

LHDiff: tracking lines across file versions

Actual command-line execution on the repository dataset, followed by the team's original design and report samples.

Run: PYTHONHASHSEED=0 python3 LHDiff/LHDiff.py

our-dataset/file17-old.txt our-dataset/file17-new.txt

```
1 -> 1
2 -> 12
3 -> 13
4 -> 3
5 -> 4
6 -> 5
7 -> 6
8 -> 7, 8
9 -> 7, 8
10 -> 15
11 -> 16, 17
12 -> 16, 17
```

```
# Unmatched deletions (only in OLD file):
(none)
```

```
# Unmatched additions (only in NEW file):
NEW 2
NEW 9
NEW 10
NEW 14
```

My contribution

Contributed across algorithm design and Python implementation, preprocessing and comparisons, test cases and expected mappings, interface mockups, and the group report.

What this run shows

Line correspondences and unmatched additions for an included file pair. These are algorithm predictions, not independently verified accuracy results.

Course implementation informed by the published LHDiff research (Asaduzzaman et al., 2013).

Proposed change-visualisation interface

Original team mockup: a design concept, not a running graphical interface.

```
text = "hello world hello program"
words = text.split()
freq = {}

for w in words:
    if w not in freq:
        freq[w] = 1
    else:
        freq[w] += 1

print(freq)
```

```
def count_words(sentence):
    result = {}
    for token in sentence.split():
        if token in result:
            result[token] += 1
        else:
            result[token] = 1
    return result

text = "hello world hello program"
counts = count_words(text)
print(counts)
```

● Added ● Removed ● Moved ● Modified

Colours distinguish proposed additions, removals, movement, and modifications.

Original report excerpt: technique and evaluation

An excerpt from the team report showing preprocessing, matching, and split/merge heuristics.

Technique Description and Evaluation

Throughout the development of our model, we weighed several options on which is most effective and efficient for our model. Normalization which includes removing punctuation, ignoring blank lines, lowercasing everything, and collapsing whitespace was determined as an important first step. We determined exact matches using `find_unchanged`, and similar matches using similarity methods as previously discussed. To detect split lines, our model analyses the lines before and after the exact match of the new file. If the similarity between the old line and the neighbouring new line is greater than the split threshold, the new line is added to the mapping. To detect merged lines, our model works inversely of the line split method. It evaluates the lines before and after the exact match of the old file and if their similarity to the same new line exceeds the merge threshold, they are also mapped to that line. After several experimental trials, we found our model to be approximately 92% accurate. To calculate accuracy, we compared the mappings produced by LHDiff against manually verified ground truth and computed the ratio of correctly identified mappings over the total number of mappings.

The report's approximately 92% result concerns the team-created 25-pair dataset. It has not been independently reproduced here.

Code and full report: github.com/salmanyoussef/LHD