Junior Data Scientist Take-Home Task: Product Catalogue Creation

Objective:

The objective of the take home task is to create a master product catalogue by merging together product-related information from different data sources. The main challenge is to identify and deduplicate identical products that have different naming conventions across the source datasets.

As an illustrative example, one of the provided datasets has a product called Amazon Simple Email Service while other Amazon Simple Email Service (Amazon SES).

Such a product catalogue serves as the foundation of a knowledge graph, which is critical for powering a recommender system that suggests relevant products to users based on their preferences and past behavior. Building this accurate and comprehensive catalogue is key to a functional recommender system.

Your task is to build a data pipeline that takes as an input n number of data sources and outputs a master catalogue of deduplicated products with information provided from the source datasets. All the necessary data will be provided. You are expected to present your process and the findings in a final interview stage to members of Dragonfly team.

Task Details

1. Data Ingestion: Load both CSV datasets (ts_technologies.csv and bd_technologies.csv).
2. Data Exploration and Cleaning: Examine the datasets for missing values, inconsistencies, and other data quality issues. Apply necessary cleaning steps.
3. Product Deduplication:

- Develop a strategy to identify products that are the same but have different names or descriptions across the two datasets. This may involve fuzzy matching, text similarity measures, or other techniques.
- Implement the deduplication logic to create a list of unique products.
- Design your implementation so that it can be reused by others on more data.
- Master Catalogue Creation: Generate a catalogue that contains the consolidated and deduplicated product information.

1. Data Ingestation

In [1]:
```
import pandas as pd
import numpy as np
import re
```

In [2]:
```
# load the CSV files
bd_tech = pd.read_csv('data/bd_technologies.csv')
ts_tech = pd.read_csv('data/ts_technologies.csv')

# reset the index of both DataFrames
bd_tech.reset_index(drop=True, inplace=True)
ts_tech.reset_index(drop=True, inplace=True)
```

2.1. Data Exploration

In [3]:
```
# display the first few rows of each DataFrame
display(bd_tech.head()), display(ts_tech.head())
```

	product_name	description	seller_description	seller_website	m
0	Stellar Toolkit for iPhone	Stellar Toolkit for iPhone is a comprehensive ...	Established in 1993, Stellar® is a global lead...	https://www.stellarinfo.com/	
1	OpenText Voltage SmartCipher	Simplify unstructured data security with persi...	OpenText software applications manage content ...	https://www.opentext.com/	
2	IBM Maximo IT	Rapid expansion of information technology has ...	IBM offers a wide range of technology and cons...	https://www.ibm.com/	
3	Panvalet	CA Panvalet is a library management system tha...	Broadcom Inc. (NASDAQ: AVGO) is a global techn...	https://www.broadcom.com/	
4	OpenText ZENworks Patch Management	Micro Focus ZENworks Patch Management (formerl...	OpenText software applications manage content ...	https://www.opentext.com/	

	technology_id	name	slug	
0	8	ActiveCampaign	activecampaign	http://www.activeca
1	12	Acuity Scheduling	acuity-scheduling	https://acuitysch
2	17	Adobe Illustrator	adobe-illustrator	https://www.adobe.com/ru/products
3	18	Adobe Photoshop	adobe-photoshop	https://www
4	36	AfterShip	aftership	https://www.a

Out[3]: (None, None)

In [4]:
```
# display the column names of each DataFrame
display(bd_tech.columns, ts_tech.columns)
```

```
Index(['product_name', 'description', 'seller_description', 'seller_websit
e',
       'main_category', 'software_product_id', 'overview', 'headquarters',
       'categories'],
      dtype='object')
Index(['technology_id', 'name', 'slug', 'url', 'description', 'category',
       'category_slug', 'parent_category', 'parent_category_slug', 'jobs',
       'companies', 'companies_found_last_week'],
      dtype='object')
```

In [5]:
```
# summary statistics for each DataFrame
display(bd_tech.describe(), ts_tech.describe())
```

	product_name	description	seller_description	seller_website
count	75975	75975	75975	75969
unique	75173	75539	57046	56989
top	Atlas	It takes an image as input and classifies the ...	By giving customers more of what they want - l...	https://aws.amazon.com/?nc2=h_lg
freq	8	12	324	324

	technology_id	jobs	companies	companies_found_last_week
count	32197.000000	3.219700e+04	32197.000000	32197.000000
mean	100681.749231	3.872473e+03	790.560332	5.626798
std	43498.661615	5.307573e+04	8162.288487	59.607930
min	8.000000	0.000000e+00	0.000000	0.000000
25%	109065.000000	0.000000e+00	0.000000	0.000000
50%	120143.000000	8.000000e+00	4.000000	0.000000
75%	128277.000000	1.210000e+02	50.000000	0.000000
max	136401.000000	3.822115e+06	605860.000000	4433.000000

In [6]:
```
# display the general information about each DataFrame
bd_tech.info(verbose=True), ts_tech.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 75975 entries, 0 to 75974
Data columns (total 9 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   product_name         75975 non-null  object
 1   description          75975 non-null  object
 2   seller_description   75975 non-null  object
 3   seller_website       75969 non-null  object
 4   main_category        75975 non-null  object
 5   software_product_id  75975 non-null  object
 6   overview             75975 non-null  object
 7   headquarters         42707 non-null  object
 8   categories           75972 non-null  object
dtypes: object(9)
memory usage: 5.2+ MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32197 entries, 0 to 32196
Data columns (total 12 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   technology_id              32197 non-null  int64
 1   name                       32197 non-null  object
 2   slug                       32197 non-null  object
 3   url                        7466 non-null   object
 4   description                8393 non-null   object
 5   category                   32197 non-null  object
 6   category_slug              32197 non-null  object
 7   parent_category            32197 non-null  object
 8   parent_category_slug       32197 non-null  object
 9   jobs                       32197 non-null  int64
 10  companies                  32197 non-null  int64
 11  companies_found_last_week  32197 non-null  int64
dtypes: int64(4), object(8)
memory usage: 2.9+ MB
```

