neleval Documentation

Release 3.1.1

neleval contributors

Feb 28, 2020

Contents

Python command-line evaluation scripts for TAC entity linking and related wikification, named entity disambiguation, and within- and cross-document coreference tasks.

It aims for **fast** and **flexible** coreference resolution and **sophisticated** named entity recognition evaluation, such as partial scores for partial overlap between gold and system mentions. CEAF, in particular, is much faster to calculate here than in the CoNLL-11/12 scorer. It boasts features such as *configurable metrics*; accounting for or ignoring cross-document coreference (see the *evaluate –by-doc* flag); plotting to compare evaluation by system, measure and corpus subset; and bootstrap-based confidence interval calculation for document-wise evaluation metrics.

CHAPTER 1

Documentation

1.1 Installation

Requirements:

- python 2.7 or >= 3.4
- numpy
- joblib
- scipy for fast CEAF calculation
- matplotlib for some commands

To install the latest release, use:

```
$ pip install neleval
```

To install the current development version, use:

```
$ pip install https://github.com/wikilinks/neleval/archive/master.zip
```

Running `neleval` on the shell should confirm success:

```
$ neleval
```

1.2 Basic Usage

The NEL evaluation tools are invoked using `neleval`, or `./nel` inside the repository. Usage:

```
neleval <command> [<args>]
```

To list available commands:

```
neleval
```

To get help for a specific command:

```
neleval <command> -h
```

See *Command-line reference*.

The commands that are relevant to TAC KBP entity linking evaluation and analysis are described below.

1.2.1 Basic usage

The following describes a typical workflow. See also *Convenience scripts for TAC KBP evaluation*.

Convert gold standard to evaluation format

For data in TAC14 format:

```
neleval prepare-tac \
    -q /path/to/gold.xml \     # gold queries/mentions file
    /path/to/gold.tab \        # gold KB/NIL annotations file
    > gold.combined.tsv
```

For data in TAC12 and TAC13 format, remove extra columns first, e.g.:

```
cat /path/to/gold.tab \
    | cut -f1,2,3 \
    > gold.tab
neleval prepare-tac \
    -q /path/to/gold.xml \
    gold.tab \
    > gold.combined.tsv
```

Convert system output to evaluation format

For data in TAC14 format:

```
neleval prepare-tac \
    -q /path/to/system.xml \   # system mentions file
    /path/to/system.tab \      # system KB/NIL annotations
    > system.combined.tsv
```

For data in TAC12 and TAC13 format, add dummy NE type column first, e.g.:

```
cat /path/to/system.tab \
    | awk 'BEGIN{OFS="\t"} {print $1,$2,"NA",$3}' \
    > system.tab
neleval prepare-tac \
    -q /path/to/gold.xml \     # gold queries/mentions file
    system.tab \               # system KB/NIL annotations
    > system.combined.tsv
```

Evaluate system output

To calculate micro-averaged scores for all evaluation measures:

```
neleval evaluate \
    -m all \                    # report all evaluation measures
    -f tab \                    # print results in tab-separated format
    -g gold.combined.tsv \      # prepared gold standard annotation
    system.combined.tsv \       # prepared system output
    > system.evaluation
```

To list available evaluation measures:

```
neleval list-measures
```

1.2.2 Advanced usage

The following describes additional commands for analysis. See also run_tac14_all.sh (TODO) and run_tac13_all.sh.

Calculate confidence intervals

To calculate confidence intervals using bootstrap resampling:

```
neleval confidence \
    -m strong_typed_link_match \ # report CI for TAC14 wikification measure
    -f tab \                    # print results in tab-separated format
    -g gold.combined.tsv \      # prepared gold standard annotation
    system.combined.tsv \       # prepared system output
    > system.confidence
```

We recommend that you `pip install joblib` and use `-j NUM_JOBS` to run this in parallel. This is also faster if an individual evaluation measure is specified (e.g., strong_typed_link_match) rather than groups of measures (e.g., tac).

The run_report_confidence.sh script is available to create reports comparing multiple systems.

Note that bootstrap resampling is not appropriate for nil clustering measures. For more detail, see the Significance wiki page.

Calculate significant differences

It is also possible to calculate pairwise differences:

```
neleval significance \
    --permute \                 # use permutation method
    -f tab \                    # print results in tab-separated format
    -g gold.combined.tsv \      # prepared gold standard annotation
    system1.combined.tsv \      # prepared system1 output
    system2.combined.tsv \      # prepared system2 output
    > system1-system2.significance
```

We recommend calculating significance for selected system pairs as it can take a while over all N choose 2 combinations of systems. You can also use `-j NUM_JOBS` to run this in parallel.

Note that bootstrap resampling is not appropriate for nil clustering measures. For more detail, see the Significance wiki page.

Analyze error types

To create a table of classification errors:

```
neleval analyze \
    -s \                        # print summary table
    -g gold.combined.tsv \      # prepared gold standard annnotation
    system.combined.tsv \       # prepared system output
    > system.analysis
```

Without the `-s` flag, the `analyze` command will list and categorize differences between the gold standard and system output.

1.2.3 Filter data for evaluation on subsets

The following describes a workflow for evaluation over subsets of mentions. See also run_tac14_filtered.sh (TODO) and run_tac13_filtered.sh.

Filter prepared data

Prepared data is in a simple tab-separated format with one mention per line and six columns: `document_id`, `start_offset`, `end_offset`, `kb_or_nil_id`, `score`, `entity_type`. It is possible to use command line tools (e.g., `grep`, `awk`) to select mentions for evaluation, e.g.:

```
cat gold.combined.tsv \         # prepared gold standard annotation
    | egrep "^eng-(NG|WL)-" \   # select newsgroup and blog (WB) mentions
    > gold.WB.tsv               # filtered gold standard annotation
cat system.combined.tsv \       # prepared system output
    | egrep "^eng-(NG|WL)-" \   # select newsgroup and blog (WB) mentions
    > system.WB.tsv             # filtered system output
```

Evaluate on filtered data

After filtering, evaluation is run as before:

```
neleval evaluate \
    -m all \                    # report all evaluation measures
    -f tab \                    # print results in tab-separated format
    -g gold.WB.tsv \            # filtered gold standard annotation
    system.WB.tsv \             # filtered system output
    > system.WB.evaluation
```

Evaluate each document or entity type

To get a score for each document, or each entity type, as well as the macro-averaged score across documents, use `--group-by` in *neleval evaluate*. See *Grouped measures*.

1.3 Measures

neleval reports precision, recall and F1 for numerous set-wise and coreference measures.

1.3.1 Basic measures

The evaluation tool provides a range of linking and clustering evaluation measures. These are described briefly below and listed by the `nel list-measures` command. For more details of correspondences between linking measures here and in the literature, see Hachey et al. (2014). For clustering, see Pradhan et al. (2014). For a quick reference, see our cheatsheet. (As described there, evaluation can be performed across the whole corpus, or with separate scores for each document/type as well as micro- and macro-averages across all types/docs.)

