
build unknown

nlg-eval

Evaluation code for various unsupervised automated metrics for NLG (Natural Language Generation). It takes as input a hypothesis file, and one or more references files and outputs values of metrics. Rows across these files should correspond to the same example.

Metrics

- BLEU
- METEOR
- ROUGE
- CIDEr
- SPICE
- SkipThought cosine similarity
- Embedding Average cosine similarity
- Vector Extrema cosine similarity
- Greedy Matching score

Setup

Install Java 1.8.0 (or higher).

Install the Python dependencies, run:

```
1 pip install git+https://github.com/Maluuba/nlg-eval.git@master
```

If you are using macOS High Sierra or higher, then run this to allow multithreading:

```
1 export OBJC_DISABLE_INITIALIZE_FORK_SAFETY=YES
```

Simple setup (download required data (e.g. models, embeddings) and external code files), run:

```
1 nlg-eval --setup
```

If you're setting this up from the source code or you're on Windows and not using a Bash terminal, then you might get errors about `nlg-eval` not being found. You will need to find the `nlg-eval` script. See [here](#) for details.

Custom Setup

```
1 # If you don't like the default path (~/.cache/nlgeval) for the
   # downloaded data,
2 # then specify a path where you want the files to be downloaded.
3 # The value for the data path is stored in ~/.config/nlgeval/rc.json
   # and can be overwritten by
4 # setting the NLGEVAL_DATA environment variable.
5 nlgeval --setup ${data_path}
```

Validate the Setup (Optional)

(These examples were made with Git Bash on Windows)

All of the data files should have been downloaded, you should see sizes like:

```
1 $ ls -l ~/.cache/nlgeval/
2 total 6003048
3 -rw-r--r-- 1 ... 289340074 Sep 12 2018 bi_skip.npz
4 -rw-r--r-- 1 ... 689 Sep 12 2018 bi_skip.npz.pkl
5 -rw-r--r-- 1 ... 2342138474 Sep 12 2018 btable.npy
6 -rw-r--r-- 1 ... 7996547 Sep 12 2018 dictionary.txt
7 -rw-r--r-- 1 ... 21494787 Jan 22 2019 glove.6B.300d.model.bin
8 -rw-r--r-- 1 ... 480000128 Jan 22 2019 glove.6B.300d.model.bin.
   vectors.npy
9 -rw-r--r-- 1 ... 663989216 Sep 12 2018 uni_skip.npz
10 -rw-r--r-- 1 ... 693 Sep 12 2018 uni_skip.npz.pkl
11 -rw-r--r-- 1 ... 2342138474 Sep 12 2018 utable.npy
```

You can also verify some checksums:

```
1 $ cd ~/.cache/nlgeval/
2 $ md5sum *
3 9a15429d694a0e035f9ee1efcb1406f3 *bi_skip.npz
4 c9b86840e1dedb05837735d8bf94cee2 *bi_skip.npz.pkl
5 022b5b15f53a84c785e3153a2c383df6 *btable.npy
6 26d8a3e6458500013723b380a4b4b55e *dictionary.txt
7 f561ab0b379e23cbf827a054f0e7c28e *glove.6B.300d.model.bin
8 be5553e91156471fe35a46f7dcd4c44e *glove.6B.300d.model.bin.vectors.npy
9 8eb7c6948001740c3111d71a2fa446c1 *uni_skip.npz
10 e1a0ead377877ff3ea5388bb11cfe8d7 *uni_skip.npz.pkl
11 5871cc62fc01b79788c79c219b175617 *utable.npy
12 $ sha256sum *
13 8ab7965d2db5d146a907956d103badfa723b57e0acffb75e10198ba9f124edb0 *
   bi_skip.npz
14 d7e81430fcdcbcb60b36b92b3f879200919c75d3015505ee76ae3b206634a0eb6 *
   bi_skip.npz.pkl
15 4a4ed9d7560bb87f91f241739a8f80d8f2ba787a871da96e1119e913ccd61c53 *
```

```
16 4dc5622978a30cddea8c975c871ea8b6382423efb107d27248ed7b6cfa490c7c *  
   dictionary.txt  
17 10c731626e1874effc4b1a08d156482aa602f7f2ca971ae2a2f2cd5d70998397 *glove  
   .6B.300d.model.bin  
18 20dfb1f44719e2d934bf5e5d39a6ffb4f248bae2a00a0d59f953ab7d0a39c879 *glove  
   .6B.300d.model.bin.vectors.npy  
19 7f40ff16ff5c54ce9b02bd1a3eb24db3e6adaf7712a7a714f160af3a158899c8 *  
   uni_skip.npz  
20 d58740d46cba28417cbc026af577f530c603d81ac9de43ffd098f207c7dc4411 *  
   uni_skip.npz.pkl  
21 790951d4b08e843e3bca0563570f4134ffd17b6bd4ab8d237d2e5ae15e4febb3 *  
   utable.npy
```

If you're ensure that the setup was successful, you can run the tests:

```
1 pip install pytest  
2 pytest
```

It might take a few minutes and you might see warnings but they should pass.