Out[6]: (None, None)

In [7]:
```
# Function to generate data quality report
def data_quality_report(df):
    return pd.DataFrame({
        "Missing Values": df.isnull().sum(),
        "Percentage Missing": (df.isnull().sum() / len(df)) * 100,
        "Data Type": df.dtypes
    })
# Generate data quality reports
bd_quality_report = data_quality_report(bd_tech)
ts_quality_report = data_quality_report(ts_tech)

display(bd_quality_report), display(ts_quality_report)
```

	Missing Values	Percentage Missing	Data Type
product_name	0	0.000000	object
description	0	0.000000	object
seller_description	0	0.000000	object
seller_website	6	0.007897	object
main_category	0	0.000000	object
software_product_id	0	0.000000	object
overview	0	0.000000	object
headquarters	33268	43.788088	object
categories	3	0.003949	object

	Missing Values	Percentage Missing	Data Type
technology_id	0	0.000000	int64
name	0	0.000000	object
slug	0	0.000000	object
url	24731	76.811504	object
description	23804	73.932354	object
category	0	0.000000	object
category_slug	0	0.000000	object
parent_category	0	0.000000	object
parent_category_slug	0	0.000000	object
jobs	0	0.000000	int64
companies	0	0.000000	int64
companies_found_last_week	0	0.000000	int64

```
Out[7]: (None, None)
```

Initial Data Quality Issues:

> bd_technologies.csv

- Major missing values in the headquarters column (~43.8% missing).
- Minimal missing values in seller_website and categories.
- Also its important to note that these values are not accurate until the inconsistencies with data types and missing values are addressed.

> ts_technologies.csv

- Significant missing values in url (~76.8%) and description (~73.9%).

In [8]:
```
# display duplicates in each DataFrame
def display_duplicates(df):
    duplicates = df[df.duplicated()]
    if not duplicates.empty:
        print(f"Duplicates found:\n{duplicates}")
    else:
        print("No duplicates found.")
display_duplicates(bd_tech)
display_duplicates(ts_tech)
```

```
No duplicates found.
No duplicates found.
```

2.2. Data Cleaning

2.2.1. Missing Values

In [9]:
```
# replace field that's entirely space (or empty) with NaN
ts_tech = ts_tech.replace(r'^\s*$', np.nan, regex=True)
bd_tech = bd_tech.replace(r'^\s*$', np.nan, regex=True)
```

2.2.2. Data Types

In [10]:
```
# convert all column data types to string for consistency in db_tech
def convert_to_string(df):
    for col in df.columns:
        df[col] = df[col].astype('string')
    return df
bd_tech = convert_to_string(bd_tech)
```

In [11]:
```
bd_tech.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 75975 entries, 0 to 75974
Data columns (total 9 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   product_name         75975 non-null  string
 1   description          75975 non-null  string
 2   seller_description   75975 non-null  string
 3   seller_website       75969 non-null  string
 4   main_category        75975 non-null  string
 5   software_product_id  75975 non-null  string
 6   overview             75975 non-null  string
 7   headquarters         42707 non-null  string
 8   categories           75972 non-null  string
dtypes: string(9)
memory usage: 5.2 MB
```

In [12]:
```
# convert the columns 1 to 9 in ts_tech to string
def convert_ts_tech_to_string(df):
    for col in df.columns[1:9]:
        df[col] = df[col].astype('string')
    return df
ts_tech = convert_ts_tech_to_string(ts_tech)
ts_tech.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32197 entries, 0 to 32196
Data columns (total 12 columns):
 #   Column                     Non-Null Count  Dtype 
---  ------                     --------------  ----- 
 0   technology_id              32197 non-null  int64 
 1   name                       32197 non-null  string
 2   slug                       32197 non-null  string
 3   url                        7466 non-null   string
 4   description                8393 non-null   string
 5   category                   32197 non-null  string
 6   category_slug              32197 non-null  string
 7   parent_category            32197 non-null  string
 8   parent_category_slug       32197 non-null  string
 9   jobs                       32197 non-null  int64 
 10  companies                  32197 non-null  int64 
 11  companies_found_last_week  32197 non-null  int64 
dtypes: int64(4), string(8)
memory usage: 2.9 MB
```

2.2.3. Standardize the URL

While the primary identifiers are obviously product name and description, we can also consider URLs and Seller Websites which are often unique and can strongly confirm product matches. Therefore it is essential we have standardized URLs (removal of protocols, www and trailing slashes).

