3. There are products in these two files which have different name and description in these two files but referring to the same entity.

a. Present your entity resolution algorithm with appropriate pseudo-code and description.

b. Present your source code.

c. Present a csv file as the output of your algorithm which contains one to one mapping between the ids of the products (one from amazon and the other from google), as they are identified as the same entity by your algorithm.

Entity Resolution refers to the task of finding records in a dataset that refer to same entity across different data sources ( In this case two csv files GoogleProducts.csv and Amazon.csv.)

In this particular question we have to perform algorithm on “Description” entity for both the files as that is present in both “GoogleProducts.csv” and “Amazon.csv” files.

1. Tokenize a String in NLTK

Implement simpleTokenize( string) which would split the string . Here the function takes the string and returns a list of non-empty tokens in the string.

1. Removing stopwords

The common English words which do not contribute to any meaning or purpose (e.g. “is”, “umm”, “the”, ”to” etc.). These words create noise in the comparison, hence they are removed.

1. Term-Frequency/Inverse-Document-Frequency (TF-IDF)

Comparison between the “Description” of both files cannot be done in optimized way when all tokens are treated in similar manner. Some tokens are more important than the others. Weight gives idea of which tokens are to be favored.

TF - a word which occurs often in a document, then it is more important to the meaning of the document.

IDF – are those words which are rarely shared among both the documents.

3.a. Implement TF function

Python dictionary is created . For each tokens in the input tokens list, count is incremented by 1 for each occurrence and is added to the dictionary. Then for each tokens in the dictionary, the token’s count is divided by the total number of tokens in input tokens list.

3.b. Implement IDF function

IDF assigns weight to every unique token. The function returns the key which is the unique token and the IDF weight of the token.

1. Cosine Similarity

The metric string distance is computed using Cosine Similarity. The two files are treated as vector in a high dimensional space and then the cosine similarity in implemented on both the vectors to find the similarity between them.