Official TAC 2014 measures

TAC 2014 reports two official measures, one for linking/wikification and one for nil clustering. For more detail, see the TAC 2014 scoring page.

Linking evaluation

`strong_typed_all_match` is a micro-averaged evaluation of all mentions. A mention is counted as correct if it is a correct link or a correct nil. A correct link must have the same span, entity type, and KB identifier as a gold link. A correct nil must have the same span as a gold nil. This is the official linking evaluation measure for TAC 2014.

Clustering evaluation

`mention_ceaf` is based on a one-to-one alignment between system and gold clusters — both KB and nil. It computes an optimal mapping based on overlap between system-gold cluster pairs. System and gold mentions must have the same span to affect the alignment. Unmatched mentions also affect precision and recall.

Additional diagnostic measures

The evaluation tool also provides a number of diagnostic measures available to isolate performance of system components and compare to numbers reported elsewhere in the literature.

Mention detection evaluation

`strong_mention_match` is a micro-averaged evaluation of entity mentions. A system span must match a gold span exactly to be counted as correct.

`strong_typed_mention_match` additionally requires the correct entity type. This is equivalent to the CoNLL NER evaluation (Tjong Kim Sang & De Meulder, 2003).

`strong_linked_mention_match` is the same as `strong_mention_match` but only considers non-nil mentions that are linked to KB identifier.

Measures sensitive to partial overlap between the system and gold mentions, using the LoReHLT metric can be constructed with aggregates such as `overlap-sumsum`. See the *Measures in detail*.

Linking evaluation

`strong_link_match` is a micro-averaged evaluation of links. A system link must have the same span and KB identifier as a gold link to be counted as correct. This is equivalent to Cornolti et al.'s (2013) strong annotation match. Recall here is equivalent to KB accuracy from TAC tasks before 2014.

`strong_nil_match` is a micro-averaged evaluation of nil mentions. A system nil must have the same span as a gold nil to be counted as correct. Recall here is equivalent to nil accuracy from TAC tasks before 2014.

`strong_all_match` is a micro-averaged link evaluation of all mentions. A mention is counted as correct if is either a link match or a nil match as defined above. This is equivalent to overall accuracy from TAC tasks before 2014.

Document-level tagging evaluation

`entity_match` is a micro-averaged document-level set-of-titles measure. It is the same as entity match reported by Cornolti et al. (2013).

Clustering evaluation

`entity_ceaf` — like `mention_ceaf` — is based on a one-to-one alignment between system and gold entity clusters. Here system-gold cluster pairs are scored by their Dice coefficient.

`b_cubed` assesses the proportion of each mention's cluster that is shared between gold and predicted clusterings.

`b_cubed_plus` is identical to `b_cubed`, but additionally requires a correct KB identifier for non-nil mentions.

`muc` counts the number of edits required to translate the gold clustering into the prediction.

`pairwise` measures the proportion of mention pairs occurring in the same cluster in both gold and predicted clusterings. It is similar to the Rand Index.

For more detail, see Pradhan et al.'s (2014) excellent overview of clustering measures for coreference evaluation, and our Coreference_Evaluation.

Custom measures

Our scorer supports specification of some custom evaluation measures. See *neleval list-measures*.

References

Cornolti et al. (2013). A framework for benchmarking entity-annotation systems. In WWW.

Hachey et al. (2014). Cheap and easy entity evaluation. In ACL.

Ji & Grishman (2011). Knowledge base population: successful approaches and challenges. In ACL.

Pradhan et al. (2014). Scoring Coreference Partitions of Predicted Mentions: A Reference Implementation. In ACL.

Tjong Kim Sang & De Meulder (2003). Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In CoNLL.

1.3.2 Coreference evaluation

Pradhan et al. have published "Scoring Coreference Partitions of Predicted Mentions: A Reference Implementation" (ACL 2014) describing their Perl-based scoring tool AKA `scorer.pl`. The neleval package reimplements these measures (MUC, B-cubed, Entity CEAF, Mention CEAF, and the pairwise coreference and non-coreference measures that constitute BLANC) with a number of efficiency improvements, particularly to CEAF, and especially valuable in the cross-document coreference evaluation setting.

CEAF calculation efficiency

The slow part of calculating CEAF is identifying the maximal linear-sum assignment between key and response entities, using the Hungarian Algorithm or a variant thereof. Our implementation is much faster because: * scorer.pl manipulates Perl arrays and may be O(n^4), though I haven't checked, where *n* is the number of key and response entities; we use an O(n^3) implementation with vectorised NumPy operations in a very efficient implementation that was recently adopted into scipy. Even before further optimisations, this resulted in an order of magnitude or more runtime improvement over . * Our *n* is much smaller in practice. We only perform the Hungarian Algorithm on each strongly connected component of the assignment graph, and explicitly eliminate trivial portions of the assignment problem (where there is no confusion with other entities). So our time complexity is O(n^3) where *n* is the number of entities in the largest component, rather than the total number of entities in the evaluation. These optimisations are particularly valuable in cross-document coref evaluation because the number of entities is large relative to the number of confusions. * We have also made some efficient choices elsewhere in processing, such as determining entity overlaps using `scipy.sparse` matrix multiplication.

Both our implementation and `scorer.pl` support $\phi 3$ and $\phi 4$ of Luo's 2005 paper introducing CEAF. Our mention_ceaf = ceafm = $\phi 3$. Our entity_ceaf = ceafe = $\phi 4$.

Note on BLANC

Note that we do not directly report BLANC, although we facilitate calculation of both its components, using `pairwise` and `pairwise_negative` aggregates (see our *neleval list-measures* command), according to Luo et al. 2015's extension of the metric to system mentions.

Validation of equivalence to reference implementation

We have empirically verified the equivalence of metric implementation between our system and `scorer.pl`. By pointing the `COREFSCORER` environment variable to a local copy of `scorer.pl`, our system will cross-check the results automatically. (This will, however, be extremely slow for large CEAF calculations.)

Importing CoNLL 2011-2012 shared task formatted data

We provide the *neleval prepare-conll-coref* command to import CoNLL shared task-formatted annotations. We have validated that our metrics match those produced by Pradhan et al.'s reference implementation for the CoNLL 2011 runs.

1.3.3 Measures in detail

This describes measures as listed by *neleval list-measures*.

