Noost Documentation

Release 0.3.3

Cole Thienes, Jack Pertschuk

Feb 28, 2020

Elasticsearch

NBoost is a scalable, search-engine-boosting platform for developing and deploying state-of-the-art models to improve the relevance of search results.

Nboost leverages finetuned models to produce domain-specific neural search engines. The platform can also improve other downstream tasks requiring ranked input, such as question answering.

<table border="1" class="docutils"> <thead> <tr> <th></th> </tr> </thead> <tbody> <tr> <td></td> </tr> </tbody> </table>

CHAPTER 1

Setting up NBoost for Elasticsearch

In this example we will set up a proxy to sit in between the client and Elasticsearch and boost the results!

1.1 Preliminaries

1. Install NBoost for Pytorch.

```
pip install nboost[pt]
```

2. Set up an Elasticsearch Server

   If you already have an Elasticsearch server, you can skip this step!

   If you don't have Elasticsearch, not to worry! You can set up a local Elasticsearch cluster by using docker. First, get the ES image by running:

```
docker pull elasticsearch:7.4.2
```

   Once you have the image, you can run an Elasticsearch server via:

```
docker run -d -p 9200:9200 -p 9300:9300 -e "discovery.type=single-node"␣
↪elasticsearch:7.4.2
```

3. Index some data

   NBoost has a handy indexing tool built in (`nboost-index`). For demonstration purposes, will be indexing a set of passages about traveling and hotels. You can add the index to your Elasticsearch server by running:

   `travel.csv` comes with NBoost

```
nboost-index --file travel.csv --name travel --delim ,
```

1.2 Deploying the proxy

Now we're ready to deploy our Neural Proxy! There are three ways to configure NBoost.

1. Via command line.

   On the command line, we can run:

```
nboost                                  \
    --uhost localhost                   \
    --uport 9200                        \
    --search_route "/<index>/_search"   \
    --query_path url.query.q            \
    --topk_path url.query.size          \
    --default_topk 10                   \
    --topn 50                           \
    --choices_path body.hits.hits       \
    --cvalues_path _source.passage
```

   The `--uhost` and `--uport` should be the same as the Elasticsearch server above! Uhost and uport are short for upstream-host and upstream-port (referring to the upstream server).

   Now let's test it out! Hit the Elasticsearch with:

```
curl "http://localhost:8000/travel/_search?pretty&q=passage:vegas&size=2"
```

2. Via json.

   On the command line, let's run:

```
nboost --search_route "/<index>/_search"
```

   In a python script, we can run:

```
import requests
from pprint import pprint

response = requests.get(
    url='http://localhost:8000/travel/_search',
    json={
        'nboost': {
            'uhost': 'localhost',
            'uport': 9200,
            'query_path': 'body.query.match.passage',
            'topk_path': 'body.size',
            'default_topk': 10,
            'topn': 50,
            'choices_path': 'body.hits.hits',
            'cvalues_path': '_source.passage'
        },
        'size': 2,
        'query': {
            'match': {'passage': 'I want a Louisiana hotel with a pool'}
        }
    }
)

pprint(response.json())
```

3. Via query params.

   On the command line, let's run:

```
nboost --search_route "/<index>/_search"
```

   In a python script, we can run:

```
import requests
from pprint import pprint

response = requests.get(
    url='http://localhost:8000/travel/_search',
    params={
        'uhost': 'localhost',
        'uport': 9200,
        'query_path': 'url.query.q',
        'topk_path': 'url.query.size',
        'default_topk': 10,
        'topn': 50,
        'choices_path': 'body.hits.hits',
        'cvalues_path': '_source.passage',
        'q': 'passage:I want a vegas hotel with a pool',
        'size': 2
    }
)

pprint(response.json())
```

No matter how we configure NBoost, if the Elasticsearch result has the `nboost` tag in it, congratulations it's working!

