"NLP course project Clothing store Recommendation system "

ENCS, Computer Systems Engineering, Professor Adnan Yahya

Birzeit University, Faculty of Engineering and Information Technology

Jana Herzallah	Ahmed Zubaidia	Lana Badwan
1201139	1200105	1200071

I. Abstract

This paper proposes a recommendation system for an online clothing store with the use of the information retrieval and natural language processing concepts. The system has the search bar option that lets users to make queries through while also taking their past purchases into consideration. Different models have been trained and tested on the H&M dataset. These models vary between the representation of the queries and the product descriptions in the store. Evaluation results demonstrates the system's strength in making personalized recommendations for each user.

II. Introduction

Everything is becoming digital and smart in today's growing world. Shopping process has become an online procedure lastly. To make shopping easier and more enjoyable for users, we are proposing a shopping smart system that hires Information Retrieval (IR) and Natural Language Processing (NLP). We have a goal to create a system that suggests a very personalized and customized recommendations for each user for clothes when he/she is shopping online.

III. Theory and Literature Review

The main theoretical fields that this paper relies on are:

1. **Similarity measures between documents**: cosine similarity was considered in calculating the similarity between the documents which is defined by:

$$\text{Cosine Similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

Equation 1: cosine similarity expression

Where A.B is the dot product between the vectors A and B, while |A| and |B| are the magnitudes of the vectors [2]. In our system, it has been used to measure the similarity between the user queries and the products description, also between the past purchased items and available products.

2. Vector representation

Queries have been represented by various types of vectors such as word2Vec, bert , tf-idf and count vectorizer. Starting off with the word2Vec vectors, it converts the words of the queries into embeddings, each word gets represented with a defined length. And its worth saying that embeddings of semantically similar words are similar but it doesn't considerate the context. However, [3] Bert model can generate different embeddings for the same word regarding the context. Another types of representing queries are the matrix of the words and documents and each index can be represented by the value zero or one depending on the presence of the word in the document which is basically the concept of count vectorizer model. And the last way is the same but represented by the tf-idf values of each word instead of zeros or ones.

3. Query expansion

Since queries are short usually represented by 4-5 words, it's crucial to expand the length of it by adding terms that will enrich the searching process. These terms can be the same meaning or similar to the original words used by the user using the model of W2vec or they can be according to the context using Bert embeddings.

IV. Dataset Description

The proposed system was built and trained on the H&M dataset from Kaggle[1], it consists of three main parts; customers, articles, images and transactions datasets

1. **Articles Dataset**: 105126 rows with 25 columns. Each item was identified by the "article id" and classified into group of products to understand the dataset as shown in figure [1].

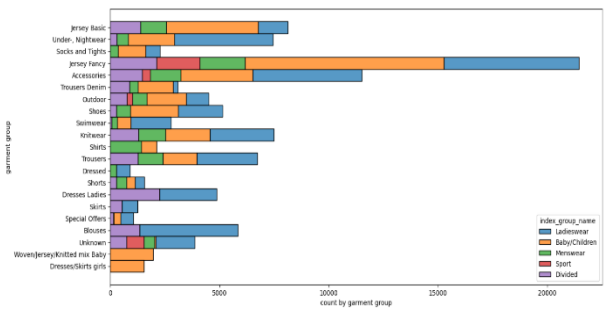

Figure 1: products group names

It's noticed that ladieswear take a significant part of the store products. The text columns of the 25 belonging to the item such as "detail_desc" and "product_group_name" and others shown in the code were concatenated in one column called text that has all the information needed to describe the product. and this huge column was a major one used in the main data frame that has the article id and text columns that got merged.

2. **Customer's dataset:** 1371980 rows were identified by the column "customer id" with some attributes for each customer such as age, postal code and other coulmns.
3. **Transaction's Dataset:** It has 31788324 entries, 0 to 31788323, the important columns were extracted to be used in the training. Each row was labeled by the transaction date, customer id, article id and price of the bought item.
4. **Images Dataset:** it consists of a folder of images related to article id, some of them doesn't have a description for them.

V. System Description: main flows of the program

Our program has two main flows that users can select between one that allows them to search for the products that they write a description for and another one that the program will recommend products based on the past purchases of the user it if it's not the first time purchasing something. However, if it's the user first time purchasing an item, and the user doesn't have a specific item that he/she wants to specifically search for, then the system allows to recommend him/ her the most bought item by other users in the closest time.

Also, we have built a model that combines both algorithms which shows strength when the past purchased items are similar to the searched items. It's a little bit weaker when the customer searches for a dissimilar item which are very far away from the past purchased items so talk about the factorization

The algorithms used in our system will be shown in the upcoming sections.

1. First flow: based on query alone

This flow has been trained on multiple models; all of the models have the in common the Arabic search bar along with the English search bar.

If the user makes the query in Arabic, it gets corrected through the model of **SpellChecker** along with the help of **Levenshtein** that finds the closest word to the typed word and then after its corrected it gets translated to English with the help of **GoogleTranslator** model. If the user makes the

query in English, it also gets corrected using **TextBlob** model for correcting the spelling mistakes for English along with the help of **Levenshtein** that finds the closest word to the typed word. So, after that we have the query corrected in English from both scenarios, so it gets cleaned from stop words and punctuations through the help of the Natural Language Toolkit -**NLTK-** resources as **wordnet** used for lemmatization which is reducing the words for their bases, **stopwords** corpus model used to filter the non-informative words and **punkt model** used for tokenizing text. Mainly, the correction and stemming parts were very beneficial but for some clothing items that are not that popular it messed up a little bit. Another thing that all models share is the expansion of the query. The idea behind that was about how the query is short and the document is long that has the text values so we needed to expand them with some keywords to make the cosine similarity between the query and products description has more understood result and we have used the Bert embeddings model to expand the queries and then three models have been applied to them as follows:

First model: TF-IDF representation of the query

After the query entered by the user gets corrected and cleaned, it gets expanded with the Bert model embeddings that converts each word to a vector according to the context. After that the TF-IDF matrix is generated out of the embeddings and the cosine similarity is measured between this matrix and the TF-IDF matrix of the Bert embeddings of the description of available products in the store and the highest score was returned to the user.