In [13]:
```
# standardize the url and seller_website columns by removing protocols, wwww
def standardize_urls(df, url_col):
    df[url_col] = df[url_col].str.lower()  # convert to lowercase
    df[url_col] = df[url_col].str.replace(r'^https?://', '', regex=True)  # 
    df[url_col] = df[url_col].str.replace(r'^www\.', '', regex=True)  # remo
    df[url_col] = df[url_col].str.rstrip('/')  # remove trailing slashes
    return df
bd_tech = standardize_urls(bd_tech, 'seller_website')
ts_tech = standardize_urls(ts_tech, 'url')
```

In [14]:
```
# pick 20 random entries from the seller_website column for quality control
bd_tech['seller_website'].sample(5)
```

Out[14]:
```
34425     graydrop.com
3432        fortra.com
33893     giighire.com
8733     woochitra.com
29871      logicare.us
Name: seller_website, dtype: string
```

In [15]:
```
ts_tech['url'].sample(5)
```

Out[15]:
```
23849              <NA>
2337     livereload.com
12434              <NA>
23753              <NA>
17260              <NA>
Name: url, dtype: string
```

2.2.4. Clean the categories column

In [16]:
```
# clean the categories column in bd_tech - removing brackets and quotes
bd_tech['categories'] = (
    bd_tech['categories']
    .str.replace(r'[\[\]"]', '', regex=True)           # Remove brackets and
    .str.replace(r',\s*', ', ', regex=True)            # Ensure single space
)
```

In [17]:
```
bd_tech.categories.sample(5)
```

Out[17]:
```
25745                        Email Management
8592                             Other Marine
16172       Applicant Tracking Systems (ATS)
68394              Education Finance Software
32229    Risk-Based Vulnerability Management
Name: categories, dtype: string
```

`software_product_id` column in `bd_tech` and `parent_category_slug` column in `ts_tech` have string values seperated by hyphens replace them with space instead.

In [18]:
```
def remove_hyphens(column):
    # Replace hyphens with spaces in the specified column
    return column.str.replace(r'(?<=[A-Za-z])-(?=[A-Za-z])', ' ', regex=True

# Apply the function to the 'software_product_id' column in bd_tech
bd_tech['software_product_id'] = remove_hyphens(bd_tech['software_product_ic

# Apply the function to the 'slug', 'category_slug' and 'parent_category_slu
ts_tech['slug'] = remove_hyphens(ts_tech['slug'])
ts_tech['category_slug'] = remove_hyphens(ts_tech['category_slug'])
ts_tech['parent_category_slug'] = remove_hyphens(ts_tech['parent_category_sl
```

In [19]:
```
# Sample the cleaned 'software_product_id' column in bd_tech
bd_tech['software_product_id'].sample(5)
```

Out[19]:
```
17489         web payment software
52843                    instasent
72627                   amrita his
62840    tunefab audible converter
12819                      agrible
Name: software_product_id, dtype: string
```

In [20]:
```
# randomly sample 5 entries from the 'slug', 'category_slug', 'parent_catego
ts_tech[['slug', 'category_slug', 'parent_category_slug']].sample(5)
```

Out[20]:

	slug	category_slug	parent_category_slug
28886	apache mynewt	server and desktop os	platform and storage
27148	blackberry enterprise consulting	professional services automation	finance and accounting
14210	akamai cdn	content delivery network cdn	devops and development
1220	chatwerk	customer satisfaction	customer management
9760	sap predictive analytics	data mining	business intelligence and analytics

In [21]:
```
# combine 'category' and 'parent_category' in ts_tech into a single column c
ts_tech_clean = ts_tech.copy()
ts_tech_clean['categories_ts'] = ts_tech_clean['category'] + ', ' + ts_tech_
# ts_tech_clean.drop(columns=['category', 'parent_category', 'parent_categor
# remove leading and trailing spaces from the 'categories' column in ts_tech
ts_tech_clean['categories_ts'] = ts_tech_clean['categories_ts'].str.strip()
# make everything in the 'categories' column lowercase and remove commas
ts_tech_clean['categories_ts'] = ts_tech_clean['categories_ts'].str.lower()
ts_tech_clean['categories_ts'] = ts_tech_clean['categories_ts'].str.replace(

# combine main_category and categories in bd_tech into a single column calle
bd_tech_clean = bd_tech.copy()
bd_tech_clean['categories_bd'] = bd_tech_clean['main_category'] + ', ' + bd_
# bd_tech_clean.drop(columns=['main_category', 'categories', 'software_produ
# remove leading and trailing spaces from the 'categories' column in bd_tech
bd_tech_clean['categories_bd'] = bd_tech_clean['categories_bd'].str.strip()
# make everything in the 'categories' column lowercase removing commas
bd_tech_clean['categories_bd'] = bd_tech_clean['categories_bd'].str.lower()
bd_tech_clean['categories_bd'] = bd_tech_clean['categories_bd'].str.replace(
# display the first few rows of each DataFrame after cleaning
display(bd_tech_clean.head()), display(ts_tech_clean.head())
```

	product_name	description	seller_description	seller_website	main_catego
0	Stellar Toolkit for iPhone	Stellar Toolkit for iPhone is a comprehensive ...	Established in 1993, Stellar® is a global lead...	stellarinfo.com	Data Recove Softwa
1	OpenText Voltage SmartCipher	Simplify unstructured data security with persi...	OpenText software applications manage content ...	opentext.com	Confidential Softwa
2	IBM Maximo IT	Rapid expansion of information technology has ...	IBM offers a wide range of technology and cons...	ibm.com	Service De Softwa
3	Panvalet	CA Panvalet is a library management system tha...	Broadcom Inc. (NASDAQ: AVGO) is a global techn...	broadcom.com	DevO Softwa
4	OpenText ZENworks Patch Management	Micro Focus ZENworks Patch Management (formerl...	OpenText software applications manage content ...	opentext.com	Vulnerabil Manageme Softwa

	technology_id	name	slug	
0	8	ActiveCampaign	activecampaign	activecampaign.c
1	12	Acuity Scheduling	acuity scheduling	acuityscheduling.c
2	17	Adobe Illustrator	adobe illustrator	adobe.com/ru/products/illustrator.ht
3	18	Adobe Photoshop	adobe photoshop	adobe.c
4	36	AfterShip	aftership	aftership.c

```
Out[21]: (None, None)
```