Measure	Key	Filter	Aggregator
Mention evaluation measures			
strong_mention_match	span	NA	sets
strong_typed_mention_match	span,type	NA	sets
strong_linked_mention_match	span	is_linked	sets
Linking evaluation measures			
strong_link_match	span,kbid	is_linked	sets
strong_nil_match	span	is_nil	sets
strong_all_match	span,kbid	NA	sets
strong_typed_link_match	span,type,kbid	is_linked	sets
strong_typed_nil_match	span,type	is_nil	sets
strong_typed_all_match	span,type,kbid	NA	sets
Document-level tagging evaluation			
entity_match	docid,kbid	is_linked	sets
Clustering evaluation measures			
muc	span	NA	muc
b_cubed	span	NA	b_cubed
b_cubed_plus	span,kbid	NA	b_cubed
entity_ceaf	span	NA	entity_ceaf
mention_ceaf	span	NA	mention_ceaf
pairwise	span	NA	pairwise

Custom measures

A custom measure can be specified on the command-line as:

`<aggregator>:<filter>:<key>`

such as

`sets:None:span+kbid` for *strong_all_match*

Grouped measures

By default measures are aggregated over the corpus as a whole. Using the `--by-doc` and/or `--by-type` flags to *neleval evaluate* will instead aggregate measures per document or entity type, and then report per-doc/type and overall (micro- and macro-averaged) performance. *Note that micro-average does not equate to whole-corpus aggregation for coreference aggregates, but represents clustering performance disregarding cross-document coreference.*

Key

The key defines how system output is matched against the gold standard.

Key	Description
docid	Document identifier must be the same
start	Start offset must be the same
end	End offset must be the same
span	Shorthand for (docid, start, end)
type	Entity type must be the same
kbid	KB identifier must be the same, or must both be NIL

Filter

The filter defines what mentions are removed before precision, recall and f-score calculations.

Filter	Description
is_linked	Only keep mentions that are resolved to known KB identifiers
is_nil	Only keep mentions that are not resolved to known KB identifiers
is_first	Only keep the first mention in a document of a given KB/NIL identifier

Note that the *is_first* filter is intended to provide clustering evaluation similar to the *entity_match* evaluation of linking performance.

Aggregator

The aggregator defines how corpus-level scores are computed from individual instances.

Aggregator	Description
Mention, linking, tagging evaluations	
sets	Take the unique set of tuples as defined by **key** across the gold and system data, then micro-average document-level tp, fp and fn counts.
overlap-{max,sum}{max,sum}	For tasks in which the gold and system must produce non-overlapping annotations, these scores account for partial overlap between gold and system mentions, as defined for the LoReHLT evaluation.
Clustering evaluation	
muc	Count the total number of edits required to translate from the gold to the system clustering
b_cubed	Assess the proportion of each mention's cluster that is shared between gold and system clusterings
entity_ceaf	Calculate optimal one-to-one alignment between system and gold clusters based on Dice coefficient, and get the total aligned score relative to aligning each cluster with itself
mention_ceaf	Calculate optimal one-to-one alignment between system and gold clusters based on number of overlapping mentions, and get the total aligned score relative to aligning each cluster with itself
pairwise	The proportion of true co-clustered mention pairs that are predicted, etc., as used in computing BLANC
pairwise_negative	The proportion of true *not* co-clustered mention pairs that are predicted, etc., as used in computing BLANC

1.3.4 Approximate matching

Measures ordinarily score 1 when gold and system annotations exist that have an exact match for all elements of the *key*.

For some kinds of measure it is possible to award partial matches for:

- mention pairs with overlapping, but not identical, spans
- mention pairs with related, but not identical, entity types
- mention pairs with related, but not identical, KB entries (disambiguands)

Overlapping spans

To give partial award to overlapping gold and system mentions, we use the scheme developed by Ryan Gabbard of BBN for LoReHLT:

> We award systems for partial matches according to the degree of character overlap between system and key names. The partial match scoring algorithm has two parameters: the recall overlap strategy and the precision overlap strategy.
>
> - The per-name recall score of a name in the answer key is the fraction of its characters which overlap with the system name set according to the recall overlap strategy parameter. For the "MAX" strategy, this will be the characters overlapping with the single system name with maximum overlap. For the "SUM" strategy, this will be the number of its characters which overlap with any system mention.
> - The recall score for a system is the mean of the per-name recall scores for all names in the answer key.
> - The per-name precision score of a name in the answer key is the fraction of its characters overlapped by the reference set, where "overlapping" is determined by the precision overlap strategy in the same manner as above for recall.
> - The precision score for a system is the mean of the per-name precision scores for all names in the answer key.

This applies to measures with *aggregator*:

- `overlap-maxmax` for recall and precision overlap strategies both MAX
- `overlap-maxsum` for recall overlap strategy MAX and precision overlap strategy SUM
- `overlap-summax` for recall overlap strategy SUM and precision overlap strategy MAx
- `overlap-sumsum` for recall and precision overlap strategies both SUM

In the following example, the gold standard includes a mention from character 1 to 10 and another from 12 to 12. The system includes a mention from 1 to 5 and another from 6 to 12.

```
$ bash -c "\
neleval evaluate \
-m overlap-maxmax::span \
-m overlap-maxsum::span \
-m overlap-summax::span \
-m overlap-sumsum::span \
-m sets::span \
-g <(echo -e 'd\t1\t10\nd\t12\t12') \
   <(echo -e 'd\t1\t5\nd\t6\t12')"
ptp        fp        rtp        fn        precis        recall        fscore        measure
1.714        0.286        1.500        0.500        0.857        0.750        0.
↪800        overlap-maxmax::span
1.857        0.143        1.500        0.500        0.929        0.750        0.
↪830        overlap-maxsum::span
1.714        0.286        2.000        0.000        0.857        1.000        0.
↪923        overlap-summax::span
1.857        0.143        2.000        0.000        0.929        1.000        0.
↪963        overlap-sumsum::span
0        2        0        2        0.000        0.000        0.000        sets::span
```

TODO: flesh out calculation

Caveats:

- All mentions within the gold annotation must be non-overlapping.

- All mentions within the system annotation must be non-overlapping.
- There is (currently) no equivalent implementation for clustering metrics.

Approximate type matching

Rather than exactly matching entity types, they can be matched using arbitrary weights. These can be specified to *neleval evaluate* with `--type-weights`. This option accepts a tab-delimited file with three columns:

- gold type
- system type
- weight

For types not in this weight file, exact matches between gold type and system type score 1, and otherwise score is 0. If multiple gold/system entries exist, the maximum weight is used.

The following example scores 0.123 where the gold type is `type1` and the system type is `type2`.

```
$ bash -c " \
neleval evaluate --by-doc \
-m strong_typed_mention_match \
--type-weights <(echo -e 'type1\ttype2\t0.123') \
--gold <( \
echo -e 'doc1\t10\t20\tkbid\t1.0\ttype1'; \
echo -e 'doc2\t10\t20\tkbid\t1.0\ttype1'; \
echo -e 'doc3\t10\t20\tkbid\t1.0\ttype2'; \
echo -e 'doc4\t10\t20\tkbid\t1.0\ttype1'; \
echo -e 'doc4\t30\t40\tkbid\t1.0\ttype1'; \
) <( \
echo -e 'doc1\t10\t20\tkbid\t1.0\ttype2'; \
echo -e 'doc2\t10\t20\tkbid\t1.0\ttype1'; \
echo -e 'doc3\t10\t20\tkbid\t1.0\ttype1'; \
echo -e 'doc4\t10\t20\tkbid\t1.0\ttype2'; \
echo -e 'doc4\t30\t40\tkbid\t1.0\ttype2'; \
) \
"
ptp        fp        rtp        fn        precis        recall        fscore        measure
0.123        0.877        0.123        0.877        0.123        0.123        0.
↪123        strong_typed_mention_match;docid="doc1"
1.000        0.000        1.000        0.000        1.000        1.000        1.
↪000        strong_typed_mention_match;docid="doc2"
0.000        1.000        0.000        1.000        0.000        0.000        0.
↪000        strong_typed_mention_match;docid="doc3"
0.246        1.754        0.246        1.754        0.123        0.123        0.
↪123        strong_typed_mention_match;docid="doc4"
0.342        0.908        0.342        0.908        0.311        0.311        0.
↪311        strong_typed_mention_match;docid=<macro>
1.369        3.631        1.369        3.631        0.274        0.274        0.
↪274        strong_typed_mention_match;docid=<micro>
```

This currently only applies to measures with the `sets` *aggregator*.

