LinguisticAnalyzer Documentation

Release 1.0

Author

Dec 12, 2017

Contents:

CHAPTER 1

Keyword module

class Keyword.**Keyword**(*nWord=''*, *nType=0*, *nSal=0*, *nFreq=0*, *nKeyscore=0*)

Bases: object

summary: Class that stores a specific keyword and it's associated information

classmethod averagedistancefromkeywordtokeyword(*keyword*)

Summary: averga distance from "word" OR SIMILAR KEYWORDS to parameter keyword AND ITS SIMILAR KEYWORDS

classmethod averagedistanceto(*word*)

"@summary: average distanceBetween main "word" and parameter word

classmethod determinesimilarity(*word*)

Summary: Checks if the given word is semantically similar to the main keyword @param word: @type word: @return: @rtype:

classmethod distancefromkeywordtonearestkeyword(*keyword*)

Summary: distance from "word" OR SIMILARKEYWORDS to parameter keyword OR ITS SIMILAR KEYWORDS @param word: @type word: @return: @rtype:

classmethod distancetonearest(*word*)

Summary: distance from main "word" to nearest instance of parameter word @param word: @type word: @return: @rtype:

isinsimilarlist(*word*)

Summary: Checks if a given word is in the "similarWords" list @param word: @type word: @return: @rtype:

classmethod issimilar(*passedWord*)

@summary: determines if the passed keyword is similar to (or exactly the same as) the main word in the class @param passedWord: word @type passedWord: string @rtype: bool

similarwordfrequency()

wordfrequency()

CHAPTER 2

KeywordList module

class KeywordList.**KeywordList**
Bases: object

calculateavgscores()
Summary: calculates a document's average score values.

existsinlist(*keyword_name*)
Summary: searches through the list of keywords and sees if any keywords shares the same Keyword.word.

Parameters **keyword_name** (*str*) – The keyword

Returns returns true if a keyword with keyword_name as Keyword.word exists in the list. False otherwise.

Return type bool.

getavgkeywordscore()
Summary: returns document's average keyword score.

getdocumentscore()
Summary: Returns document's score.

getindexofword(*keyword_name*)
Summary: returns index of a Keyword in the list of Keywords

getkeywordscore()
Summary: returns document's keyword score.

getyulesiscore()
Summary: returns document's Yule's i score.

getyuleskscore()
Summary: returns document's Yule's k score.

insertkeyword(*keyword*)
Summary: inserts new Keyword into Keyword list

Parameters **keyword** (*object.*) – an instance of the class keyword

Returns none.

Return type

CHAPTER 3

functionsv1 package

3.1 common_functions module

common_functions.**appendtokeywordlist**(*kList*, *newK*)
Summary: Checks for duplicate keywords and etc. etc. before potentially appending keyword to list @param kList: list of keywords. @type kList: @param newK: @type newK: @return: @rtype:

common_functions.**cleantext**(*text_list*)
Summary: @param textlist: a list of strings to remove strange characters from @type textlist: @return: @rtype:

common_functions.**createkeywordfromgoogleapientity**(*entity*, *file_text*)
Summary: Creates a Keyword from a single entity that is returned by the google API @param entity: google API response entity @type entity: google API response entity @param file_text: entire file's text @type file_text: list of strings @return: populated instance of Keyword class @rtype: Keyword

common_functions.**extractkeywordfromtxt**(*file*)
Summary: This function will extract keyword information from .txt file and place into KeywordList object @param file: location of .txt file @type file: .txt @return: void

common_functions.**extractmicrosoftdocxtext**(*file*, *testdownload_folder=None*)
Summary: Extracts text from any ".docx" document and returns it. @param file: doc file @type file: werkzeug filestorage @param testdownload_folder: path to test upload folder if necessary @type testdownload_folder: string @return: file's text @rtype: List[string]

common_functions.**extractpdftext**(*file*, *testdownload_folder=None*, *RegDoc=False*)
Summary: Extracts Text from PDF document referenced in given file argument @param file: the object containing the file's information @type file: fileStorage @return: list containing the text of the PDF @rtype: List[string]

common_functions.**geterrorpage**(*errtext='Unknown Error'*)
Summary: Populates error mpge with proper response and returns html @param errtext: text of error @type errtext: string @return: html page @rtype: string

