Tutorial and Usage of the Phrase-level Sentiment Analysis Toolkit

Yongfeng Zhang, Tsinghua University
zhangyf07@gmail.com, http://yongfeng.me

Overview

This tutorial gives a brief overview to the phrase-level sentiment analysis toolkit. The system consists of six tasks:

pre: Data pre-processing

pos: Part of speech tagging (for English)

bg: Build the background corpus resource file (for English)

crawler: Crawl detailed product parameter

validate: Validation to the POS tagging tools

lexicon: Sentiment lexicon construction

eval: Evaluation

profile: Construct the profile files for the demo system

Each task can be executed separately with specific task file as input.

NOTE: The 'bg' command needs only to be conducted for once, and the background corpus file for English has been built successfully. This background file is domain-independent and can be used for any domain related sentiment lexicon construction.

Project Structure

The project is organized as follows:

- *conf*: Basic configuration files for some basic and system properties. (For more details please refer to Configuration in the following.)

  - system.properties: Configuration for basic properties. Please **DO NOT** move or rename it.
  - log.properties: Custom log4j configuration for logger as requirement, since the default logging system will only print log information on console. Please **DO NOT** move or rename it.
- **corpus**: Default directory of the input corpus for **Lexicon** task.
  - 2013.tv.gbk.seg: Prepared input corpus of for smart TV domain.
  - 2014.nus.utf.seg: Prepared input corpus of cosmetic (English)
- **debug**: Default directory of the debug outputs.
- **lexicon**: Default directory of the generated lexicon by **Lexicon** task. (also default directory of to-be-evaluate results for **Eval** task)
- *pool*: Manually annotation files for evaluation.
- **preset**: Pre-set properties of the configurable thresholds, paths and metrics. (For more details please refer to Configuration in the following.)
  - default.mapping: Default properties for POS tag mapping tools.
  - default.pattern: Default properties for involved patterns.
  - english.pattern: Default properties for involved patterns for English.
  - default.resource: Default properties for resource.
  - english.resource: Default properties for resource on English.
  - relax.threshold: Weak pre-set thresholds, supposed to be applied for camera corpus, as there are fewer noises in it.
  - strict.threshold: Strong pre-set thresholds, supposed to be applied for cosmetic corpus, as there are more noises in it.
- *res*: Resource files for the project
- **task**: Pre-set task files or their templates
  - 2014.nus.bg.task: pre-set file for background resource construction (need to be run for only once)
  - 2013.tv.pre.task: pre-set file for **pre** task, and it can be directly used.

- 2014.nus.pos.task: part-of-speech tagging of English reviews based on Stanford Parser.
- 2013.tv.crawler.task: pre-set file for **crawler** task, and it can be directly used.
- 2013.tv.validate.task: pre-set file for **validate** task, and it can be directly used.
- 2013.tv.lexicon.task: pre-set file for **lexicon** task, and it can be directly used.
- 2013.tv.eval.task: pre-set files for **eval** task, and it can be directly used.
- 2013.tv.profile.task: pre-set files for **profile** task, and it can be directly used.

Notice

- All bold-text items are default directories for particular usage and can be changed to alternative paths by modifying configuration files, but which is **NOT** recommended unless necessary.
- Italic-text directories should not be moved or renamed for executing correctly.
- It is strongly recommended that **DO NOT** modify the properties in **preset**. Instead, please write another configuration file inherited from the given pre-set configuration and set it up in task file.
- We trade off between precision and recall by setting different parameters, and all the tunable parameters are determined in the *.threshold file. However, for some parameters a higher value leads to higher precision and lower recall, while for other parameters a lower value may give the same effect. As a result, we provide two *.threshold files, i.e., relax.threshold and strict.threshold, the former provides quantitively more but lower quality entries, and the latter provides quantitively less but higher quality entries. One may compare between the parameter settings in the two files and thus to know the effect of each parameter, so as to construct the threshold file of one's own.