In [22]:
```
ts_tech_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32197 entries, 0 to 32196
Data columns (total 13 columns):
 #   Column                      Non-Null Count  Dtype 
---  ------                      --------------  ----- 
 0   technology_id               32197 non-null  int64 
 1   name                        32197 non-null  string
 2   slug                        32197 non-null  string
 3   url                         7466 non-null   string
 4   description                 8393 non-null   string
 5   category                    32197 non-null  string
 6   category_slug               32197 non-null  string
 7   parent_category             32197 non-null  string
 8   parent_category_slug        32197 non-null  string
 9   jobs                        32197 non-null  int64 
 10  companies                   32197 non-null  int64 
 11  companies_found_last_week   32197 non-null  int64 
 12  categories_ts               32197 non-null  string
dtypes: int64(4), string(9)
memory usage: 3.2 MB
```

In [23]:
```
bd_tech_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 75975 entries, 0 to 75974
Data columns (total 10 columns):
 #   Column                Non-Null Count  Dtype 
---  ------                --------------  ----- 
 0   product_name          75975 non-null  string
 1   description           75975 non-null  string
 2   seller_description    75975 non-null  string
 3   seller_website        75969 non-null  string
 4   main_category         75975 non-null  string
 5   software_product_id   75975 non-null  string
 6   overview              75975 non-null  string
 7   headquarters          42707 non-null  string
 8   categories            75972 non-null  string
 9   categories_bd         75972 non-null  string
dtypes: string(10)
memory usage: 5.8 MB
```

3. Product Deduplicaton

3.1. Data Engineering

Here we will standardize our text. This ensures that data across records follows the same format, which is crucial for reliable comparison. urls have already been standardized. Next we clean text by converting to lowercase, removing punctutations, extra whitespaces and stopwords, and lastly applying lemmatization.

In [24]:
```
# Import necessary libraries
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
import nltk

# Download necessary NLTK data
nltk.download('stopwords')
nltk.download('wordnet')

bd_tech_processed = bd_tech_clean.copy()
ts_tech_processed = ts_tech_clean.copy()
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/elnaramammadova/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     /Users/elnaramammadova/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

In [25]:
```
# Define cleaning function
lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))

def clean_text(text):
    if pd.isna(text):
        return ""
    text = text.lower()
    text = re.sub(r'[^\w\s]', ' ', text)
    words = [lemmatizer.lemmatize(word) for word in text.split() if word not
    return ' '.join(words)

# Define preprocessing function for combined columns
def preprocess(df, columns):
    return df[columns].fillna('').agg(' '.join, axis=1).apply(clean_text)

# Apply preprocessing to datasets
bd_tech_processed['combined'] = preprocess(bd_tech_processed, ['product_name
ts_tech_processed['combined'] = preprocess(ts_tech_processed, ['name', 'desc

# Display the first few results
bd_tech_processed[['product_name', 'combined']].head(), ts_tech_processed[['
```

```
Out[25]: (                          product_name  \
 0          Stellar Toolkit for iPhone
 1       OpenText Voltage SmartCipher
 2                       IBM Maximo IT
 3                            Panvalet
 4  OpenText ZENworks Patch Management

                                             combined
 0  stellar toolkit iphone stellar toolkit iphone ...
 1  opentext voltage smartcipher simplify unstruct...
 2  ibm maximo rapid expansion information technol...
 3  panvalet ca panvalet library management system...
 4  opentext zenworks patch management micro focus...  ,
                  name                                           combined
 0     ActiveCampaign  activecampaign recognized leader marketing sal...
 1  Acuity Scheduling  acuity scheduling easy use user friendly sched...
 2  Adobe Illustrator  adobe illustrator industry standard vector gra...
 3    Adobe Photoshop  adobe photoshop best world graphic design imag...
 4          AfterShip  aftership aftership provides shipment tracking...)
```

```
In [26]: # display the first few rows of the processed DataFrames
display(bd_tech_processed.head()), display(ts_tech_processed.head())
```

	product_name	description	seller_description	seller_website	main_catego
0	Stellar Toolkit for iPhone	Stellar Toolkit for iPhone is a comprehensive ...	Established in 1993, Stellar® is a global lead...	stellarinfo.com	Data Recove Softwa
1	OpenText Voltage SmartCipher	Simplify unstructured data security with persi...	OpenText software applications manage content ...	opentext.com	Confidential Softwa
2	IBM Maximo IT	Rapid expansion of information technology has ...	IBM offers a wide range of technology and cons...	ibm.com	Service De Softwa
3	Panvalet	CA Panvalet is a library management system tha...	Broadcom Inc. (NASDAQ: AVGO) is a global techn...	broadcom.com	DevO Softwa
4	OpenText ZENworks Patch Management	Micro Focus ZENworks Patch Management (formerl...	OpenText software applications manage content ...	opentext.com	Vulnerabil Manageme Softwa

	technology_id	name	slug	
0	8	ActiveCampaign	activecampaign	activecampaign.c
1	12	Acuity Scheduling	acuity scheduling	acuityscheduling.c
2	17	Adobe Illustrator	adobe illustrator	adobe.com/ru/products/illustrator.h
3	18	Adobe Photoshop	adobe photoshop	adobe.c
4	36	AfterShip	aftership	aftership.c

Out[26]: (None, None)

In [27]:
```
# save the cleaned DataFrames to CSV files
bd_tech_processed.to_csv('data/bd_technologies_cleaned.csv', index=False)
ts_tech_processed.to_csv('data/ts_technologies_cleaned.csv', index=False)
```

3.2. Fuzzy Matching (Rapid Initial Matching)

Fuzzy Matching is a quick tool for calculating similarities between strings to help identify values that are "close enough". This method allows variations and/or inconsistencies in data (i.e., typos, different spelling) to be considered similar. We calculate the Levenshtein distance between our strings with a threshold of 0.85. If the similarity score exceeds the threshold, the data records are considered a fuzzy match and can be linked.

In [ ]:
```
```

In [28]:
```
# load the cleaned DataFrames
bd_tech_processed = pd.read_csv('data/bd_technologies_cleaned.csv')
ts_tech_processed = pd.read_csv('data/ts_technologies_cleaned.csv')
```

In [29]:
```
# from fuzzywuzzy import fuzz, process
```