CHAPTER 2

Setting up NBoost for the Bing Search API

The Bing Search API offers a handy REST service that we can use to get preliminary results before we rerank with NBoost, or for usage with question answering. We will walk through an example of how to set up a Bing-Powered QA system.

2.1 Preliminaries

1. Get your Bing API key.
2. Check out the Bing DSL docs for background on how to normally use the API.

2.2 Deploying the Proxy

Just like the Elasticsearch tutorial, we will go through the three ways to configure NBoost.

1. Via Command Line:

   On the command line, let's run:

```
nboost                                  \
    --uhost api.cognitive.microsoft.com \
    --uport 443                         \
    --ussl True                         \
    --topn 20                           \
    --search_route /bing/v7.0/search    \
    --query_path url.query.q            \
    --topk_path url.query.count         \
    --default_topk 10                   \
    --choices_path body.webPages.value  \
    --cvalues_path snippet              \
    --qa True                           \
    --qa_model PtDistilBertQAModelPlugin
```

Then we can query NBoost.

```
curl -H "Ocp-Apim-Subscription-Key: <BING API KEY>" localhost:8000/bing/v7.0/
↪search?q=how+old+is+obama&count=1&responseFilter=Webpages
```

2. Via json.

   On the command line, let's run:

```
nboost --search_route /bing/v7.0/search --qa True --qa_model␣
↪PtDistilBertQAModelPlugin
```

   In a python script, we can run:

```
import requests
from pprint import pprint

response = requests.get(
    url='http://localhost:8000/bing/v7.0/search',
    headers={'Ocp-Apim-Subscription-Key': '<BING API KEY>'},
    params={'q': 'how old is obama', 'responseFilter': 'Webpages'},
    json={
        'nboost': {
            'uhost': 'api.cognitive.microsoft.com',
            'uport': 443,
            'topn': 20,
            'query_path': 'url.query.q',
            'topk_path': 'url.query.count',
            'default_topk': 10,
            'choices_path': 'body.webPages.value',
            'cvalues_path': 'snippet'
        }
    }
)

pprint(response.json())
```

3. Via query params.

   On the command line, let's run:

```
nboost --search_route /bing/v7.0/search --qa True --qa_model␣
↪PtDistilBertQAModelPlugin
```

   In a python script, we can run:

```
import requests
from pprint import pprint

response = requests.get(
   url='http://localhost:8000/bing/v7.0/search',
   headers={'Ocp-Apim-Subscription-Key': '<BING API KEY>'},
   params={
       'q': 'how old is obama',
       'responseFilter': 'Webpages',
       'uhost': 'api.cognitive.microsoft.com',
       'uport': 443,
       'topn': 20,
       'query_path': 'url.query.q',
```

(continues on next page)

(continued from previous page)

```
        'topk_path': 'url.query.count',
        'default_topk': 10,
        'choices_path': 'body.webPages.value',
        'cvalues_path': 'snippet'
    }
)

pprint(response.json())
```

No matter how we query, the json response will look like this:

```
{"_type": "SearchResponse",
 "nboost": {"answer_start_pos": 115,
            "answer_stop_pos": 127,
            "answer_text": "53 years old."},
 "queryContext": {"originalQuery": "how old is obama"},
 "webPages": {"totalEstimatedMatches": 81700000,
              "value": [{"about": [{"name": "Barack Obama"}],
                         "dateLastCrawled": "2019-11-28T01:11:00.0000000Z",
                         "displayUrl": "https://www.answers.com/Q/How_old_is_Barack_
↪Obama",
                         "id": "https://api.cognitive.microsoft.com/api/v7/#WebPages.
↪11",
                         "isFamilyFriendly": true,
                         "isNavigational": false,
                         "language": "en",
                         "name": "How old is Barack Obama - Answers",
                         "snippet": "The correct spelling Barack Obama. Barack "
                                    "Obama was born on August 4th, 1961. As of "
                                    "February 2015, Barack Obama is 53 years "
                                    "old.",
                         "url": "https://www.answers.com/Q/How_old_is_Barack_Obama"},
    ...
```

The `nboost` key signifies that the webpages were reranked. NBoost returns the `answer_text`, and the offsets from the QA Model in the `answer_start_pos` and `answer_stop_pos` keys.

