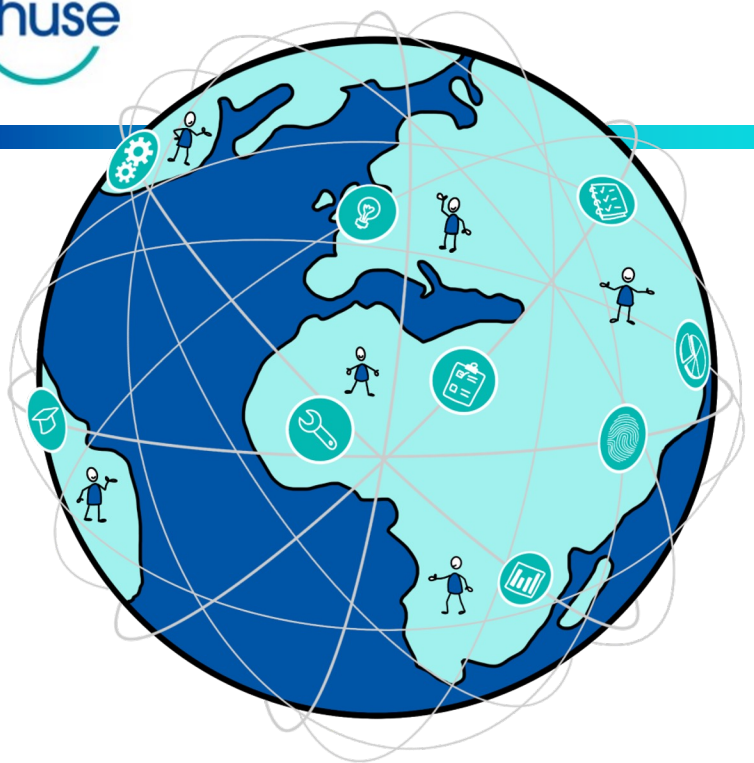


Detecting Code Plagiarism in Real-world Programming by Using Python

Jihao Yang, Sichuan Kelun Biotech Biopharmaceutical
Haiqiang Luo, Sichuan Kelun Biotech Biopharmaceutical

Shanghai, 2022.11.04

Disclaimer

The views and opinions expressed in this presentation are those of the authors and do not necessarily reflect the official policy or position of Sichuan Kelun Biotech Biopharmaceutical(科伦博泰).

Agenda

01 What’s Code Plagiarism in Clinical Programming

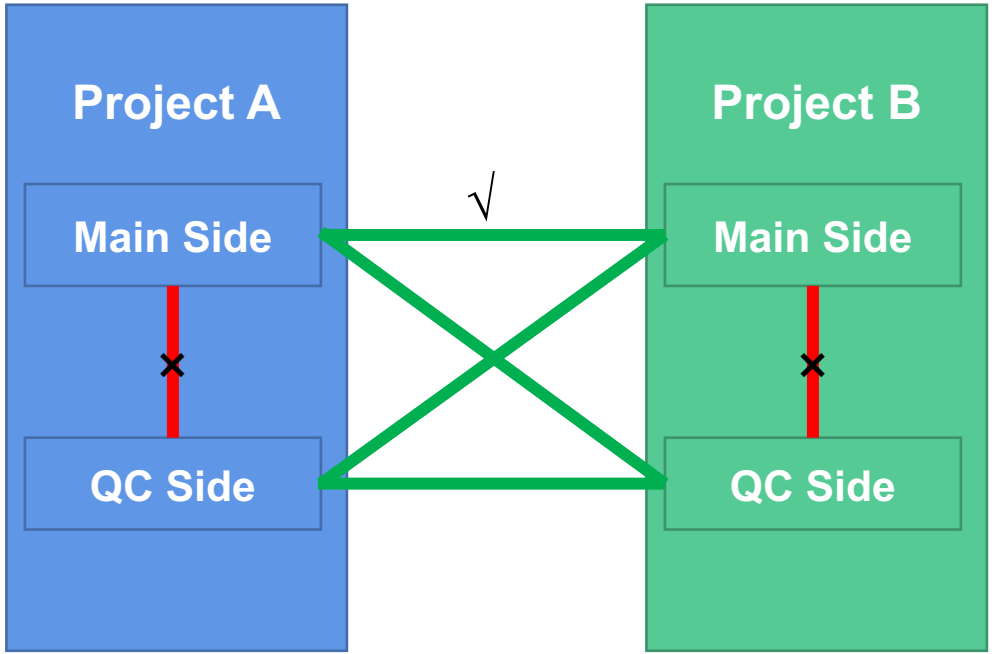
02 Why Detection

03 How to Detect

04 UI and Output

01 What's Code Plagiarism in Clinical Programming

Code Plagiarism: Copy code from other side in the same project and same task

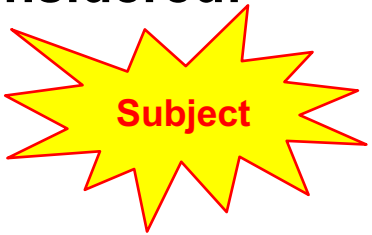


Code reference is acceptable!

