JustSnap

Data analyst Assigment

Solution of: **Konstantinos Christogeorgos**

E-Mail: **kostischristogeorgos@gmail.com**

Contents

Introduction

The problem and the approach

The company faced major challenges: manual product matching required 2–3 hours each day, caused order-fulfillment delays, introduced stock inaccuracies, led to lost sales when similar items weren't identified, and provided no systematic way to track product demand.

My approach solves these issues by fully automating the matching process using NLP, embeddings and attribute extraction, accurately identifying the correct SKUs and alternatives, dramatically reducing manual workload while improving speed, accuracy and overall operational efficiency.

Data involved

For the project I was given two datasets.

1. Inventory data, with features per product.
2. Unstructured descriptions of each product by sales teams .

General Approach

I plan to design a fully automated pipeline that cleans and normalizes customer descriptions, extracts key attributes using NLP and fuzzy logic that then converts both descriptions and products into embedding vectors for semantic similarity search. I am aiming to do a hard filtering based on extracted attributes (category, subcategory, season, year) to narrow the candidate set, then used cosine similarity and a re-ranking the retrieved items based on matches for size, color, material, brand and features, resulting in the final confidence level per item.
Finally, I will generate the top-4 recommendations with confidence scores and structured json outputs, enabling fast, accurate and consistent SKU identification.

Preprocess

Preprocess of the products dataset

All in all, the products were stored in a very structed and clean way, leaving little room for changes.

Yet, I implemented changes in the next steps:

- Transformed all text into lower case in order to have all data in the same type.
- Instead of the character ' | ' to split the features of each item, I split the features using spaces.
- The season feature had data like "Fall 2025", I kept the season as "Fall" and created a new feature called Year, to store 2025.
- I extracted all the unique values of brands, materials, features and categories/subcategories to use later for the matching with the descriptions.
- I replaced all nan values with blank spaces, since the values before were the string 'nan'.
- I created a new feature called full_text that will be used to create the embeddings. This feature has all the data of the product in a dictionary style. For example "product name, brand: actual brand name, material: actual material …" and so on.
  Below there is an example for the first 3 descriptions:

	text_full
0	nordic jacket, brand nordic, outerwear, jacket, color brown, size xl, material polyester, features stretch breathable, season fall 2025
1	elite coat, brand elite, outerwear, coat, color tan, size m, material linen, features moisture-wicking, season winter 2024
2	premium cardigan, brand premium, outerwear, cardigan, color black, size m, material fleece, features quick-dry, season spring 2025

By producing this feature, the semantic search will be able to match a very precise embedding per product.

Preprocess of the descriptions dataset

The descriptions had the following preprocesses for feature extraction:

- Transformed all text into lower case in order to have all data in the same type.
- For **colors**, using fuzzy logic and a list of common colors all the colors in the description were extracted. **Fuzzy logic is used since there were typos in some colors like gren, or blu.**
- For **brand** extraction, fuzzy logic was used in conjunction with the unique values of the brands extracted from the products dataset. Essentially it tries to match text with brands

found in the dataset. Fuzzy logic is used here because there are some typos again. For example, the brand ‘essential’ is often spelled ‘esential’.

- Fuzzy extraction is also used for the **materials, features** and **categories/subcategories** for the same reasons.
- Feature extraction of **Season, Year** and **Size** is done by regex match.
- If a subcategory is matched, then the category is automatically matched, using a dictionary. This does not work the other way around, since we can’t infer the subcategory from the category.

In the next screenshot I showcase the extracted features from each description for the first 4 records. As it can be seen, the extraction works perfectly.

	description_clean	size	color	season	year	brand	material	category	subcategory	features
0	do you carry style pants for fall 2025	None	None	fall	2025	style	None	bottoms	pants	None
1	looking for green boots size l	l	green	None	None	None	None	footwear	boots	None
2	searching for the modern blazer finished in dark grey xxl size	xxl	grey	None	None	modern	None	outerwear	blazer	None
3	need blue sundress that is breathable and flexible for fall 2024 size m	m	blue	fall	2024	None	None	tops	blouse	breathable

Some interesting cases are presented below:

10	good afternoon do you happen to have the olive gown from urban in size xs thanks in advance	xs	olive	None	None	urban	None	dresses	gown	None

This can perfectly match the not so common color ‘olive’.