CHAPTER 3

nboost command line

CHAPTER 4

nboost-index command line

CHAPTER 5

nboost package

5.1 Subpackages

5.1.1 nboost.indexers package

Submodules

nboost.indexers.base module

nboost.indexers.cli module

nboost.indexers.defaults module

Default nboost-indexer command line arguments

nboost.indexers.es module

nboost.indexers.test_es_indexer module

Module contents

5.1.2 nboost.plugins package

Subpackages

nboost.plugins.models package

Subpackages

nboost.plugins.models.qa package

Subpackages

nboost.plugins.models.qa.pt package

Submodules

nboost.plugins.models.qa.pt.distilbert module

Module contents

Submodules

nboost.plugins.models.qa.base module

Module contents

nboost.plugins.models.rerank package

Subpackages

nboost.plugins.models.rerank.onnx package

Submodules

nboost.plugins.models.rerank.onnx.bert module

Module contents

nboost.plugins.models.rerank.pt package

Submodules

nboost.plugins.models.rerank.pt.bert module

Module contents

nboost.plugins.models.rerank.tf package

Subpackages

nboost.plugins.models.rerank.tf.albert package

Submodules

Module contents

5.2 Submodules

5.2.1 nboost.cli module

5.2.2 nboost.compat module

class nboost.compat.**BackwardsCompatibility**
Bases: object

Augment global modules to be backwards compatible

set()

5.2.3 nboost.database module

5.2.4 nboost.defaults module

5.2.5 nboost.delegates module

5.2.6 nboost.exceptions module

NBoost base exceptions

exception nboost.exceptions.**FrontendRequest**
Bases: *nboost.exceptions.RequestException*

Client sent frontend request

exception nboost.exceptions.**InvalidChoices**
Bases: *nboost.exceptions.ResponseException*

The length of choices, choice ids, and choice values must be the same

exception nboost.exceptions.**MissingQuery**
Bases: *nboost.exceptions.RequestException*

Could not parse query in request

exception nboost.exceptions.**RequestException**
Bases: Exception

Exception when receiving client request

exception nboost.exceptions.**ResponseException**
Bases: Exception

Upstream response contains error message

exception nboost.exceptions.**StatusRequest**
Bases: *nboost.exceptions.RequestException*

Client sent status request

exception nboost.exceptions.**UnknownRequest**
Bases: *nboost.exceptions.RequestException*

Unrecognized url path in request

exception nboost.exceptions.**UpstreamConnectionError**
Bases: Exception

Raised when the upstream host refuses connection

exception nboost.exceptions.**UpstreamServerError**
Bases: *nboost.exceptions.ResponseException*

Raised when the upstream server sends an error status code.

5.2.7 nboost.helpers module

5.2.8 nboost.logger module

Logger for NBoost classes

class nboost.logger.**ColoredFormatter**(*fmt=None*, *datefmt=None*, *style='%'*)
Bases: logging.Formatter

Format log levels with color

Initialize the formatter with specified format strings.

Initialize the formatter either with the specified format string, or a default as described above. Allow for specialized date formatting with the optional datefmt argument. If datefmt is omitted, you get an ISO8601-like (or RFC 3339-like) format.

Use a style parameter of '%', '{' or '$' to specify that you want to use one of %-formatting, str.format() ({}) formatting or string.Template formatting in your format string.

Changed in version 3.2: Added the style parameter.