02 Why Detection

There are two things need to be considered:

- **Where the clinical data come from?**



- **The purpose of the clinical trial?**

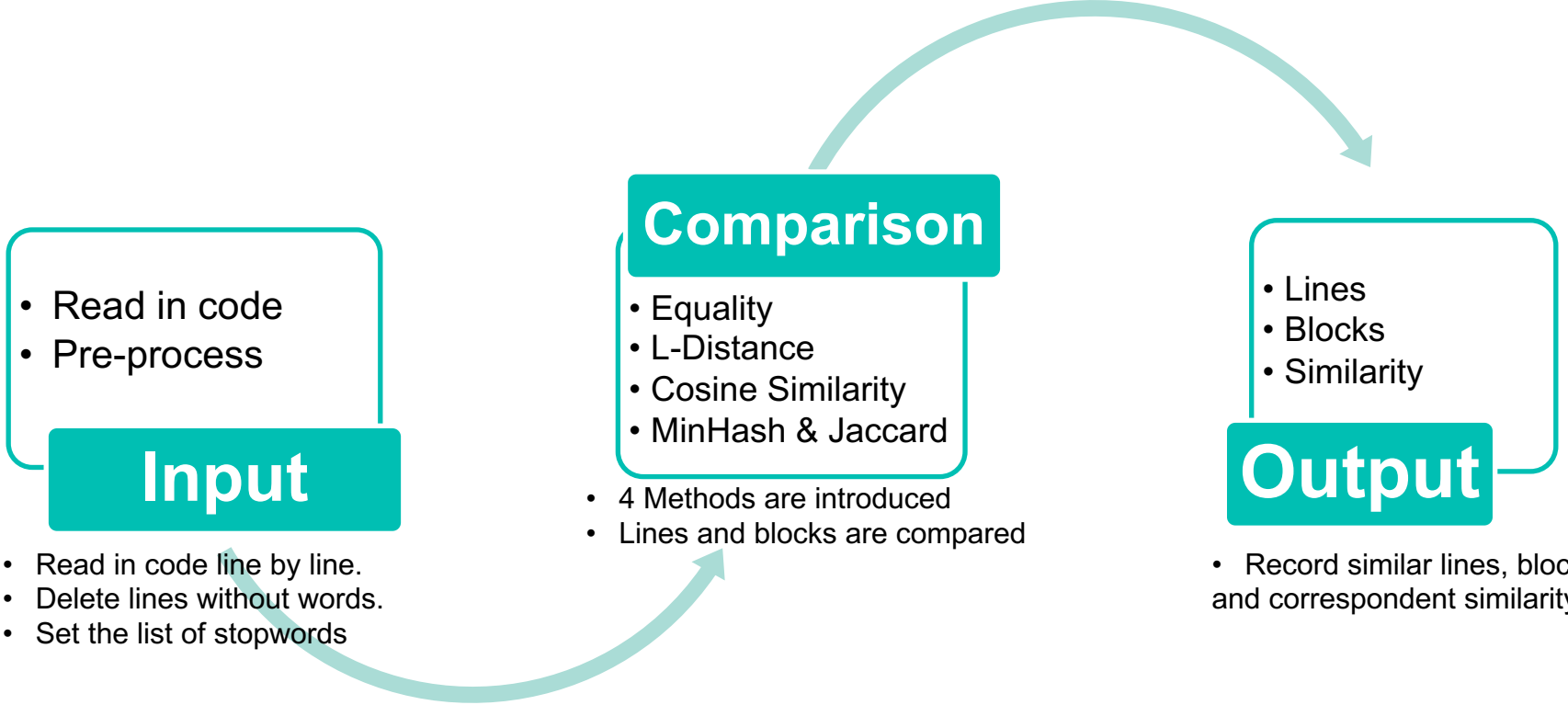
02 Why Detection

Independent double programming in Clinical Trial Analysis

- **Gold standard for the validation of programs and outputs created around the analysis of clinical data.**
  - ✓ Unbiased QC
  - ✓ Proven evidence “all values are exactly equal“ (Quality is more than Equality)
  - ✓ Validation can easily be repeated
  - ✓ 1st and 2nd line programming can be done in parallel
  - ✓ At least one programmer can be less experienced

