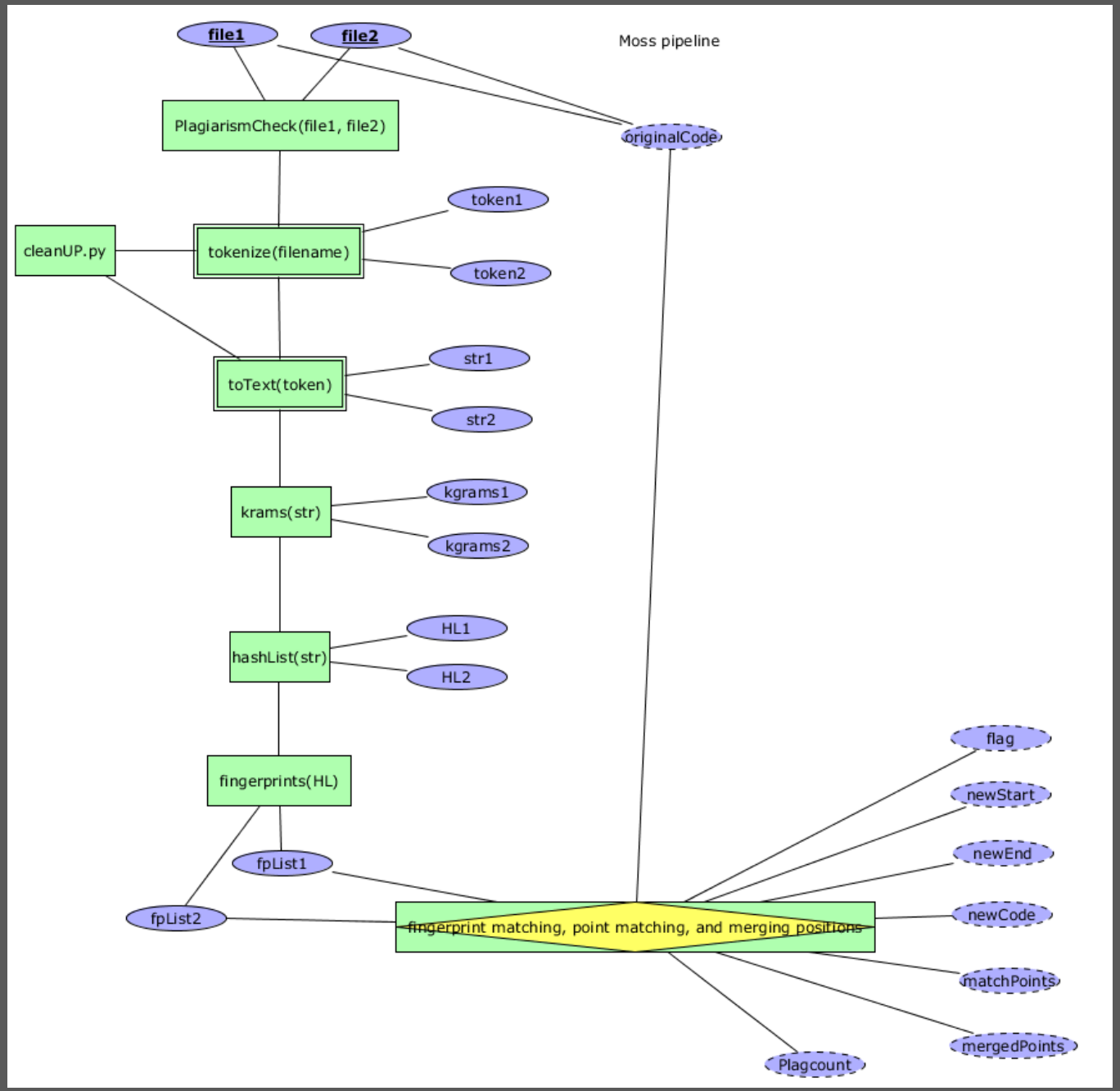
Moss Pipeline and how can we use their tool to improve ours?