MAPPING = {'CRITICAL': {'color': 'grey', 'on_color': 'on_blue'}, 'DEBUG': {'color':

PREFIX = '\x1b['

SUFFIX = '\x1b[0m'

format(*record*)
Add log ansi colors

class nboost.logger.**NTLogger**(*context*, *verbose*)
Bases: object

Windows support for logger

format_msg(*string_format*)
Format incoming logging messages with a given format

nboost.logger.**set_logger**(*context*, *verbose=False*)
Return colored logger with specified context name and debug=verbose

5.2.9 nboost.maps module

5.2.10 nboost.proxy module

5.2.11 nboost.translators module

5.3 Module contents

General nboost package parameters

CHAPTER 6

Benchmarking NBoost

NBoost comes with out-of-the-box functionality to benchmark the speed and accuracy of NBoost. If you tell NBoost which documents should be returned by a query, it will calculate how much better (or worse) the model is at ranking search results than the search api.

This is made possible by sending the correct document ids with either the `?nboost=` query parameter (comma delimited) or `"nboost"` key in the query json body (as an array).

6.1 Index the background corpus

For demonstration, we will be indexing the MS MARCO dataset. Download the necessary collection and queries tsv files from this link and extract them.

Use the `nboost-index` utility to send the data to your Elasticsearch:

```
tar -xvzf collectionandqueries.tar.gz
nboost-index --file collection.tsv --index_name ms_marco --host <Elasticsearch host>
```

The corpus is composed of 8.8 million paragraphs, so go grab a coffee while it finishes indexing…

6.2 Start the Proxy

Start the NBoost proxy using one of the main methods:

1. `nboost --uhost <Elasticsearch host> --uport 9200`
2. `docker run koursaros/nboost:latest-tf --uhost <Elasticsearch host> --uport 9200`
3. `helm install --name nboost --set uhost=<Elasticsearch host> --set uport=9200 nboost/nboost`

6.3 Benchmark on the test set

```
import requests, csv
from collections import defaultdict

with open('qrels.dev.small.tsv') as file:
    qid_map = defaultdict(list)
    for qid, _, cid, _ in csv.reader(file, delimiter='\t'):
        qid_map[qid].append(cid)

with open('queries.dev.small.tsv') as file:
    for qid, query in csv.reader(file, delimiter='\t'):
        cids = qid_map[qid]
        print(requests.post(
            url='http://localhost:8000/ms_marco/_search',
            json={
                'nboost': {
                    'rerank_cids': cids,
                }
            },
            params={
              'q': query,
            }
        ).json())
```

Now check out the frontend at localhost:8000/nboost!

You should find the model latency and calculated MRR for Elasticsearch vs NBoost. Here's our output:

MRR is short for mean reciprocal rank. Even though this model was finetuned on a different dataset, it was generalizable enough to increase Elasticsearch search relevancy by **70%** (0.29 / 0.17)!

CHAPTER 7

Data Structures

There are three main data structures in NBoost, and they are all dictionaries: the request, response, and config.

7.1 Request

```
{
  "method": "(str) the http method",
  "version": "(str) the http version",
  "headers": "(dict) the dictionary of http headers",
  "url":  {
    "scheme": "(str) URL scheme specifier",
    "netloc": "(str) network location part",
    "path": "(str) hierarchical path",
    "params": "(str) parameters for last path element",
    "query": "(dict) query parameters",
    "fragment": "(str) fragment identifier"
  },
  "body": {
    "nboost": "(dict) Configures NBoost at runtime. The same as the config.",
    "...other keys": "(dict) the rest of the body to send to the upstream server (e.g.
↪ Elasticsearch)"
  }
}
```

The url is in the form of `scheme://netloc/path;parameters?query#fragment`, but is decomposed into a dictionary, as noted above.

7.2 Response

```
{
  "status": "(int) the status code of the response message",
  "headers": "(dict) the dictionary of http headers",
  "body": {
    "nboost": "(dict) NBoost specific metadata, plugin data, etc..."
  }
}
```

7.3 Config

```
{
  "delim": "(str) the deliminator to concatenate multiple queries into a single query
↪",
  "query_path": "(str) the jsonpath in the request to find the query",
  "topk_path": "(str) the jsonpath to find the number of requested results",
  "choices_path":  "(str) the jsonpath to find the array of choices to reorder",
  "cvalues_path": "(str) the jsonpath to find the string values of the choices",
  "cids_path": "(str) the jsonpath to find the ids of the choices (for benchmarking)",
  "search_path": "(str) the route to capture for reranking search results",
  "default_topk": "(int) the default number of results to return if the topk_path is␣
↪not found"
}
```

CHAPTER 8

Contributing to NBoost

First of all, thanks for being willing to contribute to NBoost, we love to learn best practice from the community.