7	need essetnials burundy boluse sz xs asap	xs	burgundy	None	None	essential	None	tops	blouse	None

Although the brand is typed as ‘essetnials’, the algorithm matches it to essential correctly.

The dataset with all the extracted features is supplied in the project files as a .csv.

Matching algorithm

After the preprocess is done, using the 'all-mpnet-base-v2' embedding model I encode the **full_text** feature and the **unstructured_description** into **embeddings**.

The algorithm operates through a sequence of structured steps. **First**, it takes as **input** an **unstructured description** and the **features extracted** together with the pre-computed **embeddings** for all **product** and **descriptions**, as well as the **full product catalog**.
The process begins with **hard filtering**, where products are filtered by **category**, **subcategory**, **season** and **year** extracted from the description. These attributes are **highly specific**, so restricting the search space to this subset improves accuracy. If no products meet these conditions, the algorithm falls back to using the full catalog to avoid returning empty results.

Next, a **cosine similarity** search is performed between the **description embedding** and the **embeddings of the filtered products**. The algorithm selects the **top fifty** candidates based purely on semantic similarity. These candidates are then re-ranked using attribute-based boosts, where matches on size, color, material, brand, or product features increase a product's score and thus, its confidence. The final confidence score, scaled between 0 and 100 percent, **reflects both semantic similarity and attribute alignment.**

After re-ranking, the algorithm selects the **top four products**. The **first item** is treated as the **primary predicted** match, while the next **three** are returned as the most **plausible alternatives**. If fewer than four products remain after filtering, the algorithm supplements the list with the closest matches based on raw cosine similarity. Finally, the system outputs a structured **JSON** record containing the cleaned description, the extracted attributes, and the selected products along with their full metadata and confidence scores.

Reasoning behind this algorithm

The reasoning behind this algorithm is to combine the strengths of semantic understanding with the precision of structured product attributes. Sales descriptions are often incomplete, noisy, or misspelled, so relying strictly on direct text matching would lead to frequent errors. Embeddings and cosine similarity provide a way to understand the semantic meaning of a query and retrieve products that are conceptually related, even when the wording is inconsistent.

The combination of filtering before and after the semantic also adds rubustness. For this reason, the algorithm performs hard filtering first, ensuring that only products matching the essential extracted attributes are considered (Season, Category). This prevents the model from offering irrelevant suggestions and significantly narrows the search space.

The re-ranking stage adds another layer of refinement by rewarding products that match additional attributes such as material, brand, or product features. This step injects domain knowledge into the ranking by giving priority to products that are not only semantically similar to the query but also structurally aligned with the user's requirements.

Validation & Performance Analysis

I created 30 descriptions similar to the original dataset, making sure I add typos, broad meanings and non-ordinary colors.

This file is the "test_descriptions.csv" and can also be found in the project files. This file also has the label for the correct product, so we can check the metrics.

Below is a full screenshot of the data.