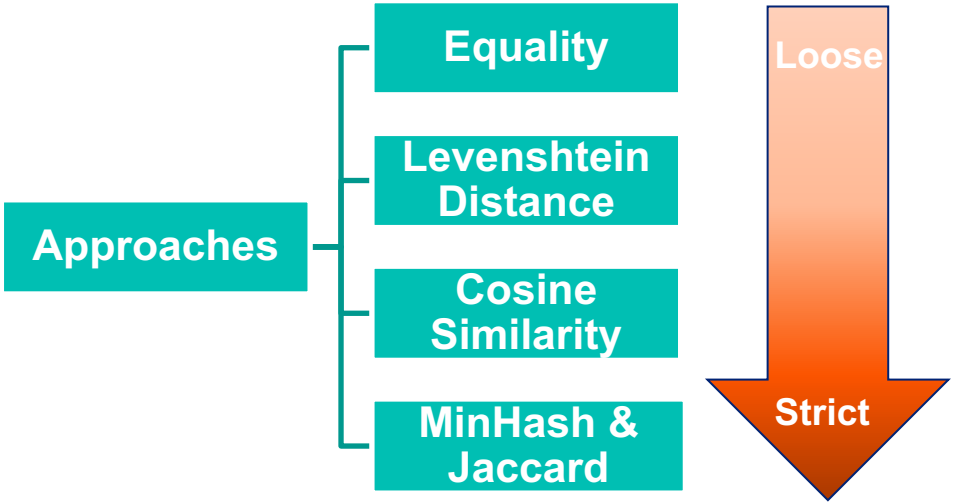
- **Applied not only for the validation of SDTM/ADaM but also for TFL.**

03 How to Detect



03 How to Detect

Four approaches are introduced to measure the similarity once the lines are well read in.



03 How to Detect - Equality

A simple brute force approach to compare two strings is to check if they are ***identical***.

Sample Code:

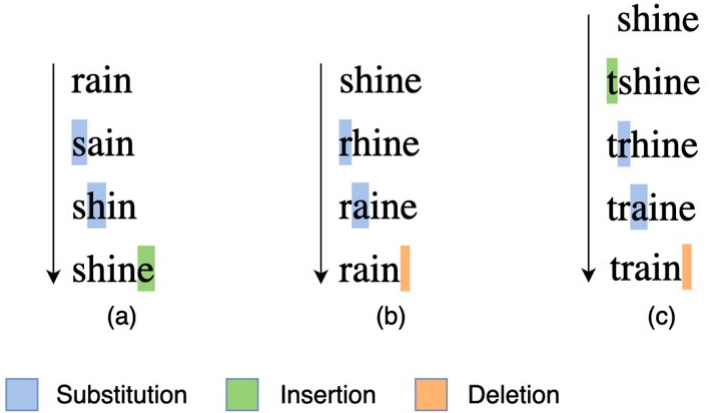
```
def equality(str1, str2):
    if str1.lower() == str2.lower():
        return 1
    else:
        return 0
```

03 How to Detect - Token Sort Ratio

Levenshtein distance is the minimum number of operations, including deletion, insertion and substitution, to make the two inputs equal.

For (a), at least three operations need to be done to get word ‘shine’ from word ‘rain’ :

operation 1: rain —substitute ‘r’ by ‘s’→ sain
operation 2: sain —substitute ‘a’ by ‘h’→ shin
operation 3: shin —insert ‘e’→ shine



In this case, we say that the Levenshtein distance between ‘rain’ and ‘shine’ is 3.

03 How to Detect - Token Sort Ratio (cont.)

One easily sees that $lev('a\ dog',' a\ cat') = 3$ and $lev('I\ have\ a\ dog','I\ have\ a\ cat') = 3$

To measure the similarity, the length of string should be taken into consideration.

