**Code Catalog**

Link to Repo: <https://github.com/vivianofsouza/linguistic-distance/blob/main/lcp_python_libs/>

**Classes**

Calculator.py

*Handles calculations of L-distance and averages.*

* voidPrintAvg(self,val\_list)
* voidPrintLDScoresDir(self,strDirPath)
  + Performs l-distance calculation on .csv files for an entire directory
* **voidPrintLDScoreFile(self,strLangA,strLangB,strFilePath)**
  + Performs l-distance calculation on one .csv file

Parser.py

*Handles most of the clean up, stemming, and devowelization of words. Has multiple methods so you can switch out which filters you want on the text.*

* strRemoveNumbers(self, strA)
* strRemovePunct(self, strA)
* listRemoveNumbersAndPunct(self, listStrList)
* strRemoveNumbersAndPunct(self,strA)
* strOnlyConsonants(self,strA)
* listStemList(self, listWordList,strLangName)
* strStemWord(self, aWord, languageStr)
* listStemList(self, aWordList, languageStr):
* listRemoveCommonWordsFromString(self,strA,strLanguageA):
* **listCleanSample(self,sampleStr, languageStr)**
  + Will remove numbers, punctuation, consonants, common words, and stems

LDistance.py

*Contains algorithms to calculate l-distances*

* floatLDistance(self,strA, strB)
* listMeasureSamples(self,listLangA, listLangB, listAlignment)
* listMeasureSamplesManyMany(self, listLangA,listLangB,dictAlignment)
* listMeasureSamplesOneMany(self,listLangA, listLangB, dictAlignment)

SheetMaker.py

*Contains methods to create, write, and read from our CSV files*

* listCreateWordWrapperList(self,listLangList)
* **boolCheckBounds(self,intIndex,intStep,listListToCheck)**
  + Works with dfGenerateAlignments
  + Checks to see if an index and its window is either contained fully within the list’s range (0 to len(list))
    - This equals “both”
    - If not, the lower bounds of the window (index – step) could be contained in the list’s range of indices, equaling “lower”
    - If not the previous two possibilities, the upper bounds of the window (index + step) could be contained within the list’s range of indices, equaling “upper”
  + Used to ensure that alignments are only occurring within the list’s range
* boolCheckEqualWordWrappers(self,intI,intJ,listLangAWrappers,listLangBWrappers)
* strCreateAlignmentStr(self,intI,intJ,intCounter,strMarker)
* **dfGenerateAlignments(self,strLangA,strLangB,intStep,df)**
  + IntStep refers to the acceptable window of indices in which matches will be found. The window goes from –intStep to +intStep. Anything matches outside of these indices will be marked as uncertain
  + First, the two lists given as parameters are converted to WordWrapper lists. See above for definition of a WordWrapper
  + A loop runs through listA. BoolCheckBounds is used to determine the which indexes can be safely analyzed
    - Safely means within the bounds of the list (0 to len(list)). See boolCheckBounds to know what values will be looked at here.
    - When both the upper and lower parts of the window can be analyzed, a for loop runs from currIndex – step to currIndex + step
      * Between this range, the respective indices in listA and listB are compared. Here, the English translations are being compared
      * If an exact match is made, the alignment is written as ‘X|X’
      * If the word in one index is a component of the word in the other index, the alignment is made, but a “#” is used to mark uncertainty: ‘#X|X’
      * When alignment is finished, the two indexes are popped off of the their respective lists. WordWrapper keeps indexes in each list intact
      * GetOutOfLoop exception required to leave the loop and move on to next pair of indices
    - When only the lower or upper bounds of the window are safe to check, one may still do alignment, but it comes with possibility of uncertainty
      * This is denoted by use of “\*”. Each \* means that the word is n number of steps away. For example ‘\*\*X|X’ means the matched words were 15 steps away (original\_step + n\*step; 5+10)
  + Keep in mind that if matches can’t be found for a word, there will be no alignment. Space will be left empty for human alignment
  + At the end, a dataframe with the alignments is returned.
* **listGetLangTransToEng(self, listForeignWordList,strLangName)**
  + Translates a list of foreign words into English
  + Takes each word and sends it through Google Translate via API
    - Implements a 0.5 second delay after each 100 requests to prevent denial of service
    - Only one result returned; most likely the first result
    - If no results found, error message printed to console and that index of the list will have “nan” written to it
  + Can specify language with full name or two-letter codes (ex: english = “en”)
* **dfMakeAlignDf(self, strLangA,strLangB,listLangAWords,listLangBWords,listLangAEngTrans,listLangBEngTrans)**
  + Takes two lists in two languages, along with their translation, and inputs them into a dataframe
    - Dataframe created by putting the lists into a dictionary
    - Dictionary may contain padding values (spaces) to make columns for both languages an equal length
  + This dataframe will be used to create a .csv
  + No alignment happens here
* voidMakeTemplateAlignDf(self,strLangAName,strLangARawSample,strLangBName, strLangBRawSample)
* **voidDfToCSV(self,dfData,strFilePath)**
  + Takes a dataframe and writes it to a .csv file
  + You must specify where the .csv will be and what it will be called through the strFilePath parameter
* **voidPopulateAlignmentCSV(self,strLangA,strLangB,strFilePath)**
  + Reads the .csv file and converts it to dataframe using pandas
  + Sends this dataframe to dfGenerateAlignments(…)
  + The returned datagrame from dfGenerateAlignments(…) is then converted into .csv file of same name
* voidPopulateAlignmentCSVsFromDict(self, dictLangDict,strDirectory,intStep=5)

Util.py

*Handles miscellaneous utilities, such as graphing.*

* dfCreateBasicDf(self,aPairingList)
  + Creates a simple dataframe displaying two languages and their l-distance
* dfCreateCorrDf(self,aPairingList)
  + Creates the dataframe from which a heatmap can be generated
* snsCreateHeatMap(self,aPairingList)
  + Creates a heat map from a dataframe holding l-distances between all 5 languages. Spots of lighter color correlate to higher l-distanceaverages.

Steps

Cleaning + Translation

1. \_p.cleanSample(str, “lang”)
2. \_s.listGetLangTransToEng(listLangWords, "lang")

Alignment

1. \_s.dfMakeAlignDf(…) - makes empty df with empty alignment column
2. \_s.voidDfToCSV(df\_lang1\_lang2, “filename.csv”) - takes df made in step 1 and converts it to .csv
3. \_s.voidPopulateAlignmentCSV("lang1", "lang2", "filename.csv") - performs alignment using .csv file created in step 2

Calculation

1. \_c.voidPrintLDScoreFile("lang1", "lang2", " filename.csv ")

Graphing

1. snsCreateHeatMap(self,aPairingList)