	Description_ID	Unstructured_Description	Source_Channel	Label
0	DESC0001	do you have the esential polo in blu, size m?	Website	SKU1000206
1	DESC0002	looking for brown shorts from alpine in xs.	Chat	SKU1000398
2	DESC0003	need cream vest that is windprof for winter 2025, size xs.	Phone	SKU1000095
3	DESC0004	want gray option in dresses, open to suggestions.	Website	SKU1000594
4	DESC0005	after cream option in s, preferably polo.	Email	SKU1000308
5	DESC0006	hi, i'm checking if you carry the white gown from nordic in size xl? thanks!	Marketplace	SKU1000613
6	DESC0007	looking for a red gown by classiz, size s.	Website	SKU1000502
7	DESC0008	do you stock essential's parka in cream? i need xs.	Chat	SKU1000118
8	DESC0009	need olive loafers from urban sz s asap.	Email	SKU1000773
9	DESC0010	good afternoon, do you happen to have the brown shorts from nordic in size xl? appreciate your help.	Phone	SKU1000388
10	DESC0011	want white jacket for summer 2025, any brand works.	Website	SKU1000036
11	DESC0012	after white option in s, preferably sneakers.	Marketplace	SKU1000721
12	DESC0013	looking for beige gown from urban in xl, ideally cotton.	Chat	SKU1000487
13	DESC0014	do you carry nordic pants for summer 2024?	Email	SKU1000405
14	DESC0015	need tan gown that is breatheble for fall 2024, size xs.	Website	SKU1000607
15	DESC0016	good afternoon, could you help me find the olive turtleneck from alpine in size m? thanks in advance!	Phone	SKU1000303
16	DESC0017	want olive loafers with wool, m size for fall 2025.	Chat	SKU1000738
17	DESC0018	do you have the nordic sandls in white, size m?	Website	SKU1000727
18	DESC0019	looking for tan vest, size l.	Marketplace	SKU1000041
19	DESC0020	after red option in m, preferably blazer.	Email	SKU1000069
20	DESC0021	need burgundy sundress from nordic sz s asap.	Website	SKU1000593
21	DESC0022	hi, i'm checking if you carry the charcoal shorts from nordic in size xxl? appreciate your help!	Chat	SKU1000448
22	DESC0023	want navy skirt for summer 2024, any brand works.	Phone	SKU1000417
23	DESC0024	looking for rain-ready red blazer in l, ideally polyester.	Website	SKU1000159
24	DESC0025	do you stock essential's blue sundres in that shade? i need m.	Marketplace	SKU1000563
25	DESC0026	good afternoon, do you happen to have the white turtleneck from classic in size m? thanks!	Email	SKU1000279
26	DESC0027	essantial navy option in xs, preferably dress.	Chat	SKU1000543
27	DESC0028	need black jeans from alpine sz xs asap.	Website	SKU1000451
28	DESC0029	want black option in dresses, open to suggestions.	Phone	SKU1000612
29	DESC0030	looking for olive sundress from comfort in xxl, ideally cashmere.	Marketplace	SKU1000623

The same preprocess as in the original descriptions is done, producing all the features extracted possible. They can be seen in the next picture:

	description_clean	size	color	season	year	brand	material	category	subcategory	features	Label
0	do you have the esential polo in blu size m	m	blue	None	None	essential	None	tops	polo	None	SKU1000206
1	looking for brown shorts from alpine in xs	xs	brown	None	None	alpine	None	bottoms	shorts	None	SKU1000398
2	need cream vest that is windprof for winter 2025 size xs	xs	cream	winter	2025	None	None	outerwear	vest	windproof	SKU1000095
3	want gray option in dresses open to suggestions	None	gray	None	None	None	None	dresses	dress	None	SKU1000594
4	after cream option in s preferably polo	s	cream	None	None	None	None	tops	polo	None	SKU1000308
5	hi im checking if you carry the white gown from nordic in size xl thanks	xl	white	None	None	nordic	None	dresses	gown	None	SKU1000613
6	looking for a red gown by classiz size s	s	red	None	None	classic	None	dresses	gown	None	SKU1000502
7	do you stock essentials parka in cream i need xs	xs	cream	None	None	essential	None	outerwear	parka	None	SKU1000118
8	need olive loafers from urban sz s asap	s	olive	None	None	urban	None	footwear	loafers	None	SKU1000773
9	good afternoon do you happen to have the brown shorts from nordic in size xl appreciate your help	xl	brown	None	None	nordic	None	bottoms	shorts	None	SKU1000388
10	want white jacket for summer 2025 any brand works	None	white	summer	2025	None	None	outerwear	jacket	None	SKU1000036
11	after white option in s preferably sneakers	s	white	None	None	None	None	footwear	sneakers	None	SKU1000721
12	looking for beige gown from urban in xl ideally cotton	xl	beige	None	None	urban	cotton	dresses	gown	None	SKU1000487
13	do you carry nordic pants for summer 2024	None	None	summer	2024	nordic	None	bottoms	pants	None	SKU1000405
14	need tan gown that is breatheble for fall 2024 size xs	xs	tan	fall	2024	None	None	dresses	gown	breathable	SKU1000607
15	good afternoon could you help me find the olive turtleneck from alpine in size m thanks in advance	m	olive	None	None	alpine	None	tops	turtleneck	None	SKU1000303
16	want olive loafers with wool m size for fall 2025	m	olive	fall	2025	None	wool	footwear	loafers	None	SKU1000738
17	do you have the nordic sandls in white size m	m	white	None	None	nordic	None	footwear	sandals	None	SKU1000727
18	looking for tan vest size l	l	tan	None	None	None	None	outerwear	vest	None	SKU1000041
19	after red option in m preferably blazer	m	red	None	None	None	None	outerwear	blazer	None	SKU1000069
20	need burgundy sundress from nordic sz s asap	s	burgundy	None	None	nordic	None	dresses	sundress	None	SKU1000593
21	hi im checking if you carry the charcoal shorts from nordic in size xxl appreciate your help	xxl	charcoal	None	None	nordic	None	bottoms	shorts	None	SKU1000448
22	want navy skirt for summer 2024 any brand works	None	navy	summer	2024	None	None	bottoms	skirt	None	SKU1000417
23	looking for rainready red blazer in l ideally polyester	l	red	None	None	None	polyester	outerwear	blazer	None	SKU1000159
24	do you stock essentials blue sundres in that shade i need m	m	blue	None	None	essential	None	tops	blouse	None	SKU1000563
25	good afternoon do you happen to have the white turtleneck from classic in size m thanks	m	white	None	None	classic	None	tops	turtleneck	None	SKU1000279
26	essantial navy option in xs preferably dress	xs	navy	None	None	essential	None	dresses	dress	None	SKU1000543
27	need black jeans from alpine sz xs asap	xs	black	None	None	alpine	None	bottoms	jeans	None	SKU1000451
28	want black option in dresses open to suggestions	None	black	None	None	None	None	dresses	dress	None	SKU1000612
29	looking for olive sundress from comfort in xxl ideally cashmere	xxl	olive	None	None	comfort	cashmere	dresses	sundress	None	SKU1000623

