WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Smart Text Classification and Categorization using Gen AI

Sandnya Dalvie, Sameer Siddiqui, Neelima Ratra

Industry Personnel, Accenture

ABSTRACT:

Simple activities on the internet like making a tweet, writing a product review, commenting your opinion on certain article, the emails you write, etc. generates text data. However, this data is unstructured as it is not organized in a pre-defined manner. Irrespective of its structure, this data does hold good amount of information that can be beneficial to business.

There are many Natural Language Processing techniques in the market today that handle such unstructured text data to get useful information from it using unsupervised learning algorithms. For example, a product owner can extract the reviews on his product to understand the sentiments or popularity of his product. But what if he wants to run this process in real-time with continuous feedback from the users that analyzes the product popularity/grievances and further helps him make product improvements. He should also be able to categorize and group the similar reviews received efficiently and quickly to get a gist of the market behavior without having to run long text analytics processes again and again.

The paper aims at establishing a process to make a perfect blend of unsupervised and supervised learning algorithms. It uses onetime topic modelling to efficiently group the textual unstructured data based on the input text similarity, assigns each group relevant labels using smart keyword extraction to quickly interpret the content of reviews to categorize it, brings it in a structured form like every text with a category label and builds a supervised Naïve Bayes Classification Model on it. The model in turn helps to directly label/categorize new text that comes in based on the previous data rather than having to re-analyze it again using Topic Modelling or Keyword Extraction. This will help in smart, real-time, quick text analysis and grouping.

Problem Statement:

Text data today is generated with various user activities on the internet. It holds humongous amount of important information, but its unstructured format sometimes becomes a restriction to handle and extract the exact required information from it efficiently. For example, a retail store has a list of products along with their descriptions and is interested in categorizing the same for administrative purpose. Rather than manually studying the products and categorizing them, it would be smart if an unsupervised learning algorithm studies the product descriptions, groups similar products based on this and then assigns label to each group of products like Diary, Shampoo, Biscuits, etc. Also, if there are any new products launched they should be automatically classified to one of the pre-defined labelled groups to avoid the process of labelling again.

Traditionally the grouping of similar products based on their descriptions can be done using the Topic Modelling Algorithm, however the conventional algorithm just helps you group the similar text together but does not give each group/topic an appropriate text label for a layman to understand the content within the grouped data. Also, once you label these text as per the groupings, in case there is a new text with content like the prior texts you might have to re-run the entire grouping process again to add a label to it or categorize it. This in turn leads to a repetitive process exhausting a lot of time and resources.

Proposed Solution:

The solution proposed solves the defined problem in two-stages:

Stage 1: This is a one-time process. With the help of the existing/historical text data at hand, we group similar texts together based on their content. Study each group to label it smartly with a term related to the exact context of the grouped text and categorize the text as per the label in an automated process.

Stage 2: Once each text has a label/category naming from **Stage 1** using this data as an input training data we are building a supervised learning Classification model. So, in future if any other similar text comes in we can directly classify it using the model built and we do not have to repeat the **Stage 1** process again.

Uniqueness of the Solution:

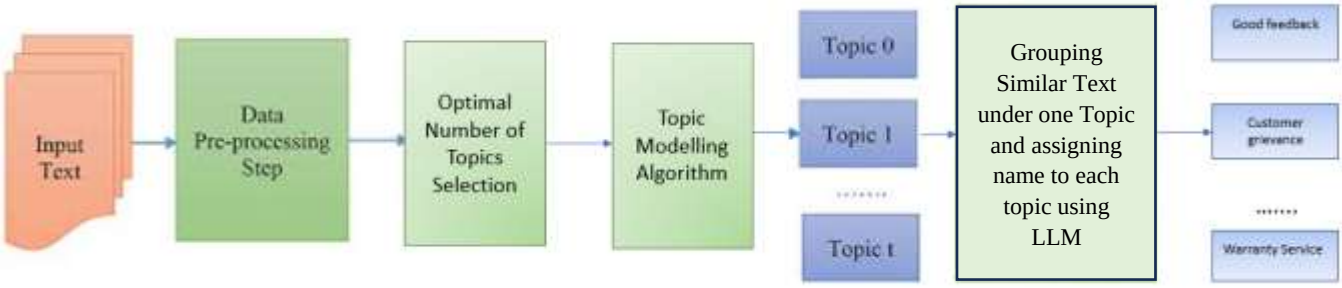
The uniqueness of the solution lies in the fact that we have tried establishing an algorithmic process to categorize and group various text documents with relevant names/labels that is not taken care of by conventional Topic Modelling Algorithm. Further it also builds a predictive model to classify future text documents automatically with labels without repeating the Stage 1 process again. As defined, this is a two-stage process.

