Task: Implementation of Weighted AllPairs

December 21, 2018

1 Task

In Task 1, you implemented a specialized version of the AllPairs algorithm. Each token contributed an equal amount to the total similarity, i.e., each token had a weight of 1. In Task 2, we generalize this such that each token may be associated with a different weight between 0 and 1.

1.1 Task 2

AllPairs can be modified to allow weighted similarity functions. Each token has a particular weight associated with it. The weight is the same for each occurrence of this token, e.g., token 14 in the input example above could have weight 0.1. It has this weight in all three sets it occurs in.

Your binary or script will therefore accept another input parameter:

```
./binary_or_script input_file weight_file jaccard_threshold
```

The weight file consists of a mapping of tokens to their weight. There is one mapping per line. Each line contains the token and the weight, separated by a colon. If a token is not mapped, 1 should be assumed as its weight. It may look like this:

```
3:0.1
2:0.2
14:0.1
```

The similarity function to implement is weighted Jaccard, which is defined as

$$J^W(r,s) = \frac{\sum_{w \in |r \cap s|} weight(w)}{\sum_{w \in |r \cup s|} weight(w)}$$

There are a few things to rethink:

1. The input set have to processed in a different order. In Task 1, the sets were processed in increasing order of their sizes. For the weighted AllPairs, this has to be changed. One option is to process sets in increasing order of their total weight.

2. The token ordering in each set has to be changed. Here, one option is to sort the tokens of a set by decreasing order of weight.

3. The lengths of probing and indexing prefix need to be adapted to take the token weights into account.

2 Further Readings

The weighted set similarity join is discussed in [1, p. 17].

References

[1] C. Xiao, W. Wang, X. Lin, J. X. Yu, and G. Wang. Efficient similarity joins for near-duplicate detection. *TODS*, 36(3):15, Aug. 2011.