This data is then embedded using the model and then fed into the matching algorithm.

Now there is also metrics about top1-top3 and precision/recall calculated presented below:

```
Top-1 Accuracy: 93.33%
Top-3 Accuracy: 96.67%
Precision: 87.50%
Recall: 87.50%
```

As we can observe the algorithm works exceptionally well, having a 93.33% accuracy in top-1 matches and 96.67% accuracy in top-3 matches.

Precision and recall (macro) are also strong at 87.50%

In the next picture we can observe the json file output for the first description.

```
{'description_index': 0,
 'description': 'do you have the esential polo in blu size m',
 'extracted_attributes': {'size': 'm',
  'color': 'blue',
  'season': None,
  'year': None,
  'brand': 'essential',
  'material': None,
  'category': 'tops',
  'subcategory': 'polo',
  'features': None},
 'primary_match': {'SKU': 'SKU1000206',
  'Product_Name': 'essential polo',
  'Category': 'tops',
  'Subcategory': 'polo',
  'Brand': 'essential',
  'Color': 'blue',
  'Size': 'm',
  'Material': 'cotton',
  'Features': 'breathable',
  'Season': 'summer',
  'Price': 103.27,
  'Year': '2024',
  'Confidence': '77.36%'},
```

The color is extracted correctly although it is typed as 'blu' and also the brand has a typo at 'esential', but this is also matched correctly!

The matched item is correct, with 77.36% confidence.

Wrong predictions analysis

```
Number of incorrect primary predictions: 2
                                              description  true_label  \
3               want gray option in dresses open to suggestions  SKU1000594
24  do you stock essentials blue sundres in that shade i need m  SKU1000563

   primary_pred                           top3_preds
3    SKU1000537  [SKU1000594, SKU1000618, SKU1000504]
24   SKU1000233  [SKU1000287, SKU1000257, SKU1000161]
```

There are two mistakes in total. For the first mistake, the first alternative is the correct answer, so it's close enough. For the second mistake, the correct item is not retrieved.