Type match weighting with a hierarchy

neleval weights-for-hierarchy converts a hierarchy of types into the above `--type-weights` format. It uses a scheme with a decay parameter $0 < d < 1$, such that a system mention is awarded:

- 0 if its type is not identical to or an ancestor of the gold type
- $d^{\text{depth(goldtype)}-\text{depth(systype)}}$ if its type is an ancestor of the gold type

Thus:

- d if its type is a parent of the gold type
- d^2 if its type is a grandparent of the gold type

etc.

1.4 File formats

1.4.1 `neleval` annotations format

Annotations provided as input to most `neleval` tools (e.g. *neleval evaluate*) consists of a tab-delimited file. Each line corresponds to an entity mention, and has the following columns:

document ID [str] Should not contain whitespace.

mention start offset [int] The units are arbitrary unless overlap aggregators are used (see *Aggregator*).

mention end offset [int] This should be inclusive of the last unit. Thus if offsets are character counts, a mention with text "Go" may have start offset 3 and end offset 4 (unlike Python slice notation).

entity ID [str] Should not contain whitespace. Should start with `NIL` for an arbitrary (cluster) identifier, or another string for a KB identifier.

score [float]

type [str] An entity type label

If there is more than one candidate, more (entity ID, score, type) column triples may be added, separated by tabs.

1.4.2 TAC data

The TAC entity linking data is available to participants in the entity linking track of NIST's knowledge base population shared task. The data format is described briefly below. For more details, see the entity linking task definition.

TAC 2014

In 2014, systems must provide two files: (1) an `xml` file containing entity mentions and (2) a `tab` file containing linking and nil clustering output.

Mention query XML

The mention `xml` file includes a query element for each mention. This element must have an `id` attribute with a unique value as well as `docid` (document identifier), `beg` (start offset), `end` (end offset) elements:

```
<kbpentlink>
    <query id="EDL14_ENG_TRAINING_0001">
        <name>Xenophon</name>
        <docid>bolt-eng-DF-170-181122-8792777</docid>
        <beg>22103</beg>
```

(continues on next page)

(continued from previous page)

```
        <end>22110</end>
    </query>
    <query id="EDL14_ENG_TRAINING_0002">
        <name>Richmond</name>
        <docid>APW_ENG_20090826.0903</docid>
        <beg>340</beg>
        <end>347</end>
    </query>
    ...
</kbpentlink>
```

Note that offsets should be character offsets over the utf8-encoded sgml source files. The end offset should be the last character that is included in the span.

Link ID file

The tab-separated link ID file includes a line for each mention. Each line includes several fields: `query_id` (matching the `id` attribute on a `query` element in the corresponding mentions `xml` file), `kb_or_nil_id` (a knowledge base or nil cluster identifier), `entity_type` (the type is required for 2014 link evaluation), and `score` (a confidence value, optional):

```
EDL14_ENG_TRAINING_0001    NIL0001     PER    1.0
EDL14_ENG_TRAINING_0002    E0604067    GPE    1.0
```

Note that it is possible to provide more than one response for a given mention by adding extra lines. However, the current set of evaluation measures only consider one response per mention (the one with the highest score).

TAC 2009-2013

Before 2014, the mention `xml` was provided and systems only need to output a tab-separated link ID file containing `query_id`, `kb_or_nil_id`, and `score` fields. To evaluate on these data sets, first add a `ne_type` field as per the 2014 format. Then use the gold `xml` file when converting system output to evaluation format with *neleval prepare-tac*.

Note that when using 2011 data, the end offset is the first character that is not part of the span (rather than the last character that is included in the span).

1.5 Command-line reference

`neleval` is mostly used through its command-line interface.

1.5.1 `neleval --help`: usage overview

```
$ neleval --help
usage: neleval [-h] [--verbose] [--quiet]
               {evaluate,validate-spans,list-measures,analyze,significance,confidence,
↪prepare-tac,prepare-tac15,prepare-brat,prepare-conll-coref,compare-measures,rank-
↪systems,plot-systems,compose-measures,to-weak,select-alternatives,weights-for-
↪hierarchy}
               ...
```

(continues on next page)

(continued from previous page)

```
Evaluation tools for Named Entity Linking output.

positional arguments:
  {evaluate,validate-spans,list-measures,analyze,significance,confidence,prepare-tac,
↪prepare-tac15,prepare-brat,prepare-conll-coref,compare-measures,rank-systems,plot-
↪systems,compose-measures,to-weak,select-alternatives,weights-for-hierarchy}
    evaluate            Evaluate system output
    validate-spans      Identify duplicate, crossing and nested spans
    list-measures       List measures schemes available for evaluation
    analyze             Analyze errors
    significance        Test for pairwise significance between systems
    confidence          Calculate percentile bootstrap confidence intervals
                        for a system
    prepare-tac         Convert TAC output format for evaluation
    prepare-tac15       Convert TAC 2015 KBP EL output format for evaluation
    prepare-brat        Convert brat format for evaluation
    prepare-conll-coref
                        Import format from CoNLL 2011-2 coreference shared
                        task for evaluation
    compare-measures    Calculate statistics of measure distribution over
                        systems
    rank-systems        Get filenames corresponding to best-ranked systems
    plot-systems        Summarise system results as scatter plots
    compose-measures    Adds composite measures rows to evaluation output
    to-weak             Convert annotations to char-level for weak evaluation
    select-alternatives
                        Handle KB ambiguity in the gold standard by modifying
                        it to match system
    weights-for-hierarchy
                        Translate a hierarchy of types into a sparse matrix of
                        type-pair weights

optional arguments:
  -h, --help            show this help message and exit
  --verbose
  --quiet
```

1.5.2 Evaluation and analysis of a single system

neleval evaluate

Evaluate system output

Usage summary

```
$ neleval evaluate --help
usage: neleval evaluate [-h] -g GOLD [-f {json,none,tab}] [-m NAME] [-b FIELD]
                        [--by-doc] [--by-type] [--overall]
                        [--type-weights FILE]
                        FILE

Evaluate system output
```