common_functions.**getregulatorydoctext**(*filename*)
Summary: Looks in the RegulatoryDocuments folder for the file with the given file name and return's its text as

a list of string @param filename: name of file to open @type filename: string @return: list of string containing text of file @rtype: List[string]

common_functions.**getscorepage**(*kw_list*, *reg_kw_list*)
Summary: Returns html page that is populated with proper calculated Keyword, Comparison, and Yule's scores. @param kw_list: user document's keyword list @type kw_list: KeywordList @param reg_kw_list: regulatory document's keyword list @type reg_kw_list: KeywordList @return: html text @rtype: string

common_functions.**getwordfrequency**(*word*, *file_text*)
Summary: determines frequency of the given word in the file's text @param word: word to find freq. of @type word: string @param file_text: text of entire file @type file_text: list of string @return: frequency @rtype: int

common_functions.**homeCount**()

common_functions.**interpretexistingfile**(*regfilename*)
Summary: Parses, identifies keywords and analyzes content of chosen regulatory file document is being compares against. @param regfilename: name of file without file ending @type regfilename: string @return: list of kwywords that have been analyzed @rtype: KeywordList

common_functions.**interpretfile**(*file*, *localuploadfolder*)
Summary: Parses uploaded file's text, identifies keywords, analyzes keywords, and returns a list of Keyword Objects @param file: werkzeug filestorage object @type file: werkzeug filestorage object @param localuploadfolder: @type localuploadfolder: string @return: keywordlist @rtype: KeywordList

common_functions.**kwhighestfrequencies**(*keyword_list*)
Summary: Returns the top 10 most frequent Keywords in the user's uploaded file @param keyword_list: lis tof file's Keywords @type keyword_list: list of keywords @return: topkeywords @rtype: list of highest frequency Keywords

common_functions.**kwhighestkeyscores**(*keyword_list*)
Summary: Returnst the twn Keywords with the highest Keyword scores @param keyword_list: @type keyword_list: list of keywords @return: topkeywords @rtype: list of top Keywords

common_functions.**longstringtostringlist**(*longstring*, *strsize*)
Summary: This functions splits a long string "longstring" into strings of size "strsize" and returns a list of strs @param longstring: long string to parse through @type longstring: @param strsize: size of strings to populate list with @type strsize: int @return: @rtype:

common_functions.**outputkeywordtotext**(*keylist*)
Summary: This function will write Keywords from an analyzed document to a .txt file @param keylist: KeywordList object containing keywords from analyzed document @type object: KeywordList @return: void

common_functions.**plotkeywordfrequency**(*keyword_list1*, *keyword_list2*, *doc1name='doc1'*, *doc2name='doc2'*)
Summary: plots keyword score of most frequently used keywords. Pulls KWs from list1, compares against list2 @param keyword_list1: @type keyword_list1: KeywordList @param keyword_list2: @type keyword_list2: KeywordList @param doc1name: name of first document @type doc1name: string @param doc2name: name of second document @type doc2name: string

common_functions.**plotkeywordsalience**(*keyword_list1*, *keyword_list2*, *doc1name='doc1'*, *doc2name='doc2'*)
Summary: plots salience of most frequently used keywords. Pulls KWs from list1, compares against list2 @param keyword_list1: @type keyword_list1: KeywordList @param keyword_list2: @type keyword_list2: KeywordList @param doc1name: name of first document @type doc1name: string @param doc2name: name of second document @type doc2name: string

common_functions.**plotkeywordscores**(*keyword_list1*, *keyword_list2*, *doc1name='doc1'*, *doc2name='doc2'*)
Summary: plots keyword score of most frequently used keywords. Pulls KWs from list1, compares against list2 @param keyword_list1: @type keyword_list1: KeywordList @param keyword_list2: @type keyword_list2:

KeywordList @param doc1name: name of first document @type doc1name: string @param doc2name: name of second document @type doc2name: string

common_functions.**printStringList**(*textList*)
Summary: Helper function that prints a list of strins @param textList: file's text @type textList: List[string] @rtype: void

common_functions.**savefile**(*file*, *download_folder=None*)

common_functions.**stringlisttolonglongstring**(*string_list*)
Summary: Helper function to turn list of string into one long long string @param string_list: file's text @type string_list: List[string] @return: file's text @rtype: long string