For the first mistake, as it can be seen from the next picture there is no actual “correct” option since the description is just for a gray dress. The match and the first alternative are indeed gray dresses.


```
{'description_index': 3,
 'description': 'want gray option in dresses open to suggestions',
 'extracted_attributes': {'size': None,
  'color': 'gray',
  'season': None,
  'year': None,
  'brand': None,
  'material': None,
  'category': 'dresses',
  'subcategory': 'dress',
  'features': None},
 'primary_match': {'SKU': 'SKU1000537',
  'Product_Name': 'classic dress',
  'Category': 'dresses',
  'Subcategory': 'dress',
  'Brand': 'classic',
  'Color': 'gray',
  'Size': 'm',
  'Material': 'linen',
  'Features': 'moisture-wicking',
  'Season': 'winter',
  'Price': 67.73,
  'Year': '2024',
  'Confidence': '57.48%'},
 'top_3_alternatives': [{'SKU': 'SKU1000594',
   'Product_Name': 'classic dress',
   'Category': 'dresses',
   'Subcategory': 'dress',
   'Brand': 'classic',
   'Color': 'gray',
   'Size': 'xs',
   'Material': 'polyester',
   'Features': 'stretch',
   'Season': 'summer',
   'Price': 130.23,
   'Year': '2025',
   'Confidence': '57.03%'},
```


For the second mistake however, the algorithm fails for the sundress and extracts blouse. Yet, it captures the brand ‘Essentials’, the color ‘Blue’ and the size ‘M’.


```
{'description_index': 24,
 'description': 'do you stock essentials blue sundres in that shade i need m',
 'extracted_attributes': {'size': 'm',
  'color': 'blue',
  'season': None,
  'year': None,
  'brand': 'essential',
  'material': None,
  'category': 'tops',
  'subcategory': 'blouse',
  'features': None},
```


There are two .json files in the project files, containing all the results for the original and synthetic data.

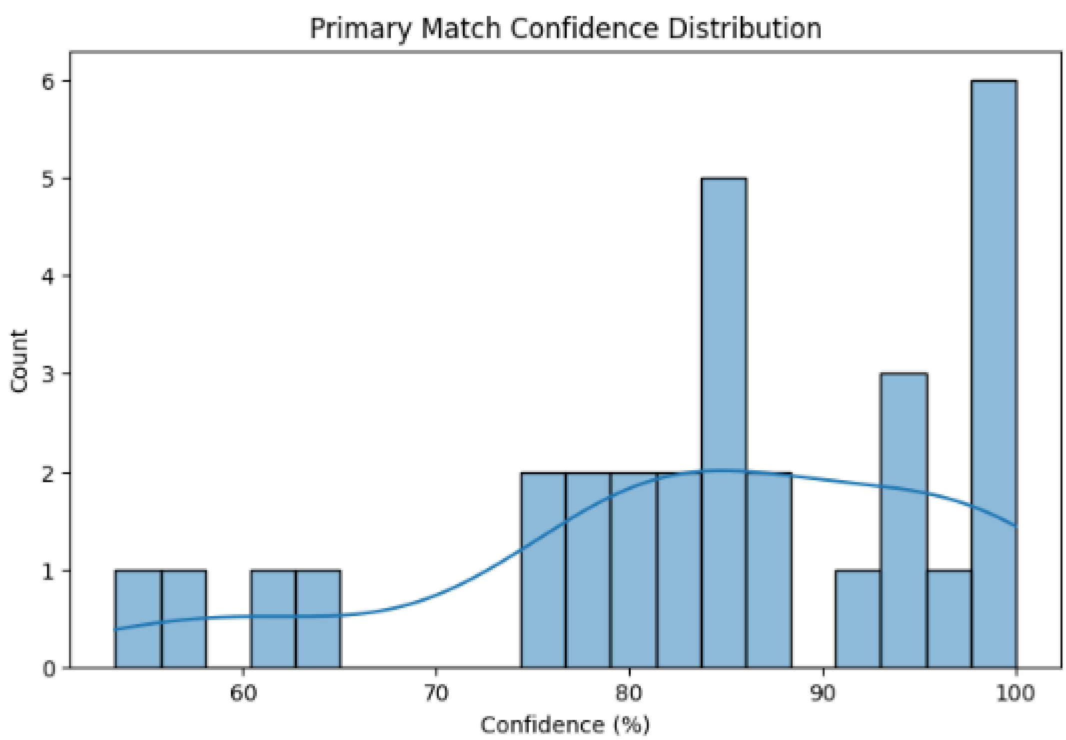
- results_from_test_descriptions
- results_from_train_descriptions