We define the ***token sort ratio*** of a and b as

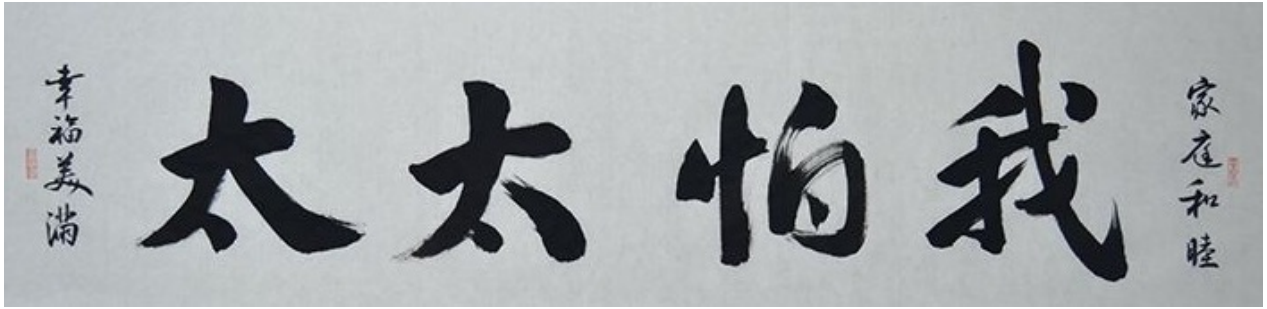
$$tsr(a,b) = \left(1 - \frac{lev(a.sort, b.sort)}{\min\big(len(a), len(b)\big)}\right) * 100\%$$

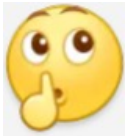
In this way, $tsr('a\ dog',' a\ cat') = 40\%$ and $tsr('I\ have\ a\ dog',' I\ have\ a\ cat') = 75\%$

03 How to Detect - Token Sort Ratio (cont.)

A method in FuzzyWuzzy provides a direct way to calculate. Sample codes:

```
from fuzzywuzzy import fuzz
def tsr(str1, str2):
  return fuzz.token_sort_ratio(str1.lower(), str2.lower())
```



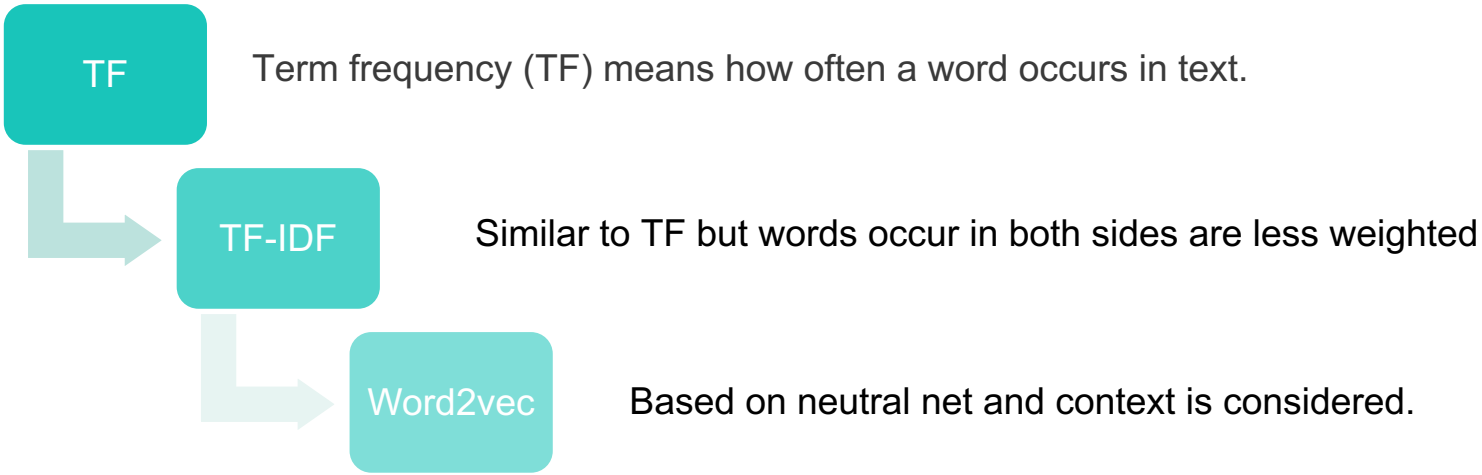


03 How to Detect - Cosine Similarity

Vectorization is widely used in Machine Learning and related fields.

Once transferred to vectors, similarity between strings can be understood as the intersection angle.

Here are three popular ways to vectorize strings.



03 How to Detect - Cosine Similarity (cont.)

Take TF as example

Say, we have two strings $a =' I\ have\ a\ red\ dog'$ and $b =' I\ have\ a\ red\ cat\ '$.
The union of words in two strings is {'I', 'have', 'a', dog', 'cat', 'red'}.

Calculate the frequencies of each word in the union occurs in a and b respectively.