```
# def fuzzy_match(ts_text, bd_choices, threshold=85):
#     match, score = process.extractOne(ts_text, bd_choices)
#     return (match, score) if score >= threshold else (None, score)

# ts_tech_processed['fuzzy_match'] = ts_tech_processed['combined'].apply(lam

# fuzzywuzzy was taking too long to run, so we will switch to thefuzz instea

# from thefuzz import fuzz, process
# from tqdm import tqdm

# # enable tqdm for pandas
# tqdm.pandas()

# def fuzzy_match_fast(ts_text, bd_choices, threshold=85):
#     match, score = process.extractOne(ts_text, bd_choices, scorer=fuzz.tok
#     return (match, score) if score >= threshold else (None, score)

# # Apply to TS dataset with progress bar
# ts_tech_processed['fuzzy_match'] = ts_tech_processed['combined'].progress_
#     lambda x: fuzzy_match_fast(x, bd_tech_processed['combined'].tolist())
# )
# The above code was still taking too long, so we will try using RapidFuzz f

# Use RapidFuzz for faster fuzzy matching
# from tqdm import tqdm
# from rapidfuzz import fuzz, process

# # enable tqdm for pandas
# tqdm.pandas()

# def fuzzy_match_fast(ts_text, bd_choices, threshold=85):
#     match = process.extractOne(ts_text, bd_choices, scorer=fuzz.token_set_
#     return match if match else (None, 0)

# # Apply to TS dataset with progress bar
# ts_tech_processed['fuzzy_match'] = ts_tech_processed['combined'].progress_
#     lambda x: fuzzy_match_fast(x, bd_tech_processed['combined'].tolist())
# )

# Even this would take 4 hrs
```

Both Fuzzy Matching methods took longer than 8 hrs to complete. We will try to speed up the fuzzy matching process by applying below methods:

- Limit to top 10 cadidates with TF-IDF Pre-Filtering by cosine similarity
- Apply fuzzy matching only on those
- use RapidFuzz instead of thefuzz and fuzzywuzzy

This method will provide a semantic matching beyond simple string similarity.

TF-IDF (Term Frequency-Inverse Document Frequency) and cosine similarity are used together to measure the similarity between text documents. TF-IDF converts text into numerical vectors, while cosine similarity calculates the similarity between those vectors. A higher cosine similarity value indicates greater similarity between documents.

1. TF-IDF: Term Frequency (TF): Measures how often a term appears in a document. Inverse Document Frequency (IDF): Measures how rare a term is across a corpus of documents. TF-IDF Value: The product of TF and IDF, weighting terms based on their importance in a specific document and the entire corpus. Vector Representation: TF-IDF converts each document into a numerical vector, where each dimension represents a term and the value in that dimension is the term's TF-IDF score.
2. Cosine Similarity: Concept: Measures the similarity between two vectors by calculating the cosine of the angle between them. Calculation:Takes the dot product of the two vectors and divides it by the product of their magnitudes. Interpretation: A cosine similarity value of 1 indicates identical vectors (perfectly similar), 0 indicates orthogonal vectors (no similarity), and -1 indicates completely opposite vectors (perfectly dissimilar). Usage: Used to compare the numerical vectors created by TF-IDF, determining how closely two documents match based on their term frequencies and importance. In summary: TF-IDF creates numerical representations of documents, and cosine similarity compares these representations to determine how similar the documents are based on their content.

In [30]:

```
# TF-IDF Vectorization
from sklearn.feature_extraction.text import TfidfVectorizer
def compute_tfidf(df, column):
    vectorizer = TfidfVectorizer(max_features=5000, stop_words='english')
    tfidf_matrix = vectorizer.fit_transform(df[column])
    return tfidf_matrix, vectorizer

# Compute TF-IDF for both datasets
bd_tfidf, bd_vectorizer = compute_tfidf(bd_tech_processed, 'combined')
ts_tfidf, ts_vectorizer = compute_tfidf(ts_tech_processed, 'combined')

# Cosine Similarity
from sklearn.metrics.pairwise import cosine_similarity
cos_sim_matrix = cosine_similarity(ts_tfidf, bd_tfidf)
```

In [31]:

```
from tqdm import tqdm
from rapidfuzz import fuzz, process

# Limit to top 5 candidates and apply fuzzy matching
tqdm.pandas()
fuzzy_matches = []

for i in tqdm(range(len(ts_tech_processed)), desc="Matching top candidates")
    ts_text = ts_tech_processed.iloc[i]['combined']
    top_indices = cos_sim_matrix[i].argsort()[-5:][::-1]
    candidates = bd_tech_processed.iloc[top_indices]['combined'].tolist()
    result = process.extractOne(ts_text, candidates, scorer=fuzz.token_set_r
```

```
    fuzzy_matches.append(result)