Business Impact

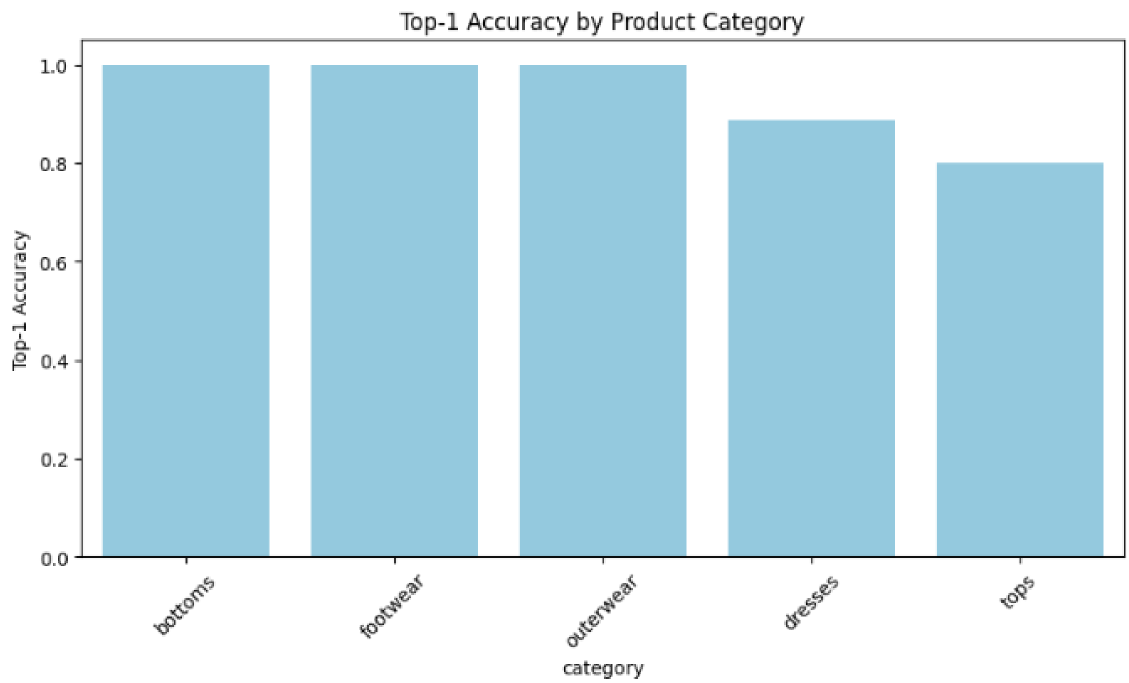
With an automated match accuracy of 93.33% on top-1 matches and 96.66% in top-3, the system eliminates all manual matches per day and instead the script can be run automatically and checked only for mistakes, saving 2–3 hours of operational time. This reduces order-processing delays, minimizes stockout miscommunication, and recovers lost revenue by surfacing similar in-stock alternatives that previously went unnoticed.