Following is the unique stage-wise work flow we have designed for this process:

Stage 1 :

This is a one time process where we try to:

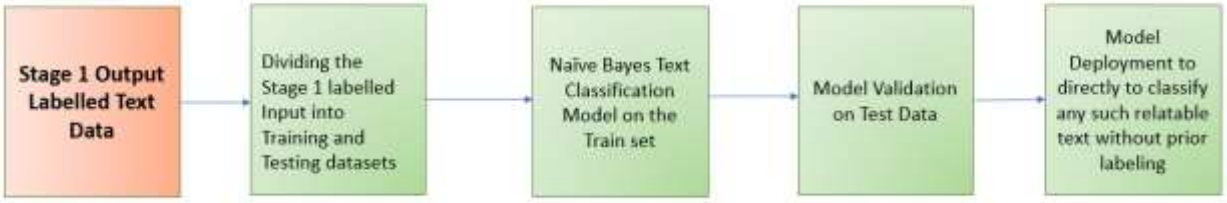
- **Group similar reviews/documents/emails together**
- **Give each review group an appropriate category naming**



Stage 2:

In this stage we try to:

- **Use the part of classified data from Stage 1 as the training data and build a predictive model to classify any such further relevant data into the identified review categories**



Validation and Experiments:

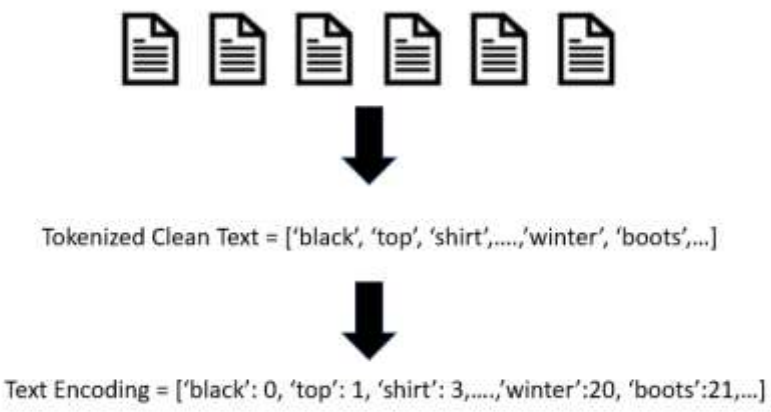
To validate the Proposed solution the following Steps for each Stage were followed in this process:

STAGE 1

STEP 1

Data Pre-processing and Cleaning:

The text data in its raw format needs to be cleaned in terms of special characters, unnecessary numbers, stop words that do not add any insightful meaning to the text. Lemmatizing and Steaming of the data also help to keep only apt text. This step is essential as it helps to keep only the clean, relevant data that can be taken ahead for a meaningful analysis We also tokenize the text, where we break a string or document into smaller chunks of words. Each text token is encoded with a unique ID/code. Encoding each token, the corpus now simply contains each ID and its frequency which helps to reduce the tool memory while analyzing large amount of text. Python libraries like **spaCy** and **NLTK** can be used to prepare this data for analysis.



STEP 2

Selecting the optimal number of Topics:

This will be done using the coherence score. With the **LdaMulticore** and **CoherenceModel** functions in Python we calculate a coherence score using the **C_v algorithm** by iteratively increasing the number of topics one by one. The point that has the max coherence is the optimal number of topics for running Topic Modelling.

STEP 3

Topic Model Building:

Once the optimal number of topics is decided the next step is to train the unsupervised machine learning topic model on the data. We again start with the **LdaMulticore** function in Python but this time we use the exact optimal number of topics and apply this to the corpus that has been a result of Step 1. After applying the model, we can then study how many documents/texts have been classified under a single topic. Every document/text will now have a topic number assigned to it. Similar documents will have similar topic number assigned.

STEP 4:

Naming Each Topic Number:

Large Language Models like GPT 3.5 Turbo can be used using prompt engineering to automatically generate names for each topic. Also, as naming the topic using historical data is a one-time process we need not worry about the output variability each time we run the model.

