Product Matching Using TF-IDF And Cosine Similarity

KHILTI DEDHIA, Dublin City University, Ireland

Zalando is a multinational retail company based in Germany. It has a variety of fashion and lifestyle products for customers to choose in nearly 23 countries. Zalando offers a dynamic range of products with multiple offers from various sellers for the customers to choose from. One such competitor is a shop called Aboutyou who has similar products just like Zalando. The main task is to map the products from different shops based on the text data available in the dataset such as title, color, brand and/or description. To go about the text classification, I have used a python library called string_grouper that finds similarity between the titles of the products using cosine similarity metric over the term frequency-inverse document frequency (TF-IDF). One can set the weights of the vectors high to get accurate predicted matches.

Keywords: *cosine similarity, TF-IDF, text classification*

1 INTRODUCTION

With the rapid growth of computer technology and the insane use of the internet these days, a large amount of data is generated every day from e-commerce, social media, science, daily life, and so on and is fed into the computers constantly. Talking about e-commerce, who does not love online shopping on various platforms such as Amazon? One such successful e-commerce shopping company is Zalando that offers a tremendous variety of fashion and lifestyle products. Zalando is a retail company that offers millions of products from different sellers and brands, without compromising with quality or prices. Zalando wants to offer competitive prices throughout their environment to ease the customer from comparing prices between brands or competitors. A shop called Aboutyou offers to present their products on the same platform due to which identical groups need to be paired together. The main challenge is to plan and map similar products from different shops using text with the attributes: title, colour, brand, description. Text classification plays a key role in extracting information since the titles, brands or descriptions might not be the same. There happens to be several methods for text classification; I chose to use the term document-inverse document frequency (TF-IDF) using which word vectors will be generated. To find the similarity between these vectors, the cosine similarity metric calculates the cosine of the angle between the vectors. If the returned value from this method is 0, matches are dissimilar and if the value is 1, matches are similar.

2 RELATED WORK

There has been a lot of advancements in text classification problems. TF-IDF has proved to be the most useful when it comes to extracting keywords from documents, strings, or lists. There have been several similarity measures such as Jaro similarity, Euclidian distance, Laplace distance, Levenshtein distance similarity, etc. Cosine similarity proves to be the fastest and most accurate measure of similarity. Defit and Nurcahyo (2021) used TF-IDF and cosine similarity for their application to get the maximum search accuracy. They claim to have achieved high accuracy with this approach. Liu et al. (2019) TF-IDF and PTF-IDF (Promoted-TF-IDF) along with cosine similarity for text classification. They found to have gotten better accuracy using PTF-IDF rather than using the traditional TF-IDF. Alodadi and Janeja (2015) used TF-IDF and cosine similarity to find similarity in patient support forms achieving 63% accuracy.

3 DATASET

Zalando dataset is divided into 3 files, training set, test set and matches made out of the training set. All the dataset files are in the parquet format and contain the offers of products of two brands, Zalando and Aboutyou. The training and test dataset contains 10 attributes whilst the matches dataset contains only 3 attributes. The training and test dataset comprises of the following attributes:

Attribute	**Description**
offer_id	Unique identifier 'id' for a particular offer of a product
shop	'Zalando', 'brand'
lang	The language, German (de)
brand	The brand of the product that can be common to both the shops, for e.g., Quiksilver
colour	Colour of the product, for e.g., blue or black
title	Brief title of the product, for e.g., Snowboard Jacket
description	Detailed product description
price	Price of the product (in Euro)
url	URL of the product's page
image_urls	List of images of the product such as with model, image taking at a different angle, etc.

Table 1. Attributes and their description of the dataset.

The matches dataset contains only the offer_id of the Zalando product, offer_id of the Aboutyou product that matches with the Zalando product and their brands. The training dataset consists of 61,980 Aboutyou products and 40,904 Zalando products. The test dataset contains 70,105 Aboutyou products and 36,636 Zalando products. The matches dataset contains 15,170 products' offer_id. Also, all the 3 datasets are in German language.

4 EXPLORATORY DATA ANALYSIS

The jupyter notebook that was provided helped me a lot in understanding and interpreting the datasets. In the test dataset, there were 65.7% Aboutyou products and 34.3% Zalando products as show in the figure below.

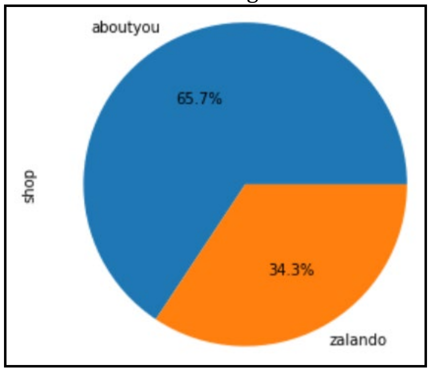

Fig. 1. Distribution of Zalando and Aboutyou's products in the dataset

Figure 2 shows the maximum prices of the products mainly ranged from 100 euros to 300 euros. As shown in figure 2, most of the products are priced in the range of 5 euros to 88 euros.