Configuration

- 2014.nus.bg.task
  - preprocessing.language=english

    //Specify the language to be English
  - background.corpus=${path.corpus}/bg.lower.seg

    //Specify the background corpus used to generate the background resource file
  - background.resource=${path.resource}/bg.eng.res

    //Specify the background resource file name to be generated
  - background.corpus.encoding=UTF8

    //Specify the background corpus file encoding
  - background.resource.encodng=UTF8

    //Specify the background resource file encoding
- 2013.tv.pre.task
  - preprocessing.raw.path=${raw.path}/2013.tv.gbk.raw

    //Defines the raw data
  - preprocessing.raw.charset=GBK

    //Defines encoding of the raw data
  - preprocessing.selected.path=${select.path}2013.tv.gbk.selected

    //Defines the destination file
  - preprocessing.selected.charset=GBK

    //Defines encoding of the destination file
- 2014.nus.pos.task
  - postagging.language=english

    //Specify the language to be English

- postagging.select.path=${select.path}/2014.nus.utf.select

  //The corpus file for part-of-speech tagging, these are the selected reviews which is the output of the pre-processing stage.

- postagging.select.charset=UTF8

  //The charset of the above file

- postagging.pos.path=${stanford.tagged.path}/2014.nus.utf.select.stanford.seg

  //The output file, which is the part-of-speech tagged file

- postagging.pos.charset=UTF8

  //The charset of the output file

- postagging.model.file=${path.resource}/english-bidirectional-distsim.tagger

  //The model file of the part-of-speech tagging tool, used by the Stanford parser

• 2013.tv.validate.task

  - validation.A.mapping.class=edu.thuir.sentires.buildin.ICTPOSMapping

    //The POS mapping class of POS tagging tool A

  - validation.B.mapping.class=edu.thuir.sentires.buildin.HLPOSMapping

    //The POS mapping class of POS tagging tool B

  - validation.source.charset=GBK

    //encoding of the source file

  - validation.output.charset=GBK

    //encoding of the output file

  - validation.A.result=${ictclas.tagged.path}/2013.tv.gbk.comment.ictclas.seg

    //The POS tagging result of POS tagging tool A

  - validation.B.result=${hl.tagged.path}/2013.tv.gbk.comment.HL.seg

//The POS tagging result of POS tagging tool B

- validation.source=${source.pathg}/2013.tv.gbk.comment

  //The preprocessed user reviews used

- validation.output=${output.path}/2013.tv.gbk.seg

  //The destination output file after cross-validation

- 2013.tv.lexicon.task

This is the main configuration file for lexicon construction, which mainly contains 4 parts:

- I/O configuration: In this part, all information of the input and output should be specified.

  Input:

  - task.io.corpus.file=${path.corpus}/2013.tv.gbk.seg

    //The input file, which is the POS tagged user reviews after cross-validation

  - task.io.corpus.charset=GBK

    //encoding of the corpus

  - task.io.feature.file=${feature.path}/manual.feature

    //The manually defined features

  - task.io.feature.charset=UTF8

    //encoding of the manually defined features

  - task.io.stop.feature.file=${feature.path}/stop.feature

    //The manually defined wrong features, features in this file will be removed from the feature word list if they are improperly extracted

  - task.io.stop.feature.charset=UTF8

    //encoding of the manually defined wrong features

  - task.io.stop.opinion.file=${opinion.path}/stop.opinion

    //the manually defined wrong opinions, opinion words in this file will

be removed from the opinion word list if they are improperly extractd.

- task.io.stop.opinion.charset=UTF8

  //encoding of the manually defined wrong opinions

Output:

- task.io.lexicon.file=${path.lexicon}/2013.tv.gbk.lexicon

  //The output file wanted

- task.io.lexicon.charset=GBK

  //encoding of the output file

- lowest.frequency = 4

  //feature word that the number of corresponding opinion words is lower than 4 will be filtered

- Metrics & Thresholds: In this part, all metrics and thresholds that are used in the approach should be specified.

  - include=${path.preset}/relax.threshold

    //defines the threshold file

- Resources: Specify all necessary resources location and other properties. It is strongly recommended to gather all configurations and organize them in one particular property file as it can be shared with other tasks.

  - include=${path.preset}/default.resource

    //integrated property file, which is the default property file

  - resource.stopword.path=stopword.res

    //The manually defined stop word file

- Morphology & POS: Specify custom POS parser and extracting patterns. Also, it is strongly recommended to gather all configurations and organize them in one particular property file as it can be shared with other tasks.

- include=${path.preset}/default.pattern

  // integrated feature word pattern file, which is the default pattern file.

- include=${path.preset}/default.mapping

  // The POS mapping class. This file specifies the default POS mapping class.

