**How it works**

What we plan to do is use a slightly modified form of Jaccard Index to find the similarity of products given in the two datasets. The Jaccard index measures similarity between finite sample sets and is defined as the size of the intersection divided by the size of the union of the sample sets.

For example,

Assume we have two products, “Indoor and outdoor foldable table for comfort and easy use” and “Home furniture indoor study table, foldable and easy to use”.

We first ignore the commonly used adjectives like ‘the’, ‘and’, ‘to’, etc. Then we use Jaccard index by calculating the number of words which appear in both divide by total number of words combined.

Therefore, the above given sentences have a coefficient of 0.5, which implies 50% similarity.

**Advantages**

* Because the number of items in the dataset are huge, we cannot use more complex algorithm as it can lead to worse run-time constraints. This should give a good balance between the result and run-time.
* This gives good results especially for this case where we compare similar products from two datasets.

**Disadvantages**

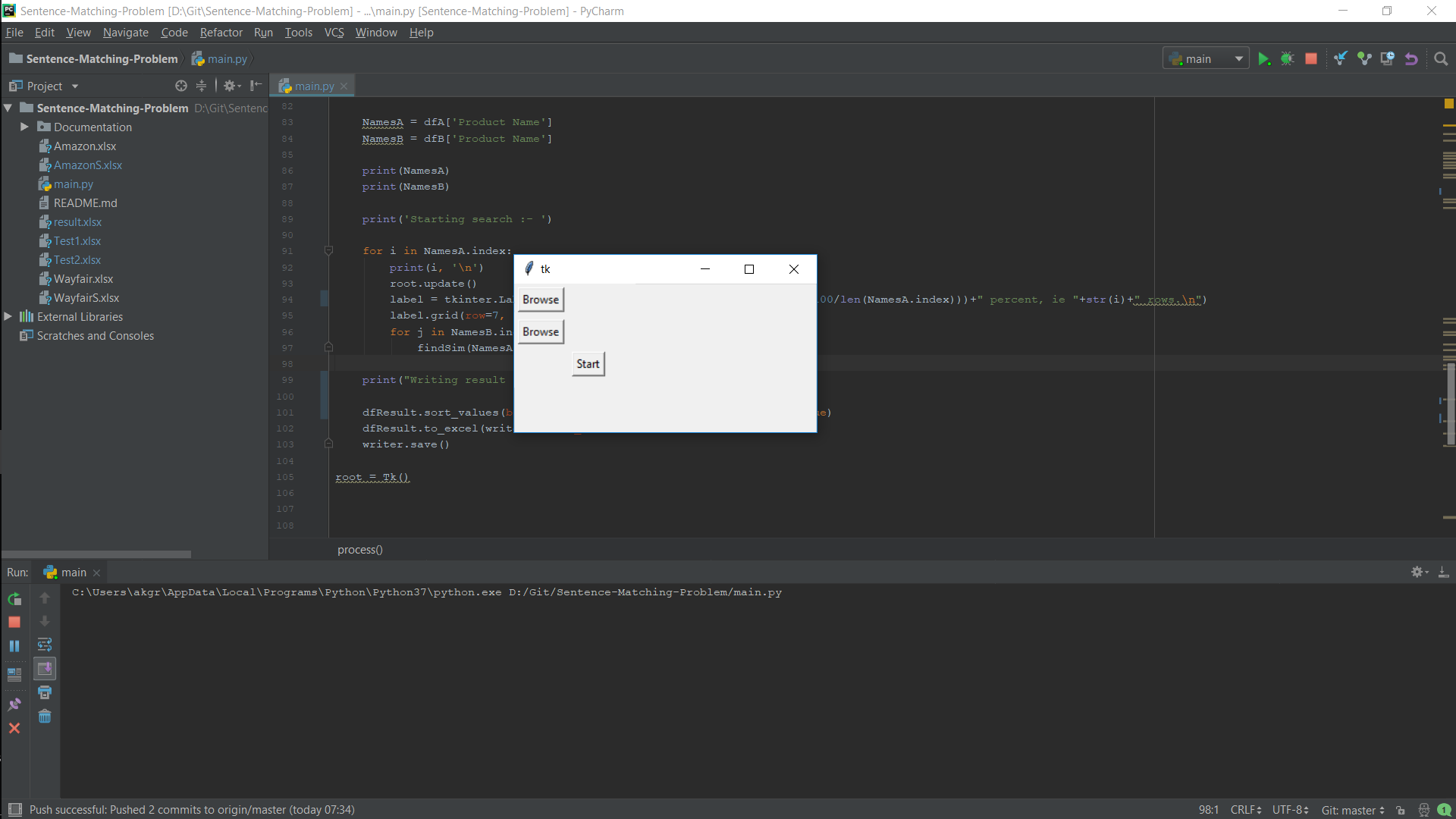
* It takes a lot of time (5-6 hours) to compare all the elements from the two original datasets. Because each of the original files has well over 4 lakh entries each, there is no simple way over the huge run-time.
* For example, let us take two products to be “Blue headphone with mic” and “Over-the-ear headphone without mic”. According to this algorithm, these will have high Jaccard index even though ‘with mic’ and ‘without mic’ are the complete opposite.

