/opt/anaconda3/lib/python3.7/site-packages (from
requests<3.0.0,>=2.13.0->spacy>=2.2.2->texthero==1.0.9) (2.8)
```

```
Requirement already satisfied: boto3 in /opt/anaconda3/lib/python3.7/site-
packages (from smart-open>=1.8.1->gensim>=3.6.0->texthero==1.0.9) (1.14.23)
Requirement already satisfied: boto in /opt/anaconda3/lib/python3.7/site-
packages (from smart-open>=1.8.1->gensim>=3.6.0->texthero==1.0.9) (2.49.0)
Requirement already satisfied: zipp>=0.5 in /opt/anaconda3/lib/python3.7/site-
packages (from importlib-metadata>=0.20; python_version <
"3.8"->catalogue<1.1.0,>=0.0.7->spacy>=2.2.2->texthero==1.0.9) (2.2.0)
Requirement already satisfied: botocore<1.18.0,>=1.17.23 in
/opt/anaconda3/lib/python3.7/site-packages (from boto3->smart-
open>=1.8.1->gensim>=3.6.0->texthero==1.0.9) (1.17.23)
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in
/opt/anaconda3/lib/python3.7/site-packages (from boto3->smart-
open>=1.8.1->gensim>=3.6.0->texthero==1.0.9) (0.10.0)
Requirement already satisfied: s3transfer<0.4.0,>=0.3.0 in
/opt/anaconda3/lib/python3.7/site-packages (from boto3->smart-
open>=1.8.1->gensim>=3.6.0->texthero==1.0.9) (0.3.3)
Requirement already satisfied: docutils<0.16,>=0.10 in
/opt/anaconda3/lib/python3.7/site-packages (from
botocore<1.18.0,>=1.17.23->boto3->smart-
open>=1.8.1->gensim>=3.6.0->texthero==1.0.9) (0.15.2)
Building wheels for collected packages: texthero
  Building wheel for texthero (setup.py) … done
  Created wheel for texthero: filename=texthero-1.0.9-py3-none-any.whl
size=43898
sha256=14cdb6375fbebb0269446cc23aade456dc8197581db83ec752c366994adf8127
  Stored in directory:
/private/var/folders/ff/v8q71qfn4hbdkzbpmf28ymsr0000gn/T/pip-ephem-wheel-cache-
iie4rql9/wheels/e7/d1/60/88628de1662df5ddf78097e355a7bea59be0a1e213f5f636e2
Successfully built texthero
```

[1]:
```python
import texthero as hero
import numpy as np
import pandas as pd
```

[2]:
```python
# init dataset for speed comparison

data_small = pd.read_csv("https://raw.githubusercontent.com/jbesomi/texthero/
 ↪master/dataset/bbcsport.csv")
data_big = pd.DataFrame([text for _ in range(100) for text in␣
 ↪data_small["text"].values], columns=["text"])
print("Big dataset has {} texts".format(len(data_big)))
```

```
Big dataset has 73700 texts
```

We can see below that with parallelization, we get an around 3x improvement with 4 cores / 8 threads, so pretty good!

[4]:
```
# turn of multithreading in texthero and take time of the single thread.
# This is exactly like before, so this is the current texthero speed.

hero.config.PARALLELIZE = False

%time hero.remove_diacritics(data_big["text"])
```

```
CPU times: user 1min 21s, sys: 419 ms, total: 1min 22s
Wall time: 1min 22s
```

[4]:
```
0         Claxton hunting first major medal\n\nBritish h…
1         O'Sullivan could run in Worlds\n\nSonia O'Sull…
2         Greene sets sights on world title\n\nMaurice G…
3         IAAF launches fight against drugs\n\nThe IAAF …
4         Dibaba breaks 5,000m world record\n\nEthiopia'…
                                 …
368495    Agassi into second round in Dubai\n\nFourth se…
368496    Mauresmo fights back to win title\n\nWorld num…
368497    Federer wins title in Rotterdam\n\nWorld numbe…
368498    GB players warned over security\n\nBritain's D…
368499    Sharapova overcomes tough Molik\n\nWimbledon c…
Name: text, Length: 368500, dtype: object
```

[5]:
```
# turn on multithreading in texthero and take time of multi threads
hero.config.PARALLELIZE = True