Tracy Hotchkiss


Moss pipeline
file1
file2
PlagiarismCheck(file1, file2)
originalCode
token1
cleanUP.py
tokenize(filename)
token2
str1
toText(token)
str2
kgrams1
krams(str)
kgrams2
HL1
hashList(str)
HL2
flag
fingerprints(HL)
newStart
newEnd
fpList1
fpList2
fingerprint matching, point matching, and merging positions
newCode
matchPoints
mergedPoints
Plagcount

Tokenize(file1)

element, pos of each element in original code, pos of each element in cleaned up (processed) code

```
[(' ', 8, 0), (' ', 27, 1), (' ', 46, 2), ('}', 4269, 2493), ('\n', 4270, 2494)] |* token1
```

```
token1: list[Union[tuple[str, int, int], tuple[Any, int, int]]] = tokenize(file1)
```

toText(token1)

```
static void F(char * N, int N) {
    char N[64];
    strerror_r(N, N, 64);
    fprintf(N, SSSS, N, N);
    exit(1);
}

int F()
{
    int N, N, N, N = 1, N = 0;
    struct sockaddr_in N, N;

    if (0 == strcmp(N[1], SSS)) {
        if (N < 3) {
            printf(SSSS, N[0] );
            return 1;
        }

        return client(N[2]);
    }

    return server();
}
 /str1
```

```
str1: str = toText(token1)
```

kgrams(str1)

```
[('          \nstatic void ', 33072, 0, 25), ('          \nstatic void F', 2633, 1, 26), ('          \nstatic void F(', 52468, 2, 27),
('\n\t}\n\n\treturn server();\n}\n', 26250, 2470, 2495)] |* kGrams1
```

```
kGrams1: list[tuple[str, int, int, int]] = kgrams(str1)
```

hashList(kGrams1)

```
[33072, 2633, 52468, 35016, 16981, 60680, 46566, 54568, 24286, 62882, 24504, 26250] |* HL1
```

```
HL1: list = hashList(kGrams1)
```

fingerprints(HL1)

```
[2633, 16981, 14653, 39835, 111, 247, 34050, 30563, 35257, 17671, 26734, 19574, 24286] |* fpList1
```

```
fpList1: list = fingerprints(HL1)
```

Comparison, points, and mergedPoints

```
[[428, 472], [436, 475], [438, 477], [439, 478], [483, 484], [4241, 4266], [4243, 4268]] |* points
```

```
points: list[list[Union[str, int]]] = []
```

```
[[428, 478], [483, 484], [488, 527], [611, 529],  [4203, 4187], [4212, 4268]] |* mergedPoints
```

```
mergedPoints: list[Union[list[Union[str, int]], list[Union[str, int]]]] = []
```

How does our plagiarism tool compare? How can we implement a method to highlight with precision instead of a whole line?

our preprocessing function works like moss in that it takes the original code and tokenizes it. It does not store each elements position relative to the original code and the processed code

Our preprocessing folder has the function translateLines(StrippedFile, OldLines, SourceFile) which will translate the lines from processed to original. For example, if it is given lines 2,3,5 of the strippedFile, then it will output 4,5,11 as lines for the original code. As is, it is not usable without a method to pull the info needed. Currently this functions use is demonstrated with hard coded input and is not implemented in our preprocessing code.

```
CombinedTesting.preprocessing.translate
def TranslateLines(StrippedFile: Any,
                   OldLines: {__getitem__},
                   SourceFile: Any) -> list[int]
```

We can pull a list of line numbers from our winnow function and convert it to a single string, but need a string variable that holds the name of the created StrippedFile.py from the preprocessing output. This can easily be added; however, this would only allow us to highlight whole lines and not partial as we need to add position tracking.

I propose we add position tracking into our preprocessing by adding a count1 and count2 that will store the pos in processed file and pos in original file like moss does theirs. This seems easiest method to do precision highlighting.

Proposed changes that should be visited