	'I'	'have'	'a'	'dog'	'cat'	'red'
a	1	1	1	1	0	1
b	1	1	1	0	1	1

The correspondent vectors are $vec(a) = (1,1,1,1,0,1)$ and $vec(b) = (1,1,1,0,1,1)$

03 How to Detect - Cosine Similarity (cont.)

Sample codes for TF and output:

```
from sklearn.feature_extraction.text import CountVectorizer

sent1 = "the cat is walking in the bedroom."
sent2 = "the dog was running across the kitchen."

count_vec = CountVectorizer()

sentences = [sent1, sent2]
print(count_vec.fit_transform(sentences).toarray())
print(count_vec.get_feature_names())

vec_1 = count_vec.fit_transform(sentences).toarray()[0]
vec_2 = count_vec.fit_transform(sentences).toarray()[1]
```

```
[[0 1 1 0 1 1 0 0 2 1 0]
 [1 0 0 1 0 0 1 1 2 0 1]]
['across', 'bedroom', 'cat', 'dog', 'in', 'is', 'kitchen', 'running',
'the', 'walking', 'was']
```

03 How to Detect - Cosine Similarity (cont.)

Sample codes for TF-IDF and output:

```
from sklearn.feature_extraction.text import TfidfVectorizer

sent1 = "the cat is walking in the bedroom."
sent2 = "the dog was running across the kitchen."

tfidf_vec = TfidfVectorizer()

sentences = [sent1, sent2]
print(tfidf_vec.fit_transform(sentences).toarray())
print(tfidf_vec.get_feature_names())
vec_1 = tfidf_vec.fit_transform(sentences).toarray()[0]
vec_2 = tfidf_vec.fit_transform(sentences).toarray()[1]
```

```
[[0.         0.37729199 0.37729199 0.         0.37729199 0.37729199
  0.         0.         0.53689271 0.37729199 0.        ]
 [0.37729199 0.         0.         0.37729199 0.         0.
  0.37729199 0.37729199 0.53689271 0.         0.37729199]]
['across', 'bedroom', 'cat', 'dog', 'in', 'is', 'kitchen', 'running',
'the', 'walking', 'was']
```

Note: the *frequency* of word ‘the‘ is no longer two times as that of ‘bedroom’ since it appears in both strings, and is less weighted.

03 How to Detect - Cosine Similarity (cont.)

Sample codes for word2vec (sketch) and output:

```
# 设置词语向量维度
num_featrues = 300
# 保证被考虑词语的最低频度
min_word_count = 20
# 设置并行化训练使用CPU计算核心数量
num_workers = 2
# 设置词语上下午窗口大小
context = 5
downsampling = 1e-3
model = word2vec.Word2Vec(sentences, workers=num_workers,
size=num_featrues, min_count=min_word_count, window=context,
sample=downsampling)
model.init_sims(replace=True)
# 输入一个路径，保存训练好的模型，其中./data/model目录事先要存在
model.save("./data/model/word2vec_gensim")
# 加载模型
model = word2vec.Word2Vec.load("./data/model/word2vec_gensim")
# Train the model by more setences
model.train(more_sentences)
# 输出某一词汇的向量
print model['morning']
```

```
[  8.76819566e-02   8.43088701e-02   3.58341374e-02   8.08330402e-02
   1.33173089e-04  -1.86459888e-02  -6.91193044e-02  -1.32215414e-02
  -1.71450190e-02   3.09370253e-02   1.34692816e-02   4.38554808e-02
  -4.92837429e-02  -2.57787853e-02  -8.59034210e-02  -2.66618636e-02
   4.18011025e-02   1.30893528e-03  -2.42225397e-02  -6.90787956e-02
  -8.79226178e-02   3.50696519e-02  -6.64477274e-02   4.99707051e-02
  -1.46166412e-02   9.98077318e-02   3.56430002e-02  -2.23921277e-02
  -1.52790584e-02  -6.97031394e-02   4.94056083e-02   7.65942335e-02
  -4.64693233e-02   1.52456546e-02  -2.85607204e-02  -2.06128284e-02
  -1.55427994e-03   1.72603950e-02   2.10295618e-02  -1.03568301e-01
```

03 How to Detect - Cosine Similarity (cont.)

After vectorization, two vectors are obtained: $A = vec(a)$ and $B = vec(b)$.

Recall that $cos\theta = \frac{A \cdot B}{|A||B|}$. And we define this value to be the cosine similarity of two strings.

Back to our example where $a =' I\ have\ a\ red\ dog'$ and $b =' I\ have\ a\ red\ cat\ '$.

In this case $vec(a) = (1,1,1,1,0,1)$ and $vec(b) = (1,1,1,0,1,1)$, and easily we get the similarity $cos\theta = \frac{4}{5} = 80\%$.

Sklearn package provides direct methods to vectorize strings and to calculate cosine similarity.