(continues on next page)

(continued from previous page)

```
positional arguments:
  FILE

optional arguments:
  -h, --help            show this help message and exit
  -g GOLD, --gold GOLD
  -f {json,none,tab}, --fmt {json,none,tab}
  -m NAME, --measure NAME
                        Which measures to use: specify a name (or group name)
                        from the list-measures command. This flag may be
                        repeated.
  -b FIELD, --group-by FIELD
                        Report results per field-value, and micro/macro-
                        averaged over these, Multiple --group-by may be used.
                        E.g. -b docid -b type. NB: micro-average may not equal
                        overall score.
  --by-doc              Alias for -b docid
  --by-type             Alias for -b type
  --overall             With --group-by, report only overall, not per-group
                        results
  --type-weights FILE   File mapping gold and sys types to a weight, such as
                        produced by weights-for-hierarchy
```

Evaluating each document separately

TODO

neleval analyze

Analyze errors

Usage summary

```
$ neleval analyze --help
usage: neleval analyze [-h] -g GOLD [-u] [-s] [-c] FILE

Analyze errors

positional arguments:
  FILE

optional arguments:
  -h, --help            show this help message and exit
  -g GOLD, --gold GOLD
  -u, --unique          Only consider unique errors
  -s, --summary         Output a summary rather than each instance
  -c, --with-correct    Output correct entries as well as errors
```

neleval significance

Test for pairwise significance between systems

Usage summary

```
$ neleval significance --help
usage: neleval significance [-h] -g GOLD [-n TRIALS] [--permute] [--bootstrap]
                            [-j N_JOBS] [-f {json,none,tab}] [-m NAME]
                            [--type-weights FILE] [--metrics METRICS]
                            FILE [FILE ...]

Test for pairwise significance between systems

positional arguments:
  FILE

optional arguments:
  -h, --help            show this help message and exit
  -g GOLD, --gold GOLD
  -n TRIALS, --trials TRIALS
  --permute             Use the approximate randomization method
  --bootstrap           Use bootstrap resampling
  -j N_JOBS, --n_jobs N_JOBS
                        Number of parallel processes, use -1 for all CPUs
  -f {json,none,tab}, --fmt {json,none,tab}
  -m NAME, --measure NAME
                        Which measures to use: specify a name (or group name)
                        from the list-measures command. This flag may be
                        repeated.
  --type-weights FILE   File mapping gold and sys types to a weight, such as
                        produced by weights-for-hierarchy
  --metrics METRICS     Test significance for which metrics (default:
                        precision,recall,fscore)
```

neleval confidence

Calculate percentile bootstrap confidence intervals for a system

Usage summary

```
$ neleval confidence --help
usage: neleval confidence [-h] -g GOLD [-n TRIALS] [-j N_JOBS]
                          [-p PERCENTILES] [--metrics METRICS] [-m NAME]
                          [--type-weights FILE] [-f {json,none,tab}]
                          FILE

Calculate percentile bootstrap confidence intervals for a system

positional arguments:
  FILE

optional arguments:
  -h, --help            show this help message and exit
  -g GOLD, --gold GOLD
  -n TRIALS, --trials TRIALS
  -j N_JOBS, --n_jobs N_JOBS
                        Number of parallel processes, use -1 for all CPUs
```

(continues on next page)

(continued from previous page)

```
  -p PERCENTILES, --percentiles PERCENTILES
                        Output confidence intervals at these percentiles
                        (default: 90,95,99)
  --metrics METRICS     Calculate CIs for which metrics (default:
                        precision,recall,fscore)
  -m NAME, --measure NAME
                        Which measures to use: specify a name (or group name)
                        from the list-measures command. This flag may be
                        repeated.
  --type-weights FILE   File mapping gold and sys types to a weight, such as
                        produced by weights-for-hierarchy
  -f {json,none,tab}, --fmt {json,none,tab}
```

1.5.3 Comparison of multiple systems' results

neleval compare-measures

Calculate statistics of measure distribution over systems

Usage summary

```
$ neleval compare-measures --help
usage: neleval compare-measures [-h] (-g GOLD | -e) [-f {plot,none,json,tab}]
                                [-o OUT_FMT] [--figsize FIGSIZE] [-m NAME]
                                [-s {none,name,eigen,mds}] [--cmap CMAP]
                                [--label-map LABEL_MAP]
                                FILE [FILE ...]

Calculate statistics of measure distribution over systems

positional arguments:
  FILE

optional arguments:
  -h, --help            show this help message and exit
  -g GOLD, --gold GOLD
  -e, --evaluation-files
                        System paths are the tab-formatted outputs of the
                        evaluate command, rather than system outputs
  -f {plot,none,json,tab}, --fmt {plot,none,json,tab}
  -o OUT_FMT, --out-fmt OUT_FMT
                        Path template for saving plots with --fmt=plot
                        (default: ./{}.pdf))
  --figsize FIGSIZE     The width,height of a figure in inches (default 8,6)
  -m NAME, --measure NAME
                        Which measures to use: specify a name (or group name)
                        from the list-measures command. This flag may be
                        repeated.
  -s {none,name,eigen,mds}, --sort-by {none,name,eigen,mds}
                        For plot, sort by name, eigenvalue, or
                        multidimensional scaling (requires scikit-learn)
  --cmap CMAP
  --label-map LABEL_MAP
```

(continues on next page)

(continued from previous page)

```
                        JSON (or file) mapping internal labels to display
                        labels
```

neleval rank-systems

Get filenames corresponding to best-ranked systems

Usage summary

```
$ neleval rank-systems --help
usage: neleval rank-systems [-h] [-m NAME] [--metric NAME]
                            [--group-re GROUP_RE] [--short-names]
                            [--group-limit GROUP_LIMIT | --group-max GROUP_MAX]
                            [--limit LIMIT | --max MAX]
                            FILE [FILE ...]

Get filenames corresponding to best-ranked systems

    Given evaluation outputs, ranks the system by some measure(s), or
    best per name group.

    This is a useful command-line helper before plotting to ensure all have
    same systems.

positional arguments:
  FILE

optional arguments:
  -h, --help            show this help message and exit
  -m NAME, --measure NAME
                        Which measures to use: specify a name (or group name)
                        from the list-measures command. This flag may be
                        repeated.
  --metric NAME
  --group-re GROUP_RE   Rank systems within groups, where a system's group
                        label is extracted from its path by this PCRE
  --short-names         Strip common prefix/suffix off system names
  --group-limit GROUP_LIMIT
                        Max number of entries per group (breaking ties
                        arbitrarily)
  --group-max GROUP_MAX
                        Max rank per group
  --limit LIMIT         Max number of entries (breaking ties arbitrarily)
  --max MAX             Max rank
```