# Store results
ts_tech_processed['fuzzy_match'] = fuzzy_matches

# Display sample results
ts_tech_processed[['name', 'fuzzy_match']].head()
```

```
Matching top candidates: 100%|██████████| 32197/32197 [00:43<00:00, 738.64i
t/s]
```

Out[31]:

	name	fuzzy_match
0	ActiveCampaign	(payquiq payquiq service include credit card a...
1	Acuity Scheduling	(roi training roi delivers customized technolo...
2	Adobe Illustrator	(ispot tv ispot tv real time tv ad data analyt...
3	Adobe Photoshop	(qr crazy qr code generator qr crazy qr code g...
4	AfterShip	(cyberwar cyberwar great addition every online...

In [32]:

```
# Extract matched product names from fuzzy match results
# Assuming each entry in 'fuzzy_match' is a tuple: (matched_text, score)
matched_names = ts_tech_processed['fuzzy_match'].dropna().apply(lambda x: x[
match_scores = ts_tech_processed['fuzzy_match'].dropna().apply(lambda x: x[1

# Create a DataFrame for matched pairs
matched_df = ts_tech_clean.loc[matched_names.index, ['name', 'description',
matched_df['matched_to'] = matched_names.values
matched_df['match_score'] = match_scores.values

# Merge with bd_tech on 'product_name' (which matched_to points to)
bd_matched = bd_tech_clean[['product_name', 'description', 'seller_website',
bd_matched.columns = ['matched_to', 'bd_description', 'bd_url', 'bd_category

# Join TS matches with BD products
merged = pd.merge(matched_df, bd_matched, on='matched_to', how='left')
# rename categories_ts to category
merged.rename(columns={'categories_ts': 'category'}, inplace=True)
display(merged.head())
```

	name	description	url	category
0	ActiveCampaign	Recognized as the leader in the marketing and ...	activecampaign.com	marketing automation platforms marketing
1	Acuity Scheduling	It is easy-to-use and user friendly scheduling...	acuityscheduling.com	appointments and scheduling customer management
2	Adobe Illustrator	The industry-standard vector graphics app lets...	adobe.com/ru/products/illustrator.html	graphic design software product and design
3	Adobe Photoshop	It is the best in the world of graphic design ...	adobe.com	graphic design software product and design
4	AfterShip	AfterShip provides shipment tracking API for o...	aftership.com	shipping and fulfillment e-commerce

In [33]:

```
# Consolidate matched fields (favor longer description, combine URLs)
merged['product_name'] = merged['matched_to']
merged['description'] = merged.apply(lambda row: row['description'] if len(s
merged['url'] = merged.apply(lambda row: row['url'] if pd.notna(row['url'])
merged['category'] = merged.apply(lambda row: row['category'] if pd.notna(rc
merged['source'] = 'matched'

# Select relevant columns
master_matched = merged[['product_name', 'description', 'url', 'category', '

# Unmatched TS entries
matched_ts_names = set(matched_df['name'])
ts_unmatched = ts_tech[~ts_tech['name'].isin(matched_ts_names)]
ts_only = ts_unmatched[['name', 'description', 'url', 'category']].copy()
ts_only.columns = ['product_name', 'description', 'url', 'category']
ts_only['source'] = 'ts_only'
ts_only['match_score'] = None

# Unmatched BD entries
matched_bd_names = set(matched_df['matched_to'])
bd_unmatched = bd_tech[~bd_tech['product_name'].isin(matched_bd_names)]
bd_only = bd_unmatched[['product_name', 'description', 'seller_website', 'ma
bd_only.columns = ['product_name', 'description', 'url', 'category']
bd_only['source'] = 'bd_only'
```