8.1 Making Commits

Contributions are greatly appreciated! You can make corrections or updates and commit them to NBoost. Here are the steps:

1. Create a new branch, say `fix-nboost-typo-1`
2. Fix/improve the codebase
3. Commit the changes. Note the **commit message must follow the naming style**, say `Fix/Model-bert: improve the readability and move sections`
4. Make a pull request. Note the **pull request must follow the naming style**. It can simply be one of your commit messages, just copy paste it, e.g. `Fix/Model-bert: improve the readability and move sections`
5. Submit your pull request and wait for all checks passed (usually 10 minutes)
   - Coding style
   - Commit and PR styles check
   - All unit tests
6. Request reviews from one of the developers from our core team.
7. Merge!

8.2 Table of Content

8.3 Commit Message Naming

To help everyone with understanding the commit history of NBoost, we employ `commitlint` in the CI pipeline to enforce the commit styles. Specifically, our convention is:

```
Type/Scope: subject
```

where `type` is one of the following:

- build
- ci
- chore
- docs
- feat
- fix
- perf
- refactor
- revert
- style
- test

`scope` is optional, represents the module your commit working on.

`subject` explains the commit.

As an example, a commit that implements a new encoder should be phrased as:

```
Fix/Model-bert: improve the readability and move sections
```

8.4 Merging Process

A pull request has to meet the following conditions to be merged into master:

- Coding style check (PEP8, via Codacy)
- Commit style check (in CI pipeline via Drone.io)
- Unit tests (via Drone.io)
- Review and approval from a Koursaros team member.

CHAPTER 9

Release 0.2.2

Jan 22nd, 2020

9.1 Hooks

- [f8974b4] fix benchmarking functionality (*colethienes*) 52

9.2 Models

- [244e547] add filter_results arg compatibility to rank fn (*TeoZosa*) 8

CHAPTER 10

Release 0.2.1

Jan 17th, 2020

10.1 Indexers

- [4dcc035] ability to index multiple cols (*colethienes*) 128

CHAPTER 11

Release 0.2.0

Jan 12th, 2020

11.1 Models

- [3641fa7] add filtering methods (*colethienes*) 20
- [6b83a08] qa model benchmarking (*colethienes*) 853
- [afb1dd1] Feature / Run QA model on GPU (*Jack Pertschuk*) 4

11.2 Hooks

- [edcc676] setting topn/topk (*colethienes*) 7
- [6cbc5cf] fix request hook url (*colethienes*) 19

11.3 Proxy

- [a479650] add debugging mode and session hooks (*colethienes*) 289
- [8fd9148] refactor to session format (*colethienes*) 698
- [6b83a08] qa model benchmarking (*colethienes*) 853

11.4 Session

- [3093247] add session and refactor defaults (*colethienes*) 562

11.5 Helpers

- [75c0352] patch jsonpath setter (*colethienes*) 22

CHAPTER 12

Release 0.1.1

Dec 18th, 2019

12.1 Proxy

- [c54b0aa] assume choices are list (*colethienes*) 22

12.2 Models

- [498b96b] Fix / char offsets for QA model (*Jack Pertschuk*) 16

CHAPTER 13

Release 0.1.0

Dec 18th, 2019

13.1 Proxy

- [7b7feda] relative response choices jsonpath (*colethienes*) 29

13.2 Models

- [03745d1] Add / Feature filter results based on classification model (*Jack Pertschuk*) 37

