Project Proposal: Analysis of Chemical Components

Created by: SHRIKRUSHNA KARALE

Cover Letter

Hello MTE,

Thank you for the opportunity to work on the project *Analysis of Chemical Components*. This project aims to simplify the process of selecting cosmetic products, especially for individuals with sensitive skin, by leveraging data science techniques to analyse the chemical ingredients in cosmetics. Below is a detailed proposal outlining the project's objectives and tasks.

Project Description

Buying new cosmetic products can be overwhelming, particularly for individuals prone to skin issues. The ingredient lists on the back of cosmetic products often contain unfamiliar chemical terms, making it challenging for consumers without a chemistry background to make informed choices.

This project seeks to build a **content-based recommendation system** using **data science**, with a focus on the chemical components of cosmetics. We will process the ingredient lists of 1472 products from **Sephora** using **word embedding** techniques and apply **t-SNE** for visualization. The goal is to group similar products based on their chemical components and provide interactive visualizations using the **Bokeh** library.

 # Task 1: Import and Inspect the Dataset

import **pandas** as **pd**

import **numpy** as **np**

from **sklearn**.**manifold** import **TSNE**

 # Load the dataset

df = **pd**.**read_csv**('datasets/cosmetics.csv')

 # Display a random sample of five rows

display(df.**sample**(5))

 # Display the counts of product types

```
display(df['Label'].value_counts())

# Task 2: Filter the Data for Moisturizers and Dry Skin
# Filter for moisturizers
moisturizers = df[df['Label'] == 'Moisturizer']

# Further filter for dry skin
moisturizers_dry = moisturizers[moisturizers['Dry'] == 1]

# Reset the index
moisturizers_dry = moisturizers_dry.reset_index(drop=True)

# Task 3: Tokenize the Ingredients
# Initialize variables
corpus = []
ingredient_idx = {}
idx = 0

# Tokenize ingredients
for i, row in moisturizers_dry.iterrows():
    ingredients = row['Ingredients'].lower().split(', ')
    corpus.append(ingredients)

    for ingredient in ingredients:
        if ingredient not in ingredient_idx:
            ingredient_idx[ingredient] = idx
            idx += 1

# Task 4: Initialize a Document-Term Matrix
M = len(moisturizers_dry)  # Total number of products
N = len(ingredient_idx)    # Total number of ingredients
```

```
A = np.zeros((M, N))      # Initialize MxN matrix with zeros

# Task 5: Create a One-Hot Encoder Function
def oh_encoder(ingredients, ingredient_idx):
    x = np.zeros(len(ingredient_idx))
    for ingredient in ingredients:
        if ingredient in ingredient_idx:
            idx = ingredient_idx[ingredient]
            x[idx] = 1
    return x

# Task 6: Populate the Document-Term Matrix
for i, tokens in enumerate(corpus):
    A[i] = oh_encoder(tokens, ingredient_idx)

# Task 7: Dimension Reduction with t-SNE
# Create a TSNE instance
model = TSNE(n_components=2, learning_rate=200, random_state=42)

# Apply t-SNE to the document-term matrix
tsne_features = model.fit_transform(A)

# Assign t-SNE features to the DataFrame
moisturizers_dry['X'] = tsne_features[:, 0]
moisturizers_dry['Y'] = tsne_features[:, 1]

# Display the resulting DataFrame with t-SNE coordinates
display(moisturizers_dry[['X', 'Y', 'Product_Name', 'Ingredients']])
from bokeh.plotting import figure, show
from bokeh.models import ColumnDataSource, HoverTool
from bokeh.io import output_notebook
```

```
from bokeh.layouts import row, widgetbox
from bokeh.models.widgets import Select

# Ensure the output is displayed within the notebook
output_notebook()

# Task 8: Mapping Ingredients with Bokeh
# Prepare the data for plotting
source = ColumnDataSource(data={
    'x': moisturizers_dry['X'],
    'y': moisturizers_dry['Y'],
    'product': moisturizers_dry['Product_Name'],
    'ingredients': moisturizers_dry['Ingredients']
})

# Create a scatter plot
plot = figure(title="t-SNE Visualization of Moisturizers",
              x_axis_label="t-SNE Feature 1",
              y_axis_label="t-SNE Feature 2",
              plot_width=800,
              plot_height=600,
              tools="pan,wheel_zoom,reset")

# Add points to the plot
plot.circle('x', 'y', size=8, source=source, fill_alpha=0.6, color="navy")

# Task 9: Adding a Hover Tool
# Create the hover tool to display product names and ingredients
hover = HoverTool()
hover.tooltips = [
    ("Product", "@product"),
```

```
    ("Ingredients", "@ingredients")
]
plot.add_tools(hover)

# Task 10: Mapping the Cosmetic Items
# Display the plot
show(plot)

# Task 11: Comparing Two Products
# Create dropdown widgets to select two products for comparison
product_list = moisturizers_dry['Product_Name'].tolist()

select1 = Select(title="Select First Product", value=product_list[0], options=product_list)
select2 = Select(title="Select Second Product", value=product_list[1], options=product_list)

# Callback function to display product comparison
def compare_products(attr, old, new):
    product1 = select1.value
    product2 = select2.value

    # Get the ingredients of the selected products
    ingredients1 = moisturizers_dry[moisturizers_dry['Product_Name'] == product1]['Ingredients'].values[0]
    ingredients2 = moisturizers_dry[moisturizers_dry['Product_Name'] == product2]['Ingredients'].values[0]

    print(f"Comparison of {product1} and {product2}:")
    print(f"Ingredients of {product1}: {ingredients1}")
    print(f"Ingredients of {product2}: {ingredients2}")

# Add event listeners to the dropdowns
select1.on_change('value', compare_products)
```

```
select2.on_change('value', compare_products)


# Display dropdown widgets for product comparison
layout = row(widgetbox(select1), widgetbox(select2))
show(layout)
```

Explanation of Code:

- **Task 1**: We import the required libraries, load the dataset, and display samples along with the product type counts.
- **Task 2**: We filter the dataset for "Moisturizer" and products suitable for dry skin, then reset the index.
- **Task 3**: Ingredients are tokenized (converted to lowercase and split by a comma). A list of all ingredients (corpus) is created, and each ingredient is indexed in ingredient_idx.
- **Task 4**: We initialize a document-term matrix A of zeros, with dimensions based on the number of products and ingredients.
- **Task 5**: The oh_encoder function converts ingredient lists into binary (one-hot encoded) arrays.
- **Task 6**: We populate the document-term matrix with these one-hot encoded ingredient vectors.
- **Task 7**: We apply t-SNE to reduce the dimensions of the matrix to two for visualization, then add these new t-SNE features (X, Y coordinates) to the Data Frame.
- **Task 8**: We use Bokeh's Column Data Source to map the t-SNE features (X and Y coordinates) to the product names and ingredient lists. The scatter plot visualizes each product based on its chemical composition.
-
- **Task 9**: A hover tool is added, allowing users to see product details (name and ingredients) when hovering over each point in the plot.
-
- **Task 10**: The plot is finalized, showing all cosmetic items mapped based on their ingredient similarities.
-
- **Task 11**: Dropdowns (Select widgets) allow users to compare two products by selecting them from the list. The ingredients of the selected products are printed for comparison.

This project will provide an insightful way to explore the chemical composition of cosmetic products, allowing users to make more informed choices when purchasing new items. By leveraging data science techniques and interactive visualizations, the system will help consumers find products that best suit their skin type and personal preferences.