1-2. Antonyms – Tags and Tagged Files

Antonym candidates are manually tagged. The tagged information is saved and used for future releases. The definition of tags and tagging processes are described below.

1. Fields of Antonym Candidates

The candidate file has 10 fields, as shown in table 1 below. Candidates from different sources might need special instructions for tagging.

Field 1	Field 2	Field 3	Field 4	Field 5	Field 6	Field 7	Field 8	Field 9	Field 10
Ant1	EUI1	Ant2	EUI2	POS	Canon	Type	Negation	Domain	Source

Table 1. Fields in antonym candidate file.

2. Definition of Tags

The generic definitions of tags are described in this section.

1. Field 1 – Antonym-1

This field specifies the 1st antonym (base form) of the aPair. It is the base form of the antonym of Ant2 (which must have same POS), that is automatically generated by programs. For the current study, we restrain antonym to be single word (can not be a multiwords).

2.2 Field 2 – EUI-1

This field specifies the associated EUI for antonym-1.

2.3 Field 3 – Antonym-2

This field specifies the 2nd antonym (base form) of the aPair. It is automatically generated by programs.

- Negative antonyms are assigned to Ant2 by programs if the sources is LEX, SD or PD. However, Ant1 and Ant2 are assigned in alphabetical order by programs if the source is CC.
- If the EUI is [EUI_TBD], this means the associated antonyms are not in the Lexicon. Ant1 or Ant2 should be added to the Lexicon if they are valid words with the given POS.

2.4 Field 4 – EUI-2

This field specifies the associated EUI for antonym-2.

2.5 Field 5 – POS

This field specifies the Part-Of-Speech (category) of the aPair. Antonyms must have the same POS. Legal values include: [adj], [noun], [verb], [adv], [prep], [modal], [aux], [pron], [det] and [conj]. POS of [compl] is not included for aPairs. POS is automatically generated by program if it is available.

2.6 Field 6 – Canon

Canonical: the domains are generic. That is, central to human life and way of living across times and cultures.

- [Y] – canonical, only canonical antonyms will be used as aPairs.

- [N] – non-canonical
  - Type must be [NA], no need to change [TYPE_TBD] to [NA] manually. It will be updated automatically by programs.
  - Domain must be [DOMAIN_NONE], no need to change [DOMAIN_TBD] to [DOMAIN_NONE] manually. It will be updated automatically by programs.

2.7 Field 7 - Type

This field specifies the type of antonyms by their meaning in relation to negation. Bounded antonyms represent two endpoints on a domain. There is no middle ground for bounded antonyms, such as dead-alive. Unbounded antonyms, such as long-short, are unbounded in the sense that extreme values never reach an endpoint. In practice, we choose the criteria (listed below) of X ≠ not Y, Y ≠ not X over endpoints if there is a conflict. For example, always-never is tagged as [UB] even though they are two endpoints ([B]). Asymmetric bounded antonyms are pairs with one negative word and/or endpoint on a scale and one non-endpoint/negative word. For example, impossible (negative/endpoint) – possible (non-negative/endpoint). This allows us to apply bounded and asymmetric bounded antonyms in subterm substitution for concept mapping applications.

- [B]: bounded antonym, if X = not Y, Y = not X
- [UB]: unbounded antonym, if X ≠ not Y, Y ≠ not X
- [AB1]: asymmetric bounded, if X = not Y, Y ≠ not X, where X is the negative/endpoint
- [AB2]: asymmetric bounded, if Y = not X, X ≠ not Y, where Y is the negative/endpoint
- [NA]: Not applicable (not of any of above cases). This tag is used when Canonical is [N] or not [B|UB|AB1|AB2].

- Examples:
  - [B]: dead|alive|adj, false|true|adj, closed|open|adj, with|without|prep, asleep|awake|adj, irregularly|regularly|adv, emergency|nonemergency|noun, exclude|include|verb
  - [UB]: narrow|wide|adj, light|dark|adj, low|high|noun, sad|happy|adj, rich|poor|adj, careful|careless|adj, first|last|adv, form|destroy|verb
  - [AB1]: bad|good|adj, asleep|alert|adj, bumpless|bumpy|adj, chilly|warm|adj, colorless|colorful|adj, exactly|approximately|adv, silence|noise|noun, disbelieve|believe|verb
  - [AB2]: good|bad|adj, alert|asleep|adj, bumpy|bumpless|adj, warm|chilly|adj, colorful|colorless|adj, approximately|exactly|adj, noise|silence|noun, believe|disbelieve|verb
  - [NA]: Used in the tagging process when the antonym candidates are not a valid canonical antonym pair.
- Usage:
  APairs with tags of [B] and [AB1/AB2] can be used in subterm substitution for better recall. For example, in the above table:
  - not asleep = awake; not awake = asleep
  - bad = not good; however, good ≠ not bad

2.8 Field 8 – Negation

This field specifies the negation of an aPair:

- [N1]: true/strict negative, ant1 is the negative antonym
- [N2]: true/strict negative, ant2 is the negative antonym
- [BN1]: broadly negative, ant1 is the negative antonym
- [BN2]: broadly negative, ant2 is the negative antonym
- [O]: Otherwise

- Examples:
  - [N1]: exclude|include|verb, false|true|adj, lack|plenty|noun, never|always|adv
  - [N2]: include|exclude|verb, true|false|adj, plenty|lack|noun, always|never|adv
  - [BN1]: failure|success|noun, fake|real|adj, rarely|usually|adv, repel|attract|verb
  - [BN2]: success|failure|noun, real|fake|adj, usually|rarely|adv, attract|repel|verb
  - [O]: ask|reply|verb, relaxed|upset|adj, slow|quick|adv, student|teacher|noun

APairs with sources of LEX, SD and PD only have negations of N2|BN2 because negative antonyms are assigned to ant2 automatically by computer programs. However, aPairs with a source of CC could have negations of N1|N2|BN1|BN2 because ant1 and ant2 are arranged by alphabetical order by computer programs.