%time hero.remove_diacritics(data_big["text"])
```

```
CPU times: user 830 ms, sys: 1.1 s, total: 1.93 s
Wall time: 28.7 s
```

[5]:
```
0         Claxton hunting first major medal\n\nBritish h…
1         O'Sullivan could run in Worlds\n\nSonia O'Sull…
2         Greene sets sights on world title\n\nMaurice G…
3         IAAF launches fight against drugs\n\nThe IAAF …
4         Dibaba breaks 5,000m world record\n\nEthiopia'…
                                 …
368495    Agassi into second round in Dubai\n\nFourth se…
368496    Mauresmo fights back to win title\n\nWorld num…
368497    Federer wins title in Rotterdam\n\nWorld numbe…
368498    GB players warned over security\n\nBritain's D…
368499    Sharapova overcomes tough Molik\n\nWimbledon c…
Name: text, Length: 368500, dtype: object
```

Here we compare our multithreading implementation with spaCy's internal one. As you can see, our method is 2x faster than the spacy one. However when executing the notebook, the spacy algorithm would only use half of the available threads, whereas our method uses all available ones. However this might just be a bug on a Mac.

[3]:
```
# Demonstration that spaCy multithreading is slower than our implementation
# our implementation
%time hero.pos_tag(data_big["text"])
```

```
CPU times: user 15.7 s, sys: 5.91 s, total: 21.6 s
Wall time: 3min 42s
```

[3]:
```
0        [(Claxton, ADJ, JJ, 0, 7), (hunting, VERB, VBG…
1        [(O'Sullivan, PROPN, NNP, 0, 10), (could, VERB…
2        [(Greene, NOUN, NN, 0, 6), (sets, VERB, VBZ, 7…
3        [(IAAF, PROPN, NNP, 0, 4), (launches, NOUN, NN…
4        [(Dibaba, PROPN, NNP, 0, 6), (breaks, NOUN, NN…
                               …
73695    [(Agassi, PROPN, NNP, 0, 6), (into, ADP, IN, 7…
73696    [(Mauresmo, PROPN, NNP, 0, 8), (fights, VERB, …
73697    [(Federer, NOUN, NN, 0, 7), (wins, VERB, VBZ, …
73698    [(GB, PROPN, NNP, 0, 2), (players, NOUN, NNS, …
73699    [(Sharapova, PROPN, NNP, 0, 9), (overcomes, VE…
Length: 73700, dtype: object
```

[11]:
```
# spaCy mulitprocessing enabled
import spacy

def pos_tag_with_spacys_internal_multithreading(s, nlp) -> pd.Series:

    pos_tags = []

    for doc in nlp.pipe(s.astype("unicode").values, batch_size=32, n_process=8):
        pos_tags.append(
            [
                (token.text, token.pos_, token.tag_, token.idx, token.idx +␣
↪len(token))
                for token in doc
            ]
        )

    return pd.Series(pos_tags, index=s.index)

nlp = spacy.load("en_core_web_sm", disable=["parser", "ner"])
```

[12]:
```
%time pos_tag_with_spacys_internal_multithreading(data_big["text"], nlp)
```

```
CPU times: user 7min 30s, sys: 55.1 s, total: 8min 25s
Wall time: 8min 33s
```

[12]:
```
0        [(Claxton, ADJ, JJ, 0, 7), (hunting, VERB, VBG…
1        [(O'Sullivan, PROPN, NNP, 0, 10), (could, VERB…
2        [(Greene, NOUN, NN, 0, 6), (sets, VERB, VBZ, 7…
3        [(IAAF, PROPN, NNP, 0, 4), (launches, NOUN, NN…
4        [(Dibaba, PROPN, NNP, 0, 6), (breaks, NOUN, NN…
                               …
73695    [(Agassi, PROPN, NNP, 0, 6), (into, ADP, IN, 7…
73696    [(Mauresmo, PROPN, NNP, 0, 8), (fights, VERB, …
73697    [(Federer, NOUN, NN, 0, 7), (wins, VERB, VBZ, …
73698    [(GB, PROPN, NNP, 0, 2), (players, NOUN, NNS, …
73699    [(Sharapova, PROPN, NNP, 0, 9), (overcomes, VE…
Length: 73700, dtype: object
```