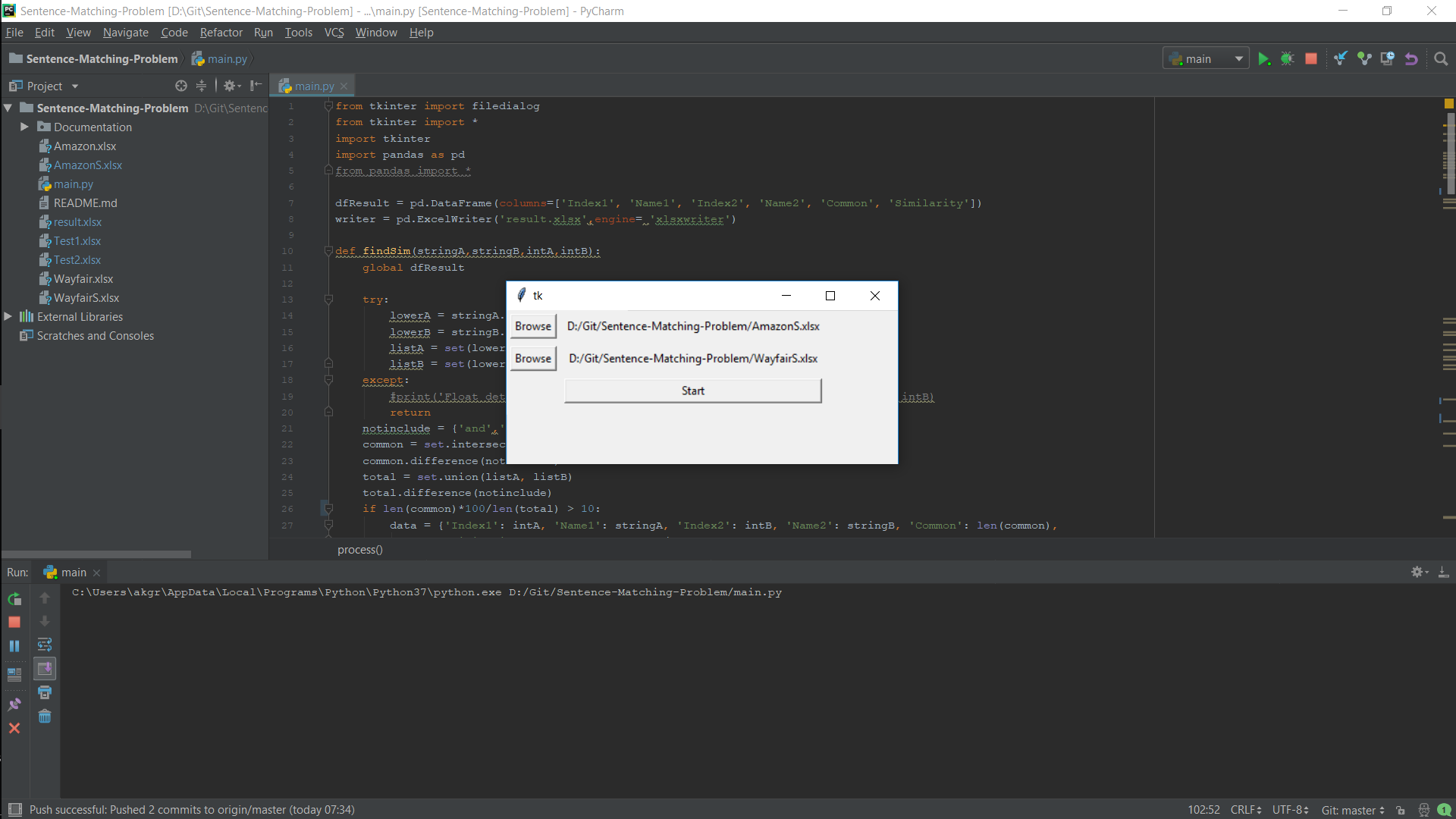
We can check for such examples and try to include a catch mechanism for the same in subsequent iterations.