03 How to Detect - Cosine Similarity (cont.)

Sample code for cosine similarity(sketch):

```
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import CountVectorizer
def cos_simi(str1, str2):
  crop = " ".join([str1.strip(), str2.strip()])
  vectorizer = CountVectorizer(token_pattern=r"(?u)\b\w\w+\b")
  vectorizer.fit([crop])
  main_vector = vectorizer.transform([str1.strip()])
  qc_vector = vectorizer.transform([str2.strip()])
  similarity1 = cosine_similarity(main_vector, qc_vector)[0][0]
  return similarity1
```

03 How to Detect - MinHash & Jaccard

In this final method, we are going to regard our strings simply as **sets**.

Jaccard Similarity Coefficient is defined as the size of their intersection divided by the size of their union.

In other words: $$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

For example, A = {‘I’, ‘have’, ‘a’, ‘dog’}, B = {‘I’, ‘have’, ‘a’, ‘cat’} then

$$J(A,B) = \frac{3}{5} = 60\%$$

03 How to Detect - MinHash & Jaccard (cont.)

As a bonus, we use ***MinHash*** to estimate Jaccard to raise efficiency.

MinHash lets us estimate the Jaccard similarity between sets of arbitrary sizes in ***linear time*** using a small and fixed memory space.

```
from datasketch import MinHash
a = 'I have a dog'
b = 'I have a dog and a cat'
m1, m2 = MinHash(), MinHash()
for i in a:
    m1.update(i.encode('utf8'))
for j in b:
    m2.update(j.encode('utf8'))
sim = m1.jaccard(m2)
print(sim)

int1 = len(list(set(a).intersection(set(b))))
uni = len(list(set(a).union(set(b))))
sim1 = float(int1/uni)
print(sim1)
```

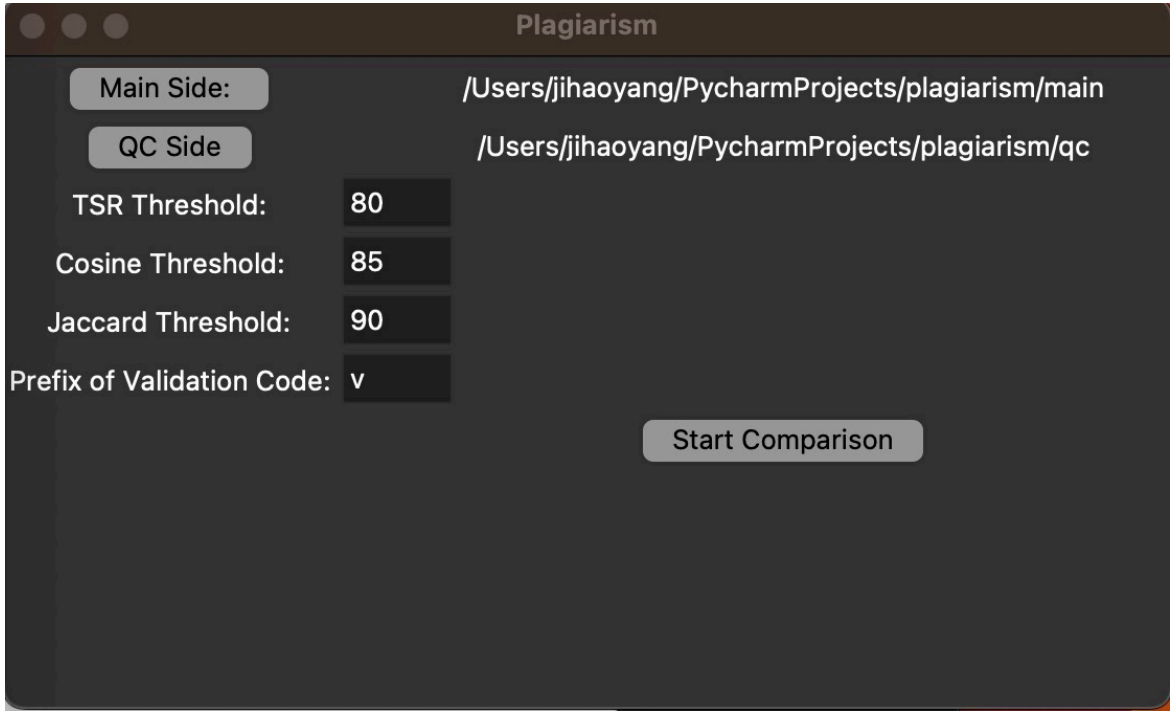
```
MinHash=  0.6796875
Jaccard= 0.75
```