neleval plot-systems

Summarise system results as scatter plots

Usage summary

```
$ neleval plot-systems --help
usage: neleval plot-systems [-h] [--by-system | --by-measure | --single-plot]
                            [--scatter | --rows | --columns | --heatmap]
                            [--pr | --prf | --recall-only] [--lines]
                            [--cmap CMAP] [--limits LIMITS]
                            [-i {evaluate,confidence}]
                            [-o OUT_FMT | --interactive [SHELL] | --run-code
                            CODE] [--figsize FIGSIZE]
                            [--legend-ncol LEGEND_NCOL] [-m NAME]
                            [--ci CONFIDENCE] [--group-re GROUP_RE]
                            [--best-in-group [BEST_IN_GROUP]] [-s SORT_BY]
                            [--at-most AT_MOST] [--label-map LABEL_MAP]
                            [--style-map STYLE_MAP] [--anon]
                            FILE [FILE ...]

Summarise system results as scatter plots

positional arguments:
  FILE

optional arguments:
  -h, --help            show this help message and exit
  --by-system           Each system in its own figure, or row with --heatmap
  --by-measure          Each measure in its own figure, or row with --heatmap
                        (default)
  --single-plot         Single figure showing fscore for all given measures
  --scatter             Plot precision and recall as separate axes with
                        different markers as needed
  --rows                Show rows of P/R/F plots
  --columns             Show columns of P/R/F plots (default)
  --heatmap             Show a heatmap comparing all systems and measures
  --pr                  In rows or columns mode, plot both precision and
                        recall, rather than F1
  --prf                 In rows or columns mode, plot precision and recall as
                        well as F1
  --recall-only
  --lines               Draw lines between points in rows/cols mode
  --cmap CMAP
  --limits LIMITS       Limits the shown score range to the specified min,max;
                        or "tight"
  -i {evaluate,confidence}, --input-type {evaluate,confidence}
                        Whether input was produced by the evaluate (default)
                        or confidence command
  -o OUT_FMT, --out-fmt OUT_FMT
                        Path template for saving plots with --fmt=plot
                        (default: ./{}.pdf))
  --interactive [SHELL]
                        Open an interactive shell with `figures` available
                        instead of saving images to file
  --run-code CODE       Run the given Python code with `figures` available
                        instead of saving images to file
  --figsize FIGSIZE     The width,height of a figure in inches (default 8,6)
  --legend-ncol LEGEND_NCOL
                        Number of columns in legend; otherwise ensures at most
                        20
```

(continues on next page)

(continued from previous page)

```
  -m NAME, --measure NAME
                        Which measures to use: specify a name (or group name)
                        from the list-measures command. This flag may be
                        repeated.
  --ci CONFIDENCE       The percentile confidence interval to display as error
                        bars (requires --input-type=confidence
  --group-re GROUP_RE   Display systems grouped, where a system's group label
                        is extracted from its path by this PCRE
  --best-in-group [BEST_IN_GROUP]
                        Only show best system per group, optionally according
                        to a given measure
  -s SORT_BY, --sort-by SORT_BY
                        Sort each plot, options include "none", "name",
                        "score", or the name of a measure.
  --at-most AT_MOST     Show the first AT_MOST sorted entries
  --label-map LABEL_MAP
                        JSON (or file) mapping internal labels to display
                        labels
  --style-map STYLE_MAP
                        JSON (or file) mapping labels to <color>/<marker>
                        settings
  --anon                Hide system/team names
```

1.5.4 Task definition and metric meddling

`neleval list-measures`

List measures schemes available for evaluation

Usage summary

```
$ neleval list-measures --help
usage: neleval list-measures [-h] [-m NAME]

List measures schemes available for evaluation

optional arguments:
  -h, --help            show this help message and exit
  -m NAME, --measure NAME
                        Which measures to use: specify a name (or group name)
                        from the list-measures command. This flag may be
                        repeated.
```

List all predefined measures

```
$ neleval list-measures
The following lists possible values for --measure (-m) in evaluate,
confidence and significance. The name from each row or the name of a
group may be used.
```

(continues on next page)

(continued from previous page)

```
Name                                Aggregate           Filter           Key Fields
↪        In groups
============================        ============        =========        ==============        =======
b_cubed                             b_cubed             None             span
↪        all, all-coref, luo, tac11, tac14
b_cubed_plus                        b_cubed             None             span+kbid
↪        all, all-coref, tac11, tac14
entity_ceaf                         entity_ceaf         None             span
↪        all, all-coref, luo, tmp
entity_match                        sets                is_linked        docid+kbid
↪        all, all-tagging, cornolti, hachey
mention_ceaf                        mention_ceaf        None             span
↪        all, all-coref, luo, tac14, tmp
mention_ceaf_plus                   mention_ceaf        None             span+kbid
↪        all, all-coref
muc                                 muc                 None             span
↪        all, all-coref, luo
pairwise                            pairwise            None             span
↪        all, all-coref, tmp
strong_all_match                    sets                None             span+kbid
↪        all, all-tagging, tac09, tac11, tac14
strong_link_match                   sets                is_linked        span+kbid
↪        all, all-tagging, cornolti, hachey, tac09, tac11, tac14
strong_linked_mention_match         sets                is_linked        span
↪        all, all-tagging, cornolti, hachey
strong_mention_match                sets                None             span
↪        all, all-tagging, hachey, tac14
strong_nil_match                    sets                is_nil           span
↪        all, all-tagging, tac09, tac11, tac14
strong_typed_all_match              sets                None
↪        span+type+kbid        all, all-tagging, tac14
strong_typed_link_match             sets                is_
↪linked        span+type+kbid        all, all-tagging
strong_typed_mention_match          sets                None             span+type
↪        all, all-tagging, tac14
strong_typed_nil_match              sets                is_nil           span+type
↪        all, all-tagging
typed_mention_ceaf                  mention_ceaf        None             span+type
↪        all, all-coref, tac14
typed_mention_ceaf_plus             mention_ceaf        None
↪        span+type+kbid        all, all-coref

Default evaluation group: all

In all measures, a set of tuples corresponding to Key Fields is
produced from annotations matching Filter. Aggregation with `sets'
compares gold and predicted tuple sets directly; coreference
aggregates compare tuples clustered by their assigned entity ID.

A measure may be specified explicitly. Thus:
  strong_all_match
may be entered as
  sets:None:span+kbid

Available aggregates are:
- non-clustering: overlap-maxmax, overlap-maxsum, overlap-summax, overlap-sumsum, sets
- clustering: b_cubed, entity_ceaf, mention_ceaf, muc, pairwise, pairwise_negative
```

(continues on next page)