Usage

Once setup has completed, the metrics can be evaluated with a Python API or in the command line.

Examples of the Python API can be found in `test_nlgeval.py`.

Standalone

```
1 nlg-eval --hypothesis=examples/hyp.txt --references=examples/ref1.txt  
   --references=examples/ref2.txt
```

where each line in the hypothesis file is a generated sentence and the corresponding lines across the reference files are ground truth reference sentences for the corresponding hypothesis.

functional API: for the entire corpus

```
1 from nlgeval import compute_metrics  
2 metrics_dict = compute_metrics(hypothesis='examples/hyp.txt',  
3                               references=['examples/ref1.txt', '  
                                           examples/ref2.txt'])
```

functional API: for only one sentence

```
1 from nlgeval import compute_individual_metrics
2 metrics_dict = compute_individual_metrics(references, hypothesis)
```

where `references` is a list of ground truth reference text strings and `hypothesis` is the hypothesis text string.

object oriented API for repeated calls in a script - single example

```
1 from nlgeval import NLGEval
2 nlgeval = NLGEval() # loads the models
3 metrics_dict = nlgeval.compute_individual_metrics(references,
    hypothesis)
```

where `references` is a list of ground truth reference text strings and `hypothesis` is the hypothesis text string.

object oriented API for repeated calls in a script - multiple examples

```
1 from nlgeval import NLGEval
2 nlgeval = NLGEval() # loads the models
3 metrics_dict = nlgeval.compute_metrics(references, hypothesis)
```

where `references` is a list of lists of ground truth reference text strings and `hypothesis` is a list of hypothesis text strings. Each inner list in `references` is one set of references for the hypothesis (a list of single reference strings for each sentence in `hypothesis` in the same order).

Reference

If you use this code as part of any published research, please cite the following paper:

Shikhar Sharma, Layla El Asri, Hannes Schulz, and Jeremie Zumer. **“Relevance of Unsupervised Metrics in Task-Oriented Dialogue for Evaluating Natural Language Generation”** *arXiv preprint arXiv:1706.09799* (2017)

```
1 @article{sharma2017nlgeval,
2   author = {Sharma, Shikhar and El Asri, Layla and Schulz, Hannes
3     and Zumer, Jeremie},
4   title = {Relevance of Unsupervised Metrics in Task-Oriented
5     Dialogue for Evaluating Natural Language Generation},
6   journal = {CoRR},
7   volume = {abs/1706.09799},
8   year = {2017},
```

```
7 url = {http://arxiv.org/abs/1706.09799}
8 }
```

Example

Running

```
1 nlgeval --hypothesis=examples/hyp.txt --references=examples/ref1.txt
  --references=examples/ref2.txt
```

gives

```
1 Bleu_1: 0.550000
2 Bleu_2: 0.428174
3 Bleu_3: 0.284043
4 Bleu_4: 0.201143
5 METEOR: 0.295797
6 ROUGE_L: 0.522104
7 CIDEr: 1.242192
8 SPICE: 0.312331
9 SkipThoughtsCosineSimilarity: 0.626149
10 EmbeddingAverageCosineSimilarity: 0.884690
11 VectorExtremaCosineSimilarity: 0.568696
12 GreedyMatchingScore: 0.784205
```

Troubleshooting

If you have issues with Meteor then you can try lowering the `mem` variable in `meteor.py`

Important Note

CIDEr by default (with `idf` parameter set to “corpus” mode) computes IDF values using the reference sentences provided. Thus, CIDEr score for a reference dataset with only 1 image (or example for NLG) will be zero. When evaluating using one (or few) images, set `idf` to “coco-val-df” instead, which uses IDF from the MSCOCO Validation Dataset for reliable results. This has not been adapted in this code. For this use-case, apply patches from `vrama91/coco-caption`.

External data directory

To mount an already prepared data directory to a Docker container or share it between users, you can set the `NLGEVAL_DATA` environment variable to let `nlgeval` know where to find its models and data. E.g.

```
1 NLGEVAL_DATA=~/.workspace/nlg-eval/nlgeval/data
```

This variable overrides the value provided during setup (stored in `~/.config/nlgeval/rc.json`)

Microsoft Open Source Code of Conduct

This project has adopted the Microsoft Open Source Code of Conduct. For more information see the Code of Conduct FAQ or contact opencode@microsoft.com with any additional questions or comments.

License

See LICENSE.md.