- 2013.tv.eval.task

This configuration file contains the configuration for final evaluation.

- evaluation.feature.lres=${path.pool}/lres/feature/2013.tv.lres

  //Specify the manually labeled feature words for feature precision evaluation.

- evaluation.feature.qrel=${path.pool}/qrel/en/2013.tv.qrel

  //Specify the manually extracted feature words for feature recall evaluation.

- evaluation.opinion.lres=${path.pool}/lres/fopair/2013.tv.gbk.lexicon.lres

  //The manually labeled feature-opinion pairs for opinion words evaluation.

- evaluation.final=${path.pool}/final/en/2013.tv.gbk.comment

  //Specify the user reviews used for calculating word frequency.

- evaluation.dict=${path.lexicon}/2013.tv.gbk.lexicon

  //Specify the final dictionary file to be evaluated.

- 2013.tv.profile.task

This configuration file contains the configuration of profile file construction.

- profile.product = ${product.path}/2013.tv.gbk.product

  //This file contains all the products and their reviews from customers

- profile.lexicon = ${path.lexicon}/2013.tv.gbk.lexicon

//Specify the sentiment lexicon

- profile.posprofile = ${path.profile}/2013.tv.utf.pos.profile

  //Specify the output file for the positive profile

- profile.negprofile = ${path.profile}/2013.tv.utf.neg.profile

  //Specify the output file for the negative profile

- profile.index = ${path.profile}/2013.tv.utf.index

  //Specify the output file for the mapping of product to its URL

- profile.rerank.index = ${path.profile}/2013.tv.utf.rerank.index

  //In this file, we re-ranked the products by their model numbers

- profile.feature = ${feature.path}/2013.tv.gbk.label.feature

  //This file is a set of correct feature words labeled manually

- profile.opinion = ${opinion.path}/2013.tv.gbk.label.opinion

  //This file is a set of popular opinion words labeled manually

- profile.productname.toshiba = ${path.profile}/2013.tv.gbk.productname.toshiba

  //Products that are in the toshiba smart TV product list

- profile.productname.other = ${path.profile}/2013.tv.gbk.productname.other

  //Other TV product that may also be taken into consideration

- profile.sbc2dbc = ${sbc2dbc.path}/sbc2dbc.dict

  //A mapping file of SBC cases to DBC cases

- *.charset = GBK

  //encoding type of corresponding file

Usage

After deployment, all tasks can be executed in the **dist** directory.

Usage:

java -jar thuir-sentires.jar -t [task] -c [conf]

-c , –conf [arg]

Configuration for the task. Pre-set configuration files can be found in **'task'**.

-t , –task [arg]

Tasks to be executed.

Available tasks are

pre: data pre-processing.

validate: cross validation of two different POS tagging tools.

lexicon: build lexicon with prepared data.

eval: evaluate lexicon results.

-h , –help

Help

For English version:

java -jar thuir-sentires.jar -t [task] -c [conf]

Sentires v1.0e: Sentiment Resources Extractor from English Reviews.

-c,--conf <arg>

Configuration for the task. Preset confs can be found in './task'.

-h,--help

Help.

-t,--task <arg>

Tasks to be executed.

Available tasks:

pre: data pre-processing.

pos: part of speech tagging.

validate: cross validation of two different POS tagging tools.

bg: build the background corpus resource file.

lexicon: build lexicon with prepared data.

eval: evaluate lexicon results.

profile : construct product profiles.

crawler : crawl parameters of products from website.

Please make sure to execute with arguments: -t [task] -c [conf]

The following steps show how to execute a lexicon task.

1. Assume that current path is the base directory of released application (or **dist** directory in project).
2. There should be a jar file named thuir-sentires.jar in this directory.
3. Use command ”java -jar **-Xmx2048m** thuir-sentires.jar -t **lexicon** -c **task/2013.tv.lexicon.task**” to execute **lexicon** task with properties in **task/2013.tv.lexicon.task**. The task will run on JVM with 2G memory, and it is recommended to execute the program with at least 2G memory.

Important Notice

This project is provided for academic use and is protected by the *Law of Patent* in China, Japan, US, and Singapore. For any commercial usage, involvement, or cooperation, please contact Yongfeng Zhang at zhangyf07@gmail.com.