(continued from previous page)

```
Available filter and key fields: candidates, docid, eid, end,
is_first, is_linked, is_nil, kbid, link, score, span, start, type.

More fields can be stored dynamically by entering a candidate's type
as a JSON key-value mapping.
```

neleval compose-measures

Adds composite measures rows to evaluation output

Usage summary

```
$ neleval compose-measures --help
usage: neleval compose-measures [-h] [-o OUT_FMT] [-r RATIOS RATIOS]
                                [FILE [FILE ...]]

Adds composite measures rows to evaluation output

positional arguments:
  FILE

optional arguments:
  -h, --help            show this help message and exit
  -o OUT_FMT, --out-fmt OUT_FMT
                        Output path format (default overwrites input path),
                        e.g. {dir}/{base}.evaluation_with_ratios
  -r RATIOS RATIOS, --ratio RATIOS RATIOS
                        Create a ratio of two other measures named
                        <measure1>/<measure2>
```

neleval select-alternatives

Handle KB ambiguity in the gold standard by modifying it to match system

Usage summary

```
$ neleval select-alternatives --help
usage: neleval select-alternatives [-h] [-f FIELDS] -g GOLD FILE

Handle KB ambiguity in the gold standard by modifying it to match system

    The following back-off strategy applies for each span with gold standard
    ambiguity:

        * attempt to match it to the top candidate for that span
        * attempt to match it to the top candidate for any span in that
          document
        * attempt to match it to the top candidate for any span in the
          collection
```

(continues on next page)

(continued from previous page)

```
        * default to select the first listed candidate

    The altered gold standard will be output.

positional arguments:
  FILE                  Path to system annotations

optional arguments:
  -h, --help            show this help message and exit
  -f FIELDS, --fields FIELDS
                        Comma-delimited list of fields to match candidates at
                        the same span between system and gold. "*" will
                        require match on all fields; default is "eid".
  -g GOLD, --gold GOLD  Path to gold standard annotations
```

`neleval to-weak`

Convert annotations to char-level for weak evaluation.

Usage summary

```
$ neleval to-weak --help
usage: neleval to-weak [-h] FILE

Convert annotations to char-level for weak evaluation

    A better approach is to use measures with partial overlap support.

positional arguments:
  FILE

optional arguments:
  -h, --help  show this help message and exit
```

`neleval weights-for-hierarchy`

Translate a hierarchy of types into a sparse matrix of type-pair weights

See *Approximate type matching*.

Usage summary

```
$ neleval weights-for-hierarchy --help
usage: neleval weights-for-hierarchy [-h] [-d DECAY] FILE

Translate a hierarchy of types into a sparse matrix of type-pair weights

    Input is a JSON object mapping parents to children in the hierarchy.
    Output is a three-column TSV with:
```

(continues on next page)

(continued from previous page)

```
        * gold type
        * system type
        * weight

    The weights are assigned such that where the system type is an ancestor of
    the gold type with d edges between them, it will score (decay ** d).

positional arguments:
  FILE                  Path to hierarchy JSON

optional arguments:
  -h, --help            show this help message and exit
  -d DECAY, --decay DECAY
                        Decay value for systems selecting an ancestor of the
                        gold type
```

Converting JSON type hierarchy to weights

```
$ bash -c "\
neleval weights-for-hierarchy --decay 0.5 <( \
echo '{\"root\": [\"A\", \"B\"], \"A\": [\"A1\", \"A2\"], \"B\": [\"B1\"], \"B1\": [\
↪"B1i\"]}' \
) \
"
A         A1         0.500000
A         A2         0.500000
B         B1         0.500000
B         B1i         0.250000
root         A         0.500000
root         A1         0.250000
root         A2         0.250000
root         B         0.500000
root         B1         0.250000
root         B1i         0.125000
B1         B1i         0.500000
```

These weights can be applied to evaluation with *neleval evaluate*'s `--type-weight` option.

1.5.5 Data preparation and validation

`neleval validate-spans`

Identify duplicate, crossing and nested spans

Usage summary

```
$ neleval validate-spans --help
usage: neleval validate-spans [-h] [--duplicate {ignore,warn,error}]
                              [--crossing {ignore,warn,error}]
                              [--nested {ignore,warn,error}]
                              [FILE]
```

(continues on next page)

(continued from previous page)

```
Identify duplicate, crossing and nested spans

    Will output warnings or errors as determined by options.

positional arguments:
  FILE

optional arguments:
  -h, --help            show this help message and exit
  --duplicate {ignore,warn,error}
  --crossing {ignore,warn,error}
  --nested {ignore,warn,error}
```

neleval prepare-tac

Convert TAC output format for evaluation

Usage summary

```
$ neleval prepare-tac --help
usage: neleval prepare-tac [-h] -q QUERIES [-x EXCLUDED_SPANS] [-m MAPPING]
                           FILE

Convert TAC output format for evaluation

    queries file looks like:

        <?xml version="1.0" encoding="UTF-8"?>
        <kbpentlink>
          <query id="doc_01">
            <name>China</name>
            <docid>bolt-eng-DF-200-192451-5799099</docid>
            <beg>2450</beg>
            <end>2454</end>
          </query>
        </kbpentlink>

    links file looks like:

        doc_01        kb_A        GPE        0.95

positional arguments:
  FILE                  link annotations

optional arguments:
  -h, --help            show this help message and exit
  -q QUERIES, --queries QUERIES
                        mention annotations
  -x EXCLUDED_SPANS, --excluded-spans EXCLUDED_SPANS
                        file of spans to delete mentions in
  -m MAPPING, --mapping MAPPING
                        mapping for titles
```

neleval prepare-tac15

Convert TAC 2015 KBP EL output format for evaluation

Usage summary

```
$ neleval prepare-tac15 --help
usage: neleval prepare-tac15 [-h] [-x EXCLUDED_SPANS] [-m MAPPING] FILE

Convert TAC 2015 KBP EL output format for evaluation

    Format is single tab-delimited file of fields:

        * system run ID (ignored)
        * mention ID (ignored)
        * mention text (ignored)
        * offset in format "<doc ID>: <start> - <end>"
        * link (KB ID beginning "E" or "NIL")
        * entity type of {GPE, ORG, PER, LOC, FAC}
        * mention type of {NAM, NOM}
        * confidence score in (0.0, 1.0]
        * web search (ignored)
        * wiki text (ignored)
        * unknown (ignored)

positional arguments:
  FILE                  link annotations

optional arguments:
  -h, --help            show this help message and exit
  -x EXCLUDED_SPANS, --excluded-spans EXCLUDED_SPANS
                        file of spans to delete mentions in
  -m MAPPING, --mapping MAPPING
                        mapping of KB IDs to titles
```

neleval prepare-brat

Convert brat format for evaluation

Usage summary

```
$ neleval prepare-brat --help
usage: neleval prepare-brat [-h] [-m MAPPING] DIR

Convert brat format for evaluation

positional arguments:
  DIR                   directory containing .ann files

optional arguments:
  -h, --help            show this help message and exit
  -m MAPPING, --mapping MAPPING
                        mapping for titles
```

neleval prepare-conll-coref

Import format from CoNLL 2011-2 coreference shared task for evaluation

Note that CoNLL coreference is not the same as the CoNLL-AIDA named entity disambiguaiton annotations.