04 Pack Up & Output

Object	Description	Related Module(s) in Python
Export	Every code-pair in the specified folders are compared. The result is exported to an excel where there is a sheet containing summarized information and individual sheets for every pair in details.	OS, Pandas
UI	In the User Interface Window, the folder path is selected through folder browser and threshold for every method is set up.	Tkinter
Pack Up	Pack everything up to obtain an executable file for convenience.	Pyinstaller

04 Pack Up & Output - Demo

UI:



04 Pack Up & Output - Demo

Output:

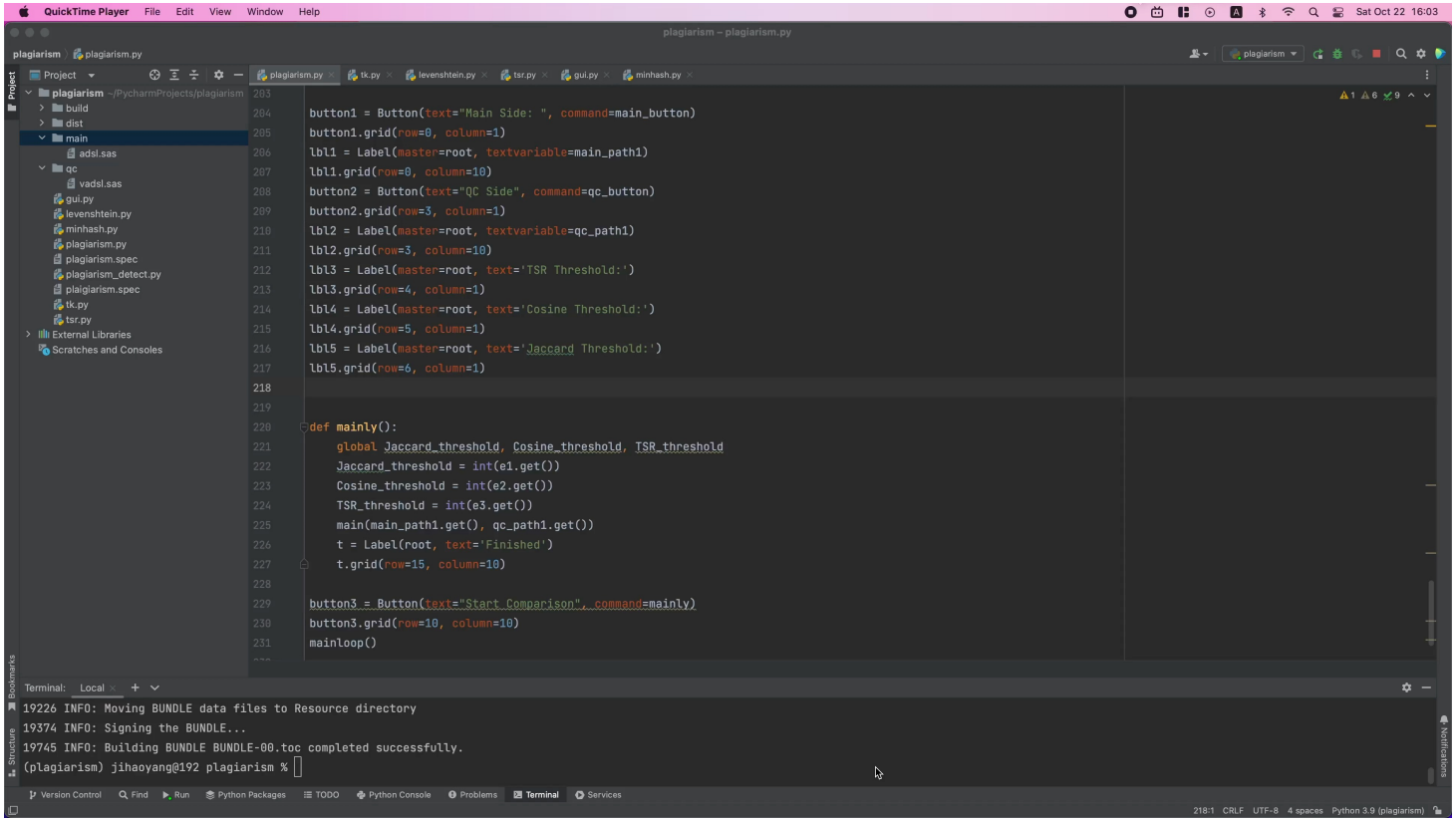
A	B	C	D	E	F	G	H	I	J	K	
M_NAME	M_LENGTH	Q_NAME	Q_LENGTH	Equality Lines	Equality Blocks	TSR Lines	TSR Blocks	Cosine Lines	Cosine Blocks	Jaccard Lines	Ja
adsl	252	vadsl	440	8	1	20	1	12	1	21	