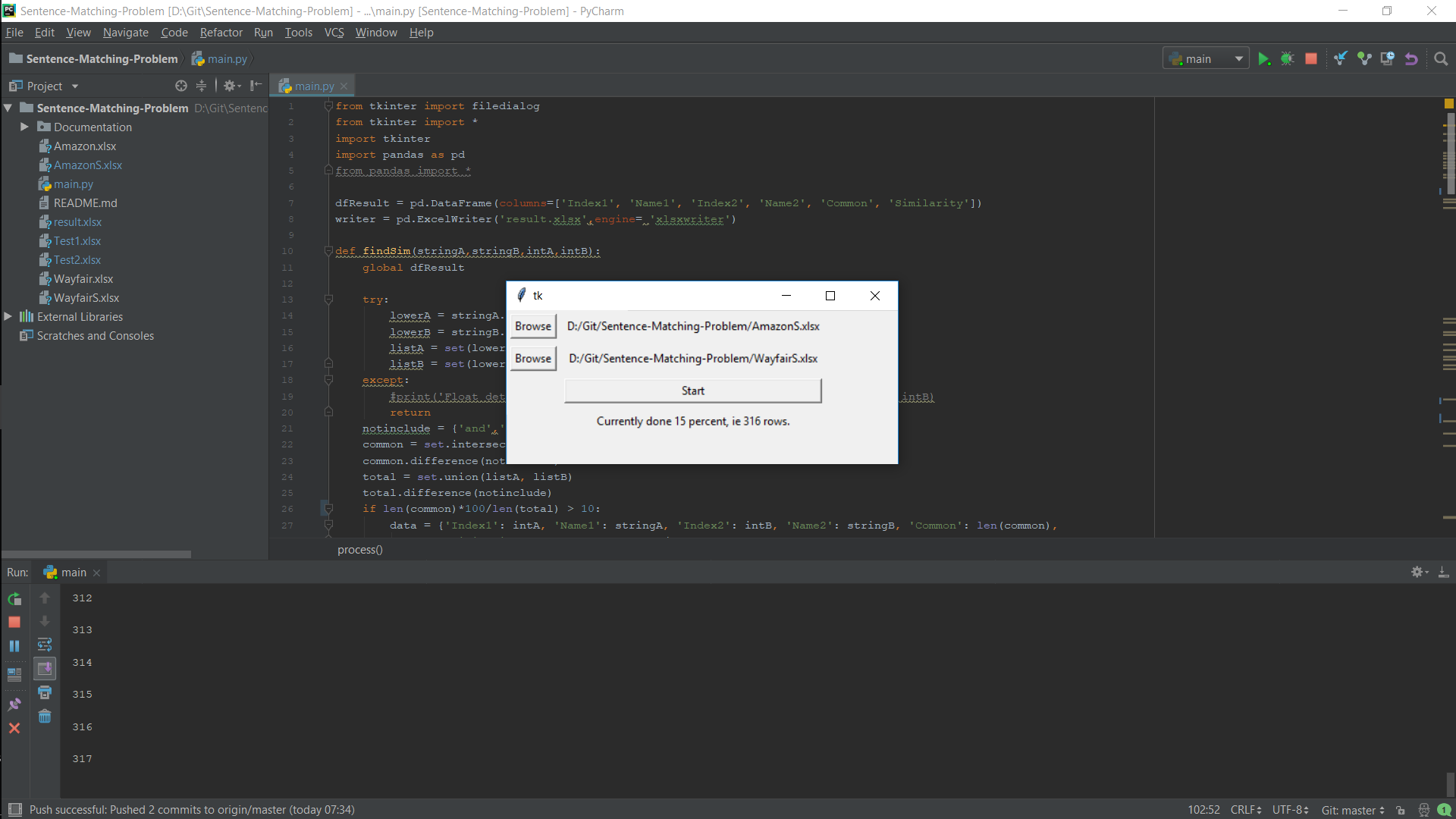
* This only works as long as similar products have similar words in the product description. If the two items uses synonyms or definitions of the same word, this algorithm will not be able to find those.

We can also try to include some dictionary type of function to find synonyms but due to run-time constraint not sure if this is viable.

**Some images of the basic GUI**

****

****

****

**What next**

* Develop better user interface (GUI)
* Try to find simpler algorithm that provides an acceptable result.
* Making it more versatile so that it works for slightly different scenarios too.