CHAPTER 14

Release 0.0.9

Dec 16th, 2019

14.1 Models

- [03745d1] Add / Feature filter results based on classification model (*Jack Pertschuk*) 37

CHAPTER 15

Release 0.0.8

Dec 15th, 2019

15.1 Cli

- [05aaec7] configurable jsonpaths (*colethienes*) 20
- [7d0fafe] tinybert default (*colethienes*) 2
- [bdd1365] add qa model (*colethienes*) 61

15.2 Proxy

- [d095b27] model directory instantiation (*colethienes*) 358
- [af9f3a8] refactor routing and protocols (*colethienes*) 369
- [1bf21e4] refactor messaging to dict type (*colethienes*) 2322

15.3 Models

- [3e8482e] Add / TinyBERT Model Code (*Jack Pertschuk*) 5
- [18d62ec] Add / QA Models Code (*Jack Pertschuk*) 82
- [8444a83] Fix/ Clean up transformers (*Jack Pertschuk*) 73
- [7a142ec] Fix/ Update transformers to latest version, remove online training functionality (*Jack Pertschuk*) 55

CHAPTER 16

Release 0.0.7

Dec 3rd, 2019

16.1 Proxy

- [7a120ce] record search boost (*colethienes*) 10
- [14446e6] fix connection dynamic (*colethienes*) 8
- [1052328] add frontend (*colethienes*) 61

CHAPTER 17

Release 0.0.6

Dec 2nd, 2019

17.1 Model

- [ce69a5b] support choices (*colethienes*) 11

17.2 Indexer

- [1e3ff4f] update benchmarking methods (*colethienes*) 56
- [7dcbc79] set default port (*colethienes*) 2
- [2dc832e] refactor indexing capabilities (*colethienes*) 7566

17.3 Proxy

- [0d493c4] add native Search boost benchmarking (*colethienes*) 121

17.4 Resources

- [5542075] add nboost/resources (*colethienes*) 6967

17.5 Helm

- [93f0774] initial helm support (*colethienes*) 13

CHAPTER 18

Release 0.0.5

Nov 30th, 2019

18.1 Benchmark

- [6e739ea] parser construct (*colethienes*) 12

18.2 Cli

- [7e03d4b] use version file (*colethienes*) 21

CHAPTER 19

Release 0.0.4

Nov 25th, 2019

19.1 Proxy

- [23063ed] remove unnecessary stats (*colethienes*) 2

19.2 Base

- [8719745] refactor to codex style (*colethienes*) 2166
- [55cac19] refactor raw http (*colethienes*) 221

19.3 Model

- [a543f26] update/add Biobert Model (*Jack Pertschuk*) 4

CHAPTER 20

Release 0.0.3

Nov 25th, 2019

20.1 Base

- [913da69] update flowchart and overview (*colethienes*) 4

20.2 Benchmark

- [d28b020] update/benchmark allow multiple shards (*Jack Pertschuk*) 10
- [eca7dc4] update/benchmark check collection length (*Jack Pertschuk*) 6

20.3 Tutorial

- [e16b8b6] add entrypoints (*colethienes*) 16

20.4 Cli

- [a27131c] change entrypoint (*colethienes*) 2

CHAPTER 21

Release 0.0.2

Nov 23rd, 2019

21.1 Protocol

- [f5ee34b] ascii encoding (*colethienes*) 18

21.2 Tutorial

- [6354cf4] proxy setup fixes (*colethienes*) 4
- [32dc7cf] add tutorial fixes (*colethienes*) 17154

21.3 Model

- [683d0f4] tf logging (*colethienes*) 2
- [77e16f1] update/ added verbose mode (*Jack Pertschuk*) 5

21.4 Base

- [05c7f70] custom model dir (*Jack Pertschuk*) 7

CHAPTER 22

Release 0.0.1

Nov 23rd, 2019

22.1 Base

- [0eafd65] v0.0.1! (*colethienes*) 2

CHAPTER 23

Indices and tables

- genindex
- modindex
- search

Python Module Index

n

Index