Second model: incidence matrix representation of the query:

After expanding the query with Bert embeddings, the incidence matrix is generated for it using the **count vectorizer** model and the cosine similarity is measured between this matrix and the count vectorizer matrix of the description of available products in the store and the highest score was returned to the user.

Third model: Bert Embeddings representation of the query:

After expanding the query with Bert embeddings, the cosine similarity is measured between these embeddings and the dataset Bert embeddings of the description of available products in the store and the highest score was returned to the user.

2. **Second flow: only past purchases**

This model only fetches the past purchased items text features after joining all the purchases made by the specific customer from the transactions dataset for the customer and get its embeddings calculated through Bert model and calculate the similarity between them and between the products in the store.

The products with the highest similarities scores were returned to the users and recommended by the system. And the results returned are discussed in the upcoming sections

3. **Third flow: based on query and past purchases merged**

This model fetches the past purchased items for a customer and generates its Bert embeddings along with taking the query embeddings into consideration and combining the Bert embeddings for them and then measuring the cosine similarity between them and the products in the store. so, the result will be a combination of the query search words and the past purchases description.

4. **Fourth flow: Most bought items in the store**

This model fetches the most purchased items in the store and recommends them to the user's when they don't want to search for anything in the search bar.

VI. Images Handling:

An HTTP GET request was made to google search engine through an API to fetch pictures according to the provided

query. This has been done because the images provided in our dataset was unlabeled and not all articles have a picture. So, this fast solution was a good one to show the idea of our project and got the concept of showing pictures with the texts.

VII. Evaluation; people testing then analyzing their reports.

These models have been tested and verified by people who haven't participated in the code. They gave some queries to the system and reported their experience and whether they found the returned items were relevant or not. Also, they were told to rank the models based on the most relevant ones. So, we can understand the strengths and weaknesses of each one. And their evaluation report is linked down below.

And the results we got back from them NLP-Evaluation - Google Drive[4] along with our testing through the journey of building the model are analyzed down below for each model:

VIII. Results of the system

a. Results from past purchased items recommendation

For a with customer id=" 001ae5408a043f64bccd32beffe2730151414cbdf18a6eb3cc8d30bdca605652", the past purchased items had so much features in common with the ones recommended by the system. Both lists focused on items as nightwear sets highlighting comfort with the elasticated waists and soft materials. While also focusing on the same colors, it has been noticed that both lists had solid colors such as black, white, light grey and dark blue. Another matched feature was regarding the materials and fabrics which both lists had jersey and weave and its also registered in the testing report. For the upper body garment, both lists had tops and blouses with the focus on specific features as narrow shoulder straps and button fronts. For the lower body garments, both of them had trousers denim and joggers. Lastly, the full body garments were similar so both recommended jumpsuits, dresses and sets.

b. Results from making queries

All models returned a textual description of the products in the store ranked according to the most recommended ones by the system along with some images fetched by the search engine in google since the images in our dataset were not appropriately labeled and dealing with them was not easy. And this can be one of the major improvements that we can do in the future. It has been noticed that longer queries have more understandable effect to the system, hence more relevant items. Also, describing the item with mentioning the body part related to it such as lower, upper and full body helps the system with recognizing the items in a better way. And finally, when the user only enters 2-3 words, we have noticed duplicating the query before expanding it with Bert embeddings worked greatly.

c. Results of recommendation based on query and past purchases merged

The model returned a textual description of the most recommended products by the system according to the past purchased items description and features along with the query embeddings along pictures having a related description of the recommended items. It's worth saying that the model got biased towards the past purchased items because the documents were much longer than the query made by the user. so, we multiplicated the query by a number of times proportional to the length of the past purchased items document until it started taking the query into more consideration and started giving more relevant items.

IX. Conclusion and future improvements and challenges

To conclude, the TF-IDF model shows great result for the exact same words searched for. While the Bert model showed better search results for longer queries and for those which have a contextual term. While the system that merges past purchases showed strength when the buyer searched for items that is similar to the ones, he/she already bought before.

For the Images, we can in the future build a model that extracts the images features and generate an appropriate description of them. Then this description gets used to be calculated the similarity between them and the searched item description Instead of using the images supported by the search engine of google.

Our vision is to create a system that selects the most appropriate model based on the query itself. So, if the query is short and the user wishes to get the exact same description then the system should pick the model of TF-IDF, while if the user makes a query that has contextual meaning and the system detects that the search results will be more precise if the query terms have been embedded with similar words, then the system should use the Bert embeddings query model.

X. References

- [1] H&M Dataset," Commits · jianchann/H-M-Dataset · GitHub", 2022, "nlb_project_dataset - Google Drive".
- [2] Tan Thongtan, Tanasanee Phienthrakul, "Sentiment Classification Using Document Embeddings Trained with Cosine Similarity | Papers With Code",2019
- [3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova "BERT Explained | Papers With Code ",3-4, May, 2019.
- [4] "NLP-Evaluation - Google Drive"