- Usage:

Negative antonyms can be used as negation detection cue words. For example, *unsuccessful*, *useless*, *without* are negation detection cue words (A sentence is detected as negative if it contains these cue words) from the above table.

2.9 Field 9 - Domain:

This field specifies the domain that is central to human life and way of living across times and cultures (generic). Domain is used to determine if an aPair is a canonical antonym. For example, color, temperature, existence, length, weight, speed, time, quality are legit domains; while chocolate, tea, fruit are not legit domains as shown in examples below.

- Canonical: black-white-color, hot-cold-temperature, dead-alive-existence, short-long-length, slow-fast-speed, slow-quick-time
- Non-Canonical: white-dark-chocolate, hot-iced-tea, dry-fleshy-fruit

Our goal is to keep the domain as small and generic as possible. A domain list will be generated automatically by the computer programs during the tag validation processes.

- [DOMAIN_NONE]: used when canonical is tagged as [N].

2.10 Field 10 – Source:

Sources are assigned by computer programs. This field specifies the source (model) from which the aPair is generated. Valid tags include:

- [LEX]: Lexical records with negative tags.
- [SD]: suffix derivation. (derived antonyms)

- [PD]: prefix derivation. (derived antonyms)
- [CC]: collocates in a corpus (co-occurring model from MEDLINE n-gram set)
- [SN]: semantic network (aPairs are found in a semantic network model)

- Examples:
    - [LEX]: always|never|adv, are|aren't|aux, can|can't|modal, either|neither|det, with|without|prep
    - [SD]: careful|careless|adj, cheery|cheerless|adj, clawed|claswless|adj
    - [PD]: able|unable|adj, regularly|irregularly|adv, smoker|nonsmoker|noun, approve|disapprove|verb
    - [CC]: rough|smooth|adj, low|high|adv, student|teacher|noun, accept|reject|verb
    - [SN]: quiet|loud|adj, approximately|exactly|adv, wealthy|poor|noun, yell|whisper|verb

Antonym candidates from Training and Test Set [TT] are re-assigned to above four sources.

3. Tag Validation - Program to verify/fix tags

A validation program is developed to:

- validate values of manual tags in all fields
- auto fix/assign (TYPE and DOMAIN) if the Canon = [N]
    - [TYPE_TBD] must be [NA]
    - [DOMAIN_TBD] must be [DOMAIN_NONE]

  The above two fields are assigned automatically by programs.
- check for new domain (set verbose option to false)

  Tagged candidate files from linguists must run through this validation program to ensure all tags are valid before updating the annual release tagged candidate file. The tagged file must be fixed until no errors are found.

3.1 Input: The linguist tagged file is put under

- ${SOURCE}/${YEAR}/output/candTagged/antCand${Source}.data.tbd.tagged.

3.2 Program – ValidateTaggedCand.java

```
Shell> cd  ${ANTONYM}/bin
Shell> GetAntonyms ${YEAR}
11|21|31|42
```

- Tagged file:
    - input - This file should be copied to ${SOURCE}/${YEAR}/output/candTagged/antCand${Source}.data.tagged.${N}
    - Use the tagged file from last year as the baseline for validation
    - output - The output fixed file of this program is ${SOURCE}/${YEAR}/output/candTagged/antCand${Source}.data.fixed.${N}
    - run the validation program until:

        - no difference (0) between input (antCandTtSet.data.tbd.tagged) and output (antCandTtSet.data.fixed).
        - there are no errors (0) found
    - Assign the final validated-fixed tagged file to: ${SOURCE}/${YEAR}/output/candTagged/antCand${Source}.data.tagged.${YEAR}

- The error msg verbose option is set to false at the very first run to avoid too many error messages. After the 1st run, all [XX_TBD] are automatically fixed by programs, then set verbose to true (in the program) to see the detail for manual fixes.

4. Update to the Annual Release Tagged File

The final validated-fixed file is then used to update the annual release tagged file.

4.1 Input:

- in file: ${SOURCE}/${YEAR}/output/candTagged/antCand${Source}.data.tagged.${YEAR}
- tag file: ${0.Antonym}/${YEAR}/input/antCand.data.tag
- domain file: ${0.Antonym}/${YEAR}/input/domain.data
- out file: ${0.Antonym}/${YEAR}/input/antCand.data.tag.updated

4.2 Program - UpdateAllTaggedFile.java

Shell> cd ${ANTONYM}/binls

Shell> GetAntonyms ${YEAR}
12|22|32|43

This program updates validated tagged antonym candidates from ${SOURCE} to the annual release tagged file. If the aPairs are double tagged from different sources, the program conducts the following actions:

- exclude the duplicates if the tags are the same
- send out an error message if the tags are different, this case requires manual fixes

4.3 Output:

- ${0.Antonym}/${YEAR}/input/antCand.data.tag.updated
- This file is renamed to antCand.data.tag.${YEAR} if it passes by visual check.
- The final antCand.data.tag.${YEAR} is used to generate annual release of antonyms and negation detection cue words.
- The candidate generation programs need to be re-run after this update until *.TBD (candidate file) is empty (which means all candidates are tagged).