3.2 analyze_functions module

analyze_functions.**calculatecomparisonscore**(*kw_list*, *reg_kw_list*)

Summary: Compares the calculated scores of the two documents and generates value based on that comparison

Parameters

- **kw_list** (KeywordList) – list of Keywords
- **reg_kw_list** (KeywordList) – list of Keywords

analyze_functions.**calculatekeywordscore**(*kw_list*, *file_text*, *kw*)
Summary: calculate a keyword score for a single keyword

Parameters

- **kw_list** (*list*) – all keywords
- **file_text** (*list of strings*) – file's entire text
- **kw** (Keyword) – keyword

Returns keyword score

Return type float

analyze_functions.**calculatescores**(*kw_list*, *file_text*)
Summary: Calculate Yule's k and i scores, and keywords scores for a given document

Parameters

- **kw_list** (KeywordList) – list of Keywords
- **file_text** (*List[string]*) – Text of file

analyze_functions.**calculateyulesscore**(*file_text*)
Summary: calculates Yule's K scores for givven keyword argument

Parameters

- **file_text** (*list*) – plain text of document
- **kw** (Keyword) – Keyword

analyze_functions.**declarelogger**()
Summary: Declares logger for the current session.

analyze_functions.**identifykeywords**(*file_text*)
Summary: Calls the Google NLP API to extract Keyword information from text

Parameters **file_text** (*str*) – text of document

Returns KeywordList object

Return type object

analyze_functions.**tokenize**(*tokenStr*)
Summary: Splits up string into individual tokens.

Parameters **tokenStr** (*string*) – a string of words

Returns tokens

Return type list

CHAPTER 4

analyze module

analyze.**analyzeText**(*fileText*)

analyze.**checkSimilarity**(*fileText*)
@param fileText: @type fileText: @return: @rtype:

analyze.**createObjects**(*fileText*)
@param fileText: @type fileText: @return: @rtype:

analyze.**scrapeText**(*fileText*)
@param fileText: @type fileText: @return: @rtype:

CHAPTER 5

app module

app.**analyze**()
Summary: Receives uploaded document and comparison document choice and executes logic to compare them. @return: Information regarding uploaded document's similarity to regulatory document @rtype: html

app.**comparisoninfo**()
Summary: Returns html page that describes the Linguistic Analyzer's Comparison Score

app.**getkwfreeqimage**()
Summary: Returns png image of a graph of most frequent keywords

app.**getkwsalienceimage**()
Summary: returns png image of a graph of top salience keywords

app.**getkwscoresimage**()
Summary: returns png image of a graph of keyword scores

app.**main**()
Summary: Home page of the Linguistic Analyzer API. In here the logger is initiated for the session and the main webpage "views/index.html" is returned to the browser. @return: Home page @rtype: html

app.**project**()
Summary: Returns an html page containing details about the Linguistic Analyzer project.

app.**yulesinfo**()
Summary: Endpoint: Returns html page that describes Yule's k and Yule's i scores

CHAPTER 6

unit_tests package

6.1 test_analyze module

class unit_tests.test_analyze.**TestAnalyze**(*methodName='runTest'*)
Bases: unittest.case.TestCase

test_analyze()
Summary: Tests the Analyze() function

6.2 test_extractmicrosoftdocxtext module

class unit_tests.test_extractmicrosoftdocxtext.**TestExtractmicrosoftdocxtext**(*methodName='runTe*
Bases: unittest.case.TestCase

test_extractmicrosoftdocxtext()
Summary: Tests the extractmicrosoftdoctet() function

6.3 test_extractpdftext module

class unit_tests.test_extractpdftext.**TestExtractpdftext**(*methodName='runTest'*)
Bases: unittest.case.TestCase

test_extractpdftext()
Summary: Tests the extractpdftext() function

6.4 test_pdfanddocxarereadthesame module

class unit_tests.test_pdfanddocxarereadthesame.**TestEnsurepdfanddocxarereadthesame**(*methodName*
Bases: unittest.case.TestCase

test_ensurepdfanddocarereadthesame()

Summary: tests whether extractpdftext() and extractdocxtext() return the same exact information when given the same document in different formats

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

c

k

u

Index