04 Pack Up & Output - Demo

Output:

A	B	C	D	E	F	G
Type	<M Line(s) vs. V Line(s)>	Main Side Code	QC Side Code			
Equality Bl	M24-38 vs. V110-124	end%macro rootpath; %global _rootpath	end%macro rootpath; %global _rootpath _pgmname; %local			
Equality LI	M26 vs. V112	%global _rootpath _pgmname;	%global _rootpath _pgmname;			
Equality LI	M29 vs. V115	%let _fpath=%sysfunc(getoption(sysin));	%let _fpath=%sysfunc(getoption(sysin));			
Equality LI	M31 vs. V117	%if %length(&_fpath)=0 %then %do;	%if %length(&_fpath)=0 %then %do;			
Equality LI	M32 vs. V118	%let _fpath=%sysget(SAS_EXECFILEPATH	%let _fpath=%sysget(SAS_EXECFILEPATH);			
Equality LI	M35 vs. V121	%let _rootpath=%ksubstr(&_fpath., 1, %(	%let _rootpath=%ksubstr(&_fpath., 1, %eval(%kindex(%upcas			
Equality LI	M36 vs. V122	%let _pgmname=%scan(&_fpath., -2, .\);	%let _pgmname=%scan(&_fpath., -2, .\);			
Equality LI	M160 vs. V261	length her2type $200;	length HER2TYPE $200;			
Equality LI	M190 vs. V243	where cmcat="其他抗肿瘤治疗";	where CMCAT="其他抗肿瘤治疗";			
TSR LINE	M20 vs. V14	DM 'OUTPUT; CLEAR; LOG; CLEAR;';	dm 'log ; clear; output; clear;';			
TSR Block	M24-38 vs. V110-124	end%macro rootpath; %global _rootpath	end%macro rootpath; %global _rootpath _pgmname; %local			
TSR LINE	M26 vs. V112	%global _rootpath _pgmname;	%global _rootpath _pgmname;			
TSR LINE	M29 vs. V115	%let _fpath=%sysfunc(getoption(sysin));	%let _fpath=%sysfunc(getoption(sysin));			
TSR LINE	M31 vs. V117	%if %length(&_fpath)=0 %then %do;	%if %length(&_fpath)=0 %then %do;			
TSR LINE	M32 vs. V118	%let _fpath=%sysget(SAS_EXECFILEPATH	%let _fpath=%sysget(SAS_EXECFILEPATH);			
TSR LINE	M35 vs. V121	%let _rootpath=%ksubstr(&_fpath., 1, %(	%let _rootpath=%ksubstr(&_fpath., 1, %eval(%kindex(%upcas			
TSR LINE	M36 vs. V122	%let _pgmname=%scan(&_fpath., -2, .\);	%let _pgmname=%scan(&_fpath., -2, .\);			
TSR LINE	M95 vs. V58	brthdt=input(brthdtc,??yymmdd10.);	BRTHDT=input(BRTHDTC,yymmdd10.);			
TSR LINE	M96 vs. V59	rficdt=input(rficdtc,??yymmdd10.);	RFICDT=input(RFICDTC,yymmdd10.);			
TSR LINE	M107 vs. V76	trtsdtm=input(rfxstdtc,??e8601dt19.);	TRTSDTM=input(RFXSTDTC,E8601DT19.);			
TSR LINE	M109 vs. V82	trtedtm=input(rfxendtc,??e8601dt19.);	TRTEDTM=input(RFXENDTC,E8601DT19.);			
TSR LINE	M110 vs. V106	lstalvdt=input(rfpendtc,??yymmdd10.);	LSTALVDT=input(RFPENDTC,yymmdd10.);			
TSR LINE	M124 vs. V158	eosdt=input(dsstdtc,??yymmdd10.);	EOSDT=input(DSSTDTC,yymmdd10.);			
TSR LINE	M129 vs. V179	eotdt=input(dsstdtc,??yymmdd10.);	/*EOTDT=input(DSSTDTC,yymmdd10.);*/			
TSR LINE	M129 vs. V188	eotdt=input(dsstdtc,??yymmdd10.);	EOTDT=input(DSSTDTC,yymmdd10.);			

04 Pack Up & Output - Demo



Q & A

Thank You!

Contact：

杨济豪 (Jihao Yang): yangjihao@kelun.com

罗海强 (Haiqiang Luo): luohq@kelun.com