Usage summary

```
$ neleval prepare-conll-coref --help
usage: neleval prepare-conll-coref [-h] [--with-kb] [--cross-doc] [system]

Import format from CoNLL 2011-2 coreference shared task for evaluation

positional arguments:
  system

optional arguments:
  -h, --help   show this help message and exit
  --with-kb    By default all cluster labels are treated as NILs. This flag
               treats all as KB IDs unless prefixed by "NIL"
  --cross-doc  By default, label space is independent per document. This flag
               assumes global label space.
```

1.6 Convenience scripts for TAC KBP evaluation

The repository includes a number of convenience scripts to illustrate and automate common usage.

1.6.1 Basic evaluation and reporting

The basic evaluation scripts automate the following workflow:

1. convert the gold data to the evaluation tool format,
2. convert each system run output to the evaluation tool format,
3. evaluate each system run.

The following are written to the output directory:

- detailed evaluation report for each run (*.evaluation),
- summary evaluation report for comparing runs (00report.tab).

Usage for TAC14 output format:

```
./scripts/run_tac14_evaluation.sh \
    /path/to/gold.xml \              # TAC14 gold standard queries/mentions
    /path/to/gold.tab \              # TAC14 gold standard link and nil annotations
    /system/output/directory \       # directory containing (only) TAC14 system␣
↪output files
    /script/output/directory \       # directory to which results are written
    number_of_jobs                   # number of jobs for parallel mode
```

Usage for TAC13 output format:

```
./scripts/run_tac13_evaluation.sh \
    /path/to/gold.xml \              # TAC13 gold standard queries/mentions
    /path/to/gold.tab \              # TAC13 gold standard link and nil annotations
    /system/output/directory \       # directory containing (only) TAC13 system␣
↪output files
    /script/output/directory \       # directory to which results are written
    number_of_jobs                   # number of jobs for parallel mode
```

1.6.2 Analysis and confidence reporting

The analysis scripts automate the following workflow:

1. *run the basic evaluation*,
2. calculate confidence intervals for each system run,
3. count errors for each system run (nil-as-link, link-as-nil, wrong-link counts).

The following are written to the output directory:

- detailed evaluation report for each run (*.evaluation),
- summary evaluation report for comparing runs (00report.tab),
- detailed confidence interval report for each run (*.confidence),
- summary confidence interval report for comparing runs (00report.*),
- error type distribution for each run (*.analysis).

Usage for TAC14 output format:

```
./scripts/run_tac14_all.sh \
    /path/to/gold.xml \              # TAC14 gold standard queries/mentions
    /path/to/gold.tab \              # TAC14 gold standard link and nil annotations
    /system/output/directory \       # directory containing (only) TAC14 system␣
↪output files
    /script/output/directory         # directory to which results are written
```

Usage for TAC13 output format:

```
/path/to/gold.xml \              # TAC13 gold standard queries/mentions
/path/to/gold.tab \              # TAC13 gold standard link and nil annotations
/system/output/directory \       # directory containing (only) TAC13 system output␣
↪files
/script/output/directory         # directory to which results are written
```

1.6.3 Filtered evaluation

The filtered evaluation scripts automate the following workflow:

1. filter gold data to include a specific subset of instances,
2. filter each system run to include a specific subset of instances,
3. *run the basic evaluation over subset data*.

The following are written to an output directory for each subset:

- detailed evaluation report for each run (*.evaluation),

- summary evaluation report for comparing runs (00report.tab).

The following subsets/directorys are defined:

- PER - mentions with person entity type,
- ORG - mentions with organisation entity type,
- GPE - mentions with geo-political entity type,
- NW - mentions from newswire documents,
- WB - mentions from newsgroup and blog documents,
- DF - mentions from discussion forum documents,
- entity-document type combinations (PER_NW, PER_WB, PER_DF, ORG_NW, etc.).

Usage for TAC14 output format:

```
./scripts/run_tac14_filtered.sh \
    /path/to/gold.xml \              # TAC14 gold standard queries/mentions
    /path/to/gold.tab \              # TAC14 gold standard link and nil annotations
    /system/output/directory \       # directory containing (only) TAC14 system␣
↪output files
    /script/output/directory         # directory to which results are written
```

Usage for TAC13 output format:

```
./scripts/run_tac13_filtered.sh \
    /path/to/gold.xml \              # TAC13 gold standard queries/mentions
    /path/to/gold.tab \              # TAC13 gold standard link and nil annotations
    /system/output/directory \       # directory containing (only) TAC13 system␣
↪output files
    /script/output/directory         # directory to which results are written
```

1.6.4 Test evaluation on TAC 2013 data

The test evaluation script automates the following workflow:

1. *run the basic evaluation*,
2. compare evaluation output to official TAC13 results.

The following are written to the output directory:

- detailed evaluation report for each run (*.evaluation),
- summary evaluation report for comparing runs (00report.tab),
- copy of the official results sorted for comparison (00official.tab),
- a diff report if the test fails (00diff.txt).

Usage for TAC13 official results:

```
./scripts/test_tac13_evaluation.sh \
    /path/to/gold.xml \              # TAC13 gold standard queries/mentions
    /path/to/gold.tab \              # TAC13 gold standard link and nil annotations
    /system/output/directory \       # directory containing (only) TAC13 system␣
↪output files
```

(continues on next page)

(continued from previous page)

```
    /system/scores/directory \       # directory containing official score summary␣
↪reports
    /script/output/directory         # directory to which results are written
```

The gold data from TAC13 is distributed by LDC. When running the test evaluation script, provide: * `LDC2013E90_TAC_2013_KBP_English_Entity_Linking_Evaluation_Queries_and_Knowledge_Base_Links_1/data/tac_2013_kbp_english_entity_linking_evaluation_queries.xml`, * `LDC2013E90_TAC_2013_KBP_English_Entity_Linking_Evaluation_Queries_and_Knowledge_Base_Links_1/data/tac_2013_kbp_english_entity_linking_evaluation_KB_links.tab`.

The system data from TAC13 is distributed by NIST. When running the test evaluation script, provide: * `KBP2013_English_Entity_Linking_Evaluation_Results/KBP2013_english_entity-linking_runs`, * `KBP2013_English_Entity_Linking_Evaluation_Results/KBP2013_english_entity-linking_scores`.

CHAPTER 2

References

This project extends the work described in:

- Ben Hachey, Joel Nothman and Will Radford (2014), "Cheap and easy entity evaluation". In Proceedings of ACL.

It was used as the official scorer for Entity (Discovery and) Linking in 2014–:

- Heng Ji, Joel Nothman and Ben Hachey (2014), "Overview of TAC-KBP2014 Entity Discovery and Linking Tasks", In Proceedings of the Text Analysis Conference.
- Heng Ji, Joel Nothman, Ben Hachey and Radu Florian (2015), "Overview of TAC-KBP2015 Tri-lingual Entity Discovery and Linking Tasks", In Proceedings of the Text Analysis Conference.
- Heng Ji and Joel Nothman (2016), "Overview of TAC-KBP2016 Tri-lingual EDL and Its Impact on End-to-End KBP", In Proceedings of the Text Analysis Conference.

CHAPTER 3

Changelog