Visualizations

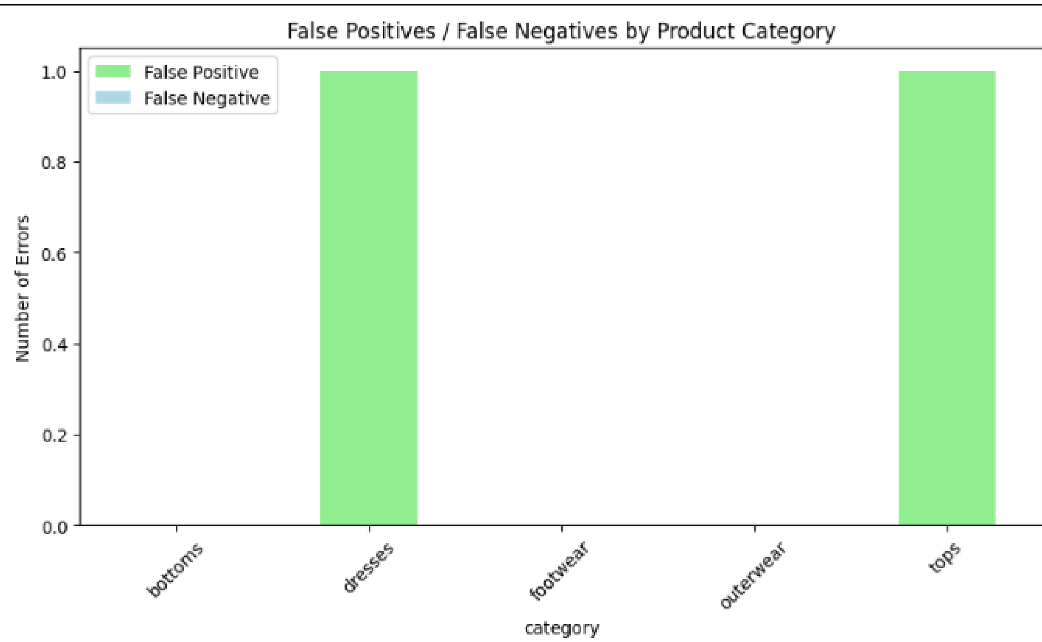
Below we can see that the average confidence level for the top-1 matches is around 85%. Yet, even the lower confidence level matches of 50%-60% can be trusted since we hit a 93.33% accuracy overall.



As we can also see, the only categories there is not an 100% accuracy are the dresses and tops.



Additionally, there is also no false negatives, while false positives account for 100% of the two mistakes made by the matching algorithm in the dresses and tops categories respectively.



Business Integration Plan

1. Integration with BI dashboards

- The matching system outputs structured CSV or JSON files with:
    - Extracted features
    - Match results and confidence scores
- These outputs can be **automatically ingested** into BI dashboards.
- Dashboards can visualize:
    - Match success rates
    - Confidence distributions
    - Top unmatched products
- Enables **real-time or scheduled reporting** on matching performance.

2. Recommended KPIs

- **Match rate:** % of queries automatically matched correctly.
- **Confidence trends:** average confidence per product category, track improvement over time.
- **Manual override rate:** % of automated matches that were corrected by human reviewers.

3. Data quality requirements

- Product catalog features should be **complete**.
    - Standardize features and not fill them using '|' as a discriminator
    - Avoid adding missing values
- Descriptions should be **rich enough** for semantic matching, having the most features possible.

4. Model retraining and update strategy

- Retrain periodically to capture:

  1) New products added to the catalog

  2) Changes in naming conventions

  3) Evolving synonyms and attributes

- frequency depends on the number of catalogue updates.

Operational Recommendations

1. Confidence threshold for auto-matching

The algorithm works very well even in the lower confidence levels I propose that any value below 60% confidence needs manual check just for redundancy.

2. Alert system for low-confidence matches

Alerts can be generated for an unusual high number of low-confidence matches or repeated false predictions in specific categories/subcategories or brands.

3. Catalog improvement suggestions

What definitely needs to be addressed is the many missing values. For these descriptions types I think there is no current need for changes. Given the changing nature of the descriptions, more features may need to be added to the products.

4. Scalability considerations

Before deployment, the algorithm needs to be tested against real time predictions and higher to address speed/efficiency and robustness. Additionally, the use of GPU servers to run the code and regular re-trainings can help the algorithm succeed.

Files provided

- **extracted_descriptions.csv**, features from the original descriptions.
- **results_from_train_descriptions.json**, json representation of the original data for features extracted, the matched item and the top-3 alternatives along with confidence levels.
- **test_descriptions.csv**, synthetic descriptions following common representations of the original data used for testing and validating.
- **results_from_test_descriptions.json**, json representation of the synthetic descriptions used for testing with features extracted, the matched item and the top-3 alternatives along with confidence levels.
- **Christogeorgos_solution.ipynb,** jupyter file with the python code for this project.
- **Readme file**
- **PowerPoint Presentation**