```
bd_only['match_score'] = None

# Combine all parts into master catalogue
master_catalogue = pd.concat([master_matched, ts_only, bd_only], ignore_inde

display(master_catalogue.head())
```

```
/var/folders/vd/k74_1qd12wv9pzhrjx9lkbzr0000gn/T/ipykernel_17010/4288397362.
py:28: FutureWarning: The behavior of DataFrame concatenation with empty or
all-NA entries is deprecated. In a future version, this will no longer exclu
de empty or all-NA columns when determining the result dtypes. To retain the
old behavior, exclude the relevant entries before the concat operation.
  master_catalogue = pd.concat([master_matched, ts_only, bd_only], ignore_in
dex=True)
```

	product_name	description	url	category
0	payquiq payquiq service include credit card ac...	Recognized as the leader in the marketing and ...	activecampaign.com	marketing automation platforms marketing
1	roi training roi delivers customized technolog...	It is easy-to-use and user friendly scheduling...	acuityscheduling.com	appointments and scheduling customer management
2	ispot tv ispot tv real time tv ad data analyti...	The industry-standard vector graphics app lets...	adobe.com/ru/products/illustrator.html	graphic design software product and design
3	qr crazy qr code generator qr crazy qr code ge...	It is the best in the world of graphic design ...	adobe.com	graphic design software product and design
4	cyberwar cyberwar great addition every online ...	AfterShip provides shipment tracking API for o...	aftership.com	shipping and fulfillment e-commerce

In [34]:
```
# Display the info on source=matched data only
matched_stats = master_catalogue[master_catalogue['source'] == 'matched'].ir
display(matched_stats)
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 32197 entries, 0 to 32196
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   product_name  32197 non-null  object 
 1   description   8392 non-null   object 
 2   url           7464 non-null   object 
 3   category      32197 non-null  object 
 4   source        32197 non-null  object 
 5   match_score   32197 non-null  float64
dtypes: float64(1), object(5)
memory usage: 1.7+ MB
None
```

In [35]:
```
# Save the master catalogue to a CSV file
master_catalogue.to_csv('data/master_catalogue_fuzzy_matching.csv', index=Fa
```

3.3. Hyprid feature-based entity resolution with fuzzy token-set ratio

Fuzzy matching and entity resolution (ER) can be used together, and doing so often leads to stronger, more accurate results. Since they are not mutually exclusive — instead, we can design a hybrid system that uses both for complementary strengths.

Breakdown on how they differe.

Feature	Fuzzy Matching	Entity Resolution (ER)
Focus	String similarity (e.g., name matching)	Matching across multiple fields
Typical Tools	`fuzz`, `RapidFuzz`, `TheFuzz`	`recordlinkage`, `dedupe`, ML models
Strength	Flexible for spelling/name variations	Uses multiple features for stronger logic
Limitation	Works best on single fields	Slower unless blocked or trained well

This will mirror the fuzzy matching approach, but using recordlinkage to:

1. Identify matched product pairs (same real-world entity).
2. Merge their information.
3. Add unmatched products from each dataset.
4. Output a clean, deduplicated catalogue.

After loading cleaned and processed input features `name`, `description`, `url`, `category` we create candidate pairs while blocking by category to limit comparisons.

Compare fields using string similarity: `product_name ↔ name` `description ↔ description` `seller_website ↔ url`

Match if >=2 of 3 fields exceed similarity thresholds.

Finally construct master catalogue by merging matched pairs.

In [36]:
```
import pandas as pd
from recordlinkage import Index
from rapidfuzz import fuzz
from tqdm.auto import tqdm
```

In [37]:
```
# loaad the cleaned ts and bd tech DataFrames
bd = pd.read_csv('data/bd_technologies_cleaned.csv')
ts = pd.read_csv('data/ts_technologies_cleaned.csv')
```

In [38]:
```
bd.info(), ts.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 75975 entries, 0 to 75974
Data columns (total 11 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   product_name         75975 non-null  object
 1   description          75975 non-null  object
 2   seller_description   75975 non-null  object
 3   seller_website       75969 non-null  object
 4   main_category        75975 non-null  object
 5   software_product_id  75975 non-null  object
 6   overview             75975 non-null  object
 7   headquarters         42707 non-null  object
 8   categories           75972 non-null  object
 9   categories_bd        75972 non-null  object
 10  combined             75975 non-null  object
dtypes: object(11)
memory usage: 6.4+ MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32197 entries, 0 to 32196
Data columns (total 14 columns):
 #   Column                     Non-Null Count  Dtype 
---  ------                     --------------  ----- 
 0   technology_id              32197 non-null  int64 
 1   name                       32197 non-null  object
 2   slug                       32197 non-null  object
 3   url                        7464 non-null   object
 4   description                8393 non-null   object
 5   category                   32197 non-null  object
 6   category_slug              32197 non-null  object
 7   parent_category            32197 non-null  object
 8   parent_category_slug       32197 non-null  object
 9   jobs                       32197 non-null  int64 
 10  companies                  32197 non-null  int64 
 11  companies_found_last_week  32197 non-null  int64 
 12  categories_ts              32197 non-null  object
 13  combined                   32197 non-null  object
dtypes: int64(4), object(10)
memory usage: 3.4+ MB
```

Out[38]: (None, None)

`categories_bd` and `categories_ts` are not guaranteed to match exactly:

These columns are concatenations of multiple categories (often as a single string), not single, normalized category labels.

For blocking to work, the values in the left and right columns must match **exactly** (string equality) for a candidate pair to be generated. Since `categories_bd` and `categories_ts` are not normalized and may contain multiple categories in different formats, blocking on them will result in very few or no candidate pairs.

In [39]:
```python
def normalize_categories(col):
    # Split by comma, strip whitespace, lowercase, remove empty, sort, join
    return (
        col.fillna('')
        .apply(lambda x: ','.join(sorted(set(
            c.strip().lower() for c in re.split(r'[,\s]+', x) if c.strip()
        ))))
    )

# Apply to both columns
bd['categories_bd_norm'] = normalize_categories(bd['categories_bd'])
ts['categories_ts_norm'] = normalize_categories(ts['categories_ts'])

# Block on categories to reduce candidate pairs
indexer = Index()
indexer.block(left_on='categories_bd_norm', right_on='categories_ts_norm')
candidates = indexer.index(bd, ts)
```

In [40]:
```python
# Block on categories to reduce candidate pairs
indexer = Index()
indexer.block(left_on='product_name', right_on='name')
candidates = indexer.index(bd, ts)
```

In [41]:
```python
# For each candidate pair, compute fuzzy score on `combined`
matches = []
scores = []
```

In [42]:
```python
for bd_idx, ts_idx in tqdm(candidates, desc="Blocking + Fuzzy matching"):
    bd_text = bd.at[bd_idx, 'combined']
    ts_text = ts.at[ts_idx, 'combined']
    score = fuzz.token_set_ratio(bd_text, ts_text)
    if score >= 85:
        matches.append((bd_idx, ts_idx))
        scores.append(score)
```