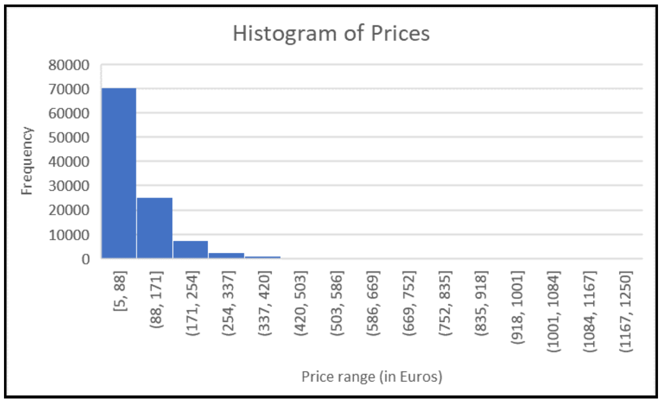

Fig. 2. Histogram of the prices of products

Finally, I also checked the number of various products such as jeans in both the shops as shown in figure 3.

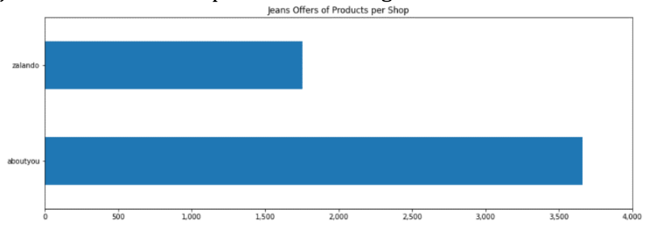

Fig. 3. Jeans offers of Products per Shop

5 METHODOLOGY

For this task, we were asked to work on the test dataset provided. Upon spending enough time on exploring the datasets and gaining a thorough knowledge of them, I strategized to go forward with the task in the following manner:

A. **Text pre-processing**:
The dataset requires a lot of text cleaning. The first step is to convert the text to lower case to compute smoothly. Then, I removed punctuations, if any, that were present in the text columns of the datasets. After removing punctuations, I removed the stop words, which happen to be the most frequently used words in the document that are not of any significance. Finally, I performed stemming on the text which is the process of removing the end of words to return to their original form. For the removal of stop words and stemming, I used the python library NLTK which is a Natural Language Tool Kit which has various inbuilt algorithms for text-preprocessing. Finally, I split the dataset into two sub datasets based on the shops, one for Zalando and one for Aboutyou

B. **TF-IDF and Cosine similarity**:
Term Frequency is the number of times a particular word occurs in a document. *Inverse Document Frequency* is supposed to reflect how important a word is in a document. In other words, it is an eloquent way of knowing if the word is frequent or rare in a document. So, if the word is common and appears in several documents, the value will tend to 0. TF-IDF is a useful method for keyword extraction. TF-IDF vectorizer then converts the text into a vector. *Cosine Similarity* measures the similarity between two vectors. It measures the cosine of the angle made by vectors and checks if they are pointing at the same direction. Python provides a library called string_grouper that easily and rapidly matches groups of strings with a single or multiple lists or strings. It uses TF-IDF to compute the cosine similarity within a single list or between multiple lists. This library contains 4 main functions, match_strings, match_most_similar, group_similar_strings, and compute_pairwise_similarities that perform efficiently. Each function has its own beauty of working. The function match_strings take the title of both the split datasets as the input to check similarities between them and returns the left_title, right_title and similarity score. The left_title is the title of the first dataset and the right_title is the title of the second dataset. The minimum similarity score has been set default to 0.8 and goes up to 1.0, i.e., exact matches.

C. After acquiring the matches, I removed the matches with similarity scores of 1.0. The match_string function also checks for matches by itself generating self matches. For self matches, I checked if the left_title and right_title had the same shops and decided to drop such matches. Finally, I obtained their respective offer_ids to put in a separate file named matches_test_predicted.

6 RESULTS

Once I found the matches, I used the provided function 'plot_images' to check whether the matches are true matches using images. The matches that I obtained have a similarity score of 0.8 and above which means the accuracy is 80% and above.

7 REFERENCES

[1] Mohammad Alodadi and Vandana P. Janeja. 2015. Similarity in patient support forums using TF-IDF and cosine similarity metrics. In *2015 International Conference on Healthcare Informatics*, IEEE.

[2] Chris van den Berg. *string_grouper: Super Fast String Matching in Python*.

[3] Yunxiang Liu, Qi Xu, and Zeshen Tang. 2019. Research on text classification method based on PTF-IDF and cosine similarity. In *2019 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)*, IEEE.

[4] Sintia Sintia, Sarjon Defit, and Gunadi Widi Nurcahyo. 2021. Product codefication accuracy with Cosine Similarity and weighted term Frequency and Inverse Document Frequency (TF-IDF). *Journal of Applied Engineering and Technological Science (JAETS)* 2, 2 (2021), 62–69.