Final Group Project - USC DSO560/Behold Spring 2021

Due Tuesday, May 11th, at 11:59pm PST

OVERVIEW

Behold would like to partner with USC's DSO560 NLP class to **develop a recommendation system that is based on the retail fashion brand's brand descriptive information.**

AVAILABLE DATASET FILES

1. **Behold Brands** (*https://dso-560-nlp-text-analytics.s3.amazonaws.com/behold_brands.csv*): Contains a listing of all brands sold and managed by Behold, including natural language descriptions of the brands themselves.
2. **Behold Products** (*https://dso-560-nlp-text-analytics.s3.amazonaws.com/Behold+product+data+04262021.xlsx*): Contains a 50,000+ row dataset of all products sold or cataloged by Behold, past and present.
3. **Outfits and Items** (*https://dso-560-nlp-text-analytics.s3.amazonaws.com/outfit_combinations.csv*): Contains custom curated outfits created by Behold's human domain experts (clothing/outfit curators)
4. **Additional Tags** (*https://dso-560-nlp-text-analytics.s3.amazonaws.com/usc_additional_tags.csv*): Contains tags for each product chosen by Behold's human domain experts (clothing/outfit curators).

DELIVERABLES

1. A common problem for Behold is the extremely manual classification of new products into their specific brands. Behold has supplier relationships from many global clothing vendors and marketplaces, but has found that the bottleneck for scaling out the number of products they can sell is identifying the brands associated with each new product. Use the brands' biographical description (**behold_brands.csv**), product descriptions (**products.csv**) and any externally collected metadata, ***build an NLP classification model to predict which brand a new product should be assigned.***

You are given brands for each product in the **products.csv** file, along with a variety of metadata fields, including:

- **Name**: the name of the product
- **Details**: details about the product that the supplier has elected to provide

- **Description:** generally a natural language text field that contains a description of the product, materials used in its construction, and recommended pairings

You can assume your features include all columns except for the **brand** column. To build and evaluate your model, you should create a holdout set from the set of products of between 10-20% of all products. Your model must predict successfully from the multiple brands available. **Note that this is a more difficult problem than simply binary classification, because we are predicting one of many different possible brands.**

2. **Create a brand recommender algorithm that would recommend an outfit given a customer's search query.** This algorithm can be simply a Python function:

```
def search(user_query: str)-> Dict:
   """
   user_query is a string that is passed in by the user, and this function
returns a dictionary of outfit results. Example:

   search("pleated casual skirt") -> {
    "top": "...",
    "bottom": "...",
    "shoe": "..."
   }

   """
```

Examples of queries and a sample outfit result:

QUERY: slim fitting, straight leg pant with a center back zipper and slightly cropped leg
OUTPUT (Recommended Outfit Combination): • bottom: **Marlon Pant** (01DPKMH0D252JKMAA27MFCT5GM) • shoe: **Giulia Satin Heel** (01DPNHQDG6GPTKV97CFQRJDHE) • accessory: **Cassi Belt Bag** (01DPEHS0XH9PDD1GH5ZE4P43A2)

Note: If you match a product that already exists in the outfit_combinations.csv file, you can simply return the rest of its outfit recommendations from that same table, since this is a curated set of outfits picked by experts.

GENERAL GUIDELINES

- If you're stuck, message us in the private Slack channel. This is a "messy", real-life project, so feel free to ask as many questions as you'd like.
- Look closely at the pattern of text in the **description** and **details** fields. It is a challenging mixture of both natural language text descriptions but also some tags, such

as the phrase "100% wool". These tags should be treated separately from the rest of the text, since they aren't a logical part of the sequential tokens that constitutes the rest of the product description. Should phrases like "100% wool" be extracted out into their own feature?