```
Blocking + Fuzzy matching:   0%|          | 0/9440 [00:00<?, ?it/s]
```

In [43]:
```python
# Build a DataFrame of matched index-pairs + score
matches_df = pd.DataFrame(matches, columns=['bd_idx','ts_idx'])
matches_df['match_score'] = scores
```

In [44]:
```python
# Extract matched rows, reset index to align with matches_df
bd_matched = bd.loc[matches_df['bd_idx']].reset_index(drop=True)
ts_matched = ts.loc[matches_df['ts_idx']].reset_index(drop=True)
bd_matched.index = matches_df.index
ts_matched.index = matches_df.index
```

In [45]:
```python
# Fuse matched rows into one DataFrame (suffixing columns)
matched_full = pd.concat([
    bd_matched.add_suffix('_bd'),
    ts_matched.add_suffix('_ts')
], axis=1)
matched_full['match_score'] = matches_df['match_score']
matched_full['source'] = 'matched'
```

In [46]:

```
# Prepare TS-only records
ts_unmatched_idx = ts.index.difference(matches_df['ts_idx'])
ts_unmatched = ts.loc[ts_unmatched_idx].reset_index(drop=True)
ts_only = ts_unmatched.add_suffix('_ts')
ts_only['match_score'] = pd.NA
ts_only['source'] = 'ts_only'
# add blank BD columns
for col in bd.columns:
    ts_only[col + '_bd'] = pd.NA

# Prepare BD-only records
bd_unmatched_idx = bd.index.difference(matches_df['bd_idx'])
bd_unmatched = bd.loc[bd_unmatched_idx].reset_index(drop=True)
bd_only = bd_unmatched.add_suffix('_bd')
bd_only['match_score'] = pd.NA
bd_only['source'] = 'bd_only'
# add blank TS columns
for col in ts.columns:
    bd_only[col + '_ts'] = pd.NA
```

In [47]:

```
# Reorder TS-only and BD-only to match matched_full columns
all_cols = list(matched_full.columns)
ts_only = ts_only[all_cols]
bd_only = bd_only[all_cols]
```

In [48]:

```
# Concatenate into final master catalogue
master_catalogue_hybrid = pd.concat(
    [matched_full, ts_only, bd_only],
    ignore_index=True
)

# eorder columns: put product identifiers front
cols = (
    ['product_name_bd','name_ts']
    + [c for c in all_cols if c not in ('product_name_bd','name_ts')]
)
master_catalogue_hybrid = master_catalogue_hybrid[cols]
```

```
/var/folders/vd/k74_1qd12wv9pzhrjx9lkbzr0000gn/T/ipykernel_17010/3305352965.
py:2: FutureWarning: The behavior of DataFrame concatenation with empty or a
ll-NA entries is deprecated. In a future version, this will no longer exclud
e empty or all-NA columns when determining the result dtypes. To retain the
old behavior, exclude the relevant entries before the concat operation.
  master_catalogue_hybrid = pd.concat(
```

In [49]:

```
master_catalogue_hybrid.to_csv('data/master_catalogue_hybrid.csv', index=Fal
display("Master catalogue shape:", master_catalogue_hybrid.shape)
display(master_catalogue_hybrid.head())
```

```
'Master catalogue shape:'
(106029, 29)
```

	product_name_bd	name_ts	description_bd	seller_description_bd	seller_
0	Mighty Networks	Mighty Networks	Mighty is where creators, entrepreneurs, and b...	Mighty Networks is a platform that enables cre...	mightyn
1	Microsoft Purview Information Protection	Microsoft Purview Information Protection	Protect your sensitive information. Learn how ...	Every company has a mission. What's ours? To e...	microsc
2	Microsoft Power Apps	Microsoft Power Apps	Create apps that bring together the services y...	Every company has a mission. What's ours? To e...	microsc
3	Microsoft Dynamics 365 Supply Chain Management	Microsoft Dynamics 365 Supply Chain Management	Microsoft Dynamics 365 for Operations is the c...	Every company has a mission. What's ours? To e...	microsc
4	Microsoft Project Server	Microsoft Project Server	Microsoft Project Server 2013 is a flexible on...	Every company has a mission. What's ours? To e...	microsc

5 rows × 29 columns

In [50]:

```
# display master catalogue with matched entries and the info
matched_stats_hybrid = master_catalogue_hybrid[master_catalogue_hybrid['sour
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2148 entries, 0 to 2147
Data columns (total 29 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   product_name_bd               2148 non-null   object 
 1   name_ts                       2148 non-null   object 
 2   description_bd                2148 non-null   object 
 3   seller_description_bd         2148 non-null   object 
 4   seller_website_bd             2147 non-null   object 
 5   main_category_bd              2148 non-null   object 
 6   software_product_id_bd        2148 non-null   object 
 7   overview_bd                   2148 non-null   object 
 8   headquarters_bd               1591 non-null   object 
 9   categories_bd                 2148 non-null   object 
 10  categories_bd_bd              2148 non-null   object 
 11  combined_bd                   2148 non-null   object 
 12  categories_bd_norm_bd         2148 non-null   object 
 13  technology_id_ts              2148 non-null   object 
 14  slug_ts                       2148 non-null   object 
 15  url_ts                        262 non-null    object 
 16  description_ts                291 non-null    object 
 17  category_ts                   2148 non-null   object 
 18  category_slug_ts              2148 non-null   object 
 19  parent_category_ts            2148 non-null   object 
 20  parent_category_slug_ts       2148 non-null   object 
 21  jobs_ts                       2148 non-null   object 
 22  companies_ts                  2148 non-null   object 
 23  companies_found_last_week_ts  2148 non-null   object 
 24  categories_ts_ts              2148 non-null   object 
 25  combined_ts                   2148 non-null   object 
 26  categories_ts_norm_ts         2148 non-null   object 
 27  match_score                   2148 non-null   float64
 28  source                        2148 non-null   object 
dtypes: float64(1), object(28)
memory usage: 503.4+ KB
```

In [ ]:

```

```