- Divide up the tasks for this project. There is a reason why there are multiple team members. Some team members can build the recommendation algorithm while others build the classification model. Others can focus on writing the markdown/comments to explain the model. Others can look through the dataset and identify patterns in the text descriptions and details that you want to extract via regex.
- At some point, you'll need to find the similarity/distance between different products. Think about what features you can use to measure distance/similarity, both text-based and generic categorical features (ie. Location made in)
- Add comments and markdown explanations for code cells. The goal is to be able to have someone from Behold open your group's notebook, understand the logic your team is using to make brand classifications and recommendations for outfits, and from there implement it into production.
- When thinking about combinations of products to generate an outfit recommendation, look through the product description and see if you can identify **any phrases or keywords that indicate a logical pairing of one product with another**. For example, consider the following part of a product description

*the collared top **allows for** an optional opening with a figure-flattering elasticated waistband and a beautifully tiered skirt.*

Here, clearly the phrase "**allows for**" signals that whatever comes after is a strong candidate to be paired with this product in question. These phrases should be modelled and accounted for in your recommendation system.

FREQUENTLY ASKED QUESTIONS

What is the difference between the description and the brand_description fields?

brand_description is the original description from the brands. Sometimes there are parts of the brand_descriptions that we would clean up (such as, final sale, found only on xyz site, etc), the post clean up description is in description.

What is the difference between the difference the brand and the brand_name fields?

The brand_name is the name of the product that the brands use to call their items. For example, the brand Rails has many styles of button down shirts. One style is called Hunter,

another style is called Ellis and then there's Ingrid (different fabric from both Hunter and Ellis), etc.

SUBMISSION INSTRUCTIONS

Please create:

- **A team Github repository** with your group's code and notebooks used.
- **Invite as collaborators the following individuals:**
    - Yu Chen (ychennay)
    - Rajat Gaur (Rajat-Gaur)
    - Karin Chu

Please also submit **via Direct Message in Slack to me (Yu Chen) the 360 feedback evaluations.**

Scoring Rubric	**Points Available**			
TOTAL: 20pts	**1pt**	**2pts**	**3pts**	**4pts**
The classification model passes tests for the correct product brand when provided sample product inputs.	Model does not work and instructor/TA encounter numerous errors when attempting to run it	Model's classification performance is below baseline – just guess the mode would yield better performance.	The model's classification performance is higher than baseline but below 60% overall accuracy.	6 or more of the 10 tests return an appropriate brand match
Classification model shows evidence of incorporating domain context (women's retail) – either via SME rules, or via embeddings. **For example, new features are engineered to capture product attributes, like occasion, or manufacturing location (Made in USA / Europe /etc.)**	No evidence of anything beyond a full, exact string match	Basic cleaning/preprocessing evident, but the model itself relies entirely on simple regex/string match	Some basic business domain logic is encoded either via a word embedding scheme, or via preprocessing rules (ie. groupings)	Business domain logic is encoded either via a word embedding scheme, or via preprocessing rules (ie. groupings)

Recommendation user to outfit query model shows evidence of incorporating domain context. **For example, a product in the pumps category will likely co-occur with open-toed / close-toed, pointed-toe.** **For example, human domain experts' outfit groupings are used and considered in creating the outfit recommendations.**	No evidence of anything beyond a full, exact string match	Basic cleaning/preprocessing evident, but the model itself relies entirely on simple regex/string match	Some basic business domain logic is encoded either via a word embedding scheme, or via preprocessing rules (ie. Groupings, collocations)	Business domain logic is encoded either via a word embedding scheme, or via advanced preprocessing rules (ie. Groupings, collocations)
Code is documented with thought process easily visible to instructor/client to review	No documentation or comments	Documentation and comments are haphazard or unclear	Functions are documented with docstrings, comments and markdown explain the purpose of each cell	Functions are documented with docstrings, comments and markdown explain the purpose of each cell. Assumptions and discussions are captured in markdown and the client judges the notebook logic to be easy to understand and follow.
Code is published to a Github repository with team members added as collaborators	**No (0 pt)**		**Yes (2 pt)**	

All group members have submitted 360 feedback reviews by deadline and completed in detail, specifying specific contributions.	**No (0 pt)**	**Yes (2 pt)**
Group members' 360 feedback reviews show no major divergences in perceived effort	Grades may be adjusted either upwards or downwards for individual team members.	