Define a set of prompts that instruct GPT-3.5 Turbo to generate names or summaries for each topic and ask the model to generate a name or summary for the topic. Be explicit in your instructions and consider providing some context about the desired output.

Make sure to do initial testing of your prompt and Refine your prompts based on the initial outputs if the generated names are not satisfactory.

STEP 5:

Divide Data Into Train and Test sets:

With STEP 4 the input data now has proper labeling. We will now use this data as our input to the classification model for which we will divide it into Train and Test datasets with a ratio of 60:40.

STEP 6:

Classification Model Building:

The Train data created in STEP 5 is now used to build a Naïve Bayes Text Classification Model. The Naive Bayes text classification algorithm is a type of probabilistic model that uses Bayes' theorem with the assumption of "naive" independence between the variables (features), making it effective for small datasets.

$$P(y \mid X) = \frac{P(X \mid y)P(y)}{P(X)}$$

where, **y** is **class variable** and **X** is a **dependent feature vector with dimension d the number of variables/features of the sample.**

STEP 7:

Model Validation and Deployment:

The model created in the above STEP 6 will now be validated with the help of AUC. Any value of the **AUC > 0.80** , the model is a good fit. Post validation the model can be deployed to predict the labels of unknown texts now directly without having to run the STEPS till Topic modelling again.

STEP 8: (Optional)

Customer Feedback Loop

Once the model is productionized timely customer feedbacks about the model inaccuracies (eg: incorrect labelling of product category for a given product description would also be helpful to improve and refresh the model if needed.

Results and Analysis

Below is a working example for the above explained process in Python. The problem statement defined is the business wants to categorize and group product descriptions as per their content. We have used 7 unlabeled product descriptions for the same, of which first two are related to Shampoo , next three to Chocolates and last two to Juices.

Topic Modelling

After data cleansing and pre-processing we are using the LdaMulticore model for topic modelling.

Code Snippet

```
from gensim import corpora
from gensim.models import LdaMulticore
from pprint import pprint

# Sample data
product_descriptions = [
    "Immerse yourself our Refreshing Shampoo. Specially formulated to cleanse and revitalize your hair, le
    "Indulge in Silky Smooth Shampoo. Imbued with the natural goodness of coconut, this shampoo transforms
    "Dive into the world of indulgence with our Decadent Dark Chocolate Delight. Crafted from the finest b
    "Experience pure bliss with our Irresistible Milk Chocolate. Silky-smooth and mouthwateringly sweet, t
    "Immerse yourself in the symphony of flavors with our Hazelnut Praline Chocolate. Each piece is a mast
    "Quench your thirst with the Tropical Paradise Mango Juice. Bursting with the exotic flavor of sun-ripe
    "Awaken your senses with the Zesty Citrus Burst Orange Juice. Packed with the tangy freshness of handp
]

# Tokenize the descriptions
tokenized_descriptions = [description.lower().split() for description in product_descriptions]

# Create a dictionary representation of the documents
dictionary = corpora.Dictionary(tokenized_descriptions)

# Create a bag-of-words representation of the documents
corpus = [dictionary.doc2bow(tokens) for tokens in tokenized_descriptions]

# Train the LDA model
lda_model = LdaMulticore(corpus, num_topics=3, id2word=dictionary, passes=10, workers=2)

# Print the topics and associated words
pprint(lda_model.print_topics())

# Assign topics to each document
topics = lda_model.get_document_topics(corpus)
```

Output Snippet

```
Document 1: [(0, 0.96455413), (1, 0.017373653), (2, 0.018072188)]
Document 2: [(0, 0.9698017), (1, 0.014854417), (2, 0.015343874)]
Document 3: [(2, 0.984544)]
Document 4: [(2, 0.9813187)]
Document 5: [(2, 0.9829713)]
Document 6: [(0, 0.012718555), (1, 0.013948754), (2, 0.9733327)]
Document 7: [(0, 0.01176352), (1, 0.012668971), (2, 0.9755675)]
```

The model correctly groups the descriptions where descriptions 1 and 2 fall under Topic 0, descriptions 3,4,5 under Topic 1 and descriptions 5 and 7 under Topic 2

Naming Each Topic

Once each description was tagged under each topic next step is to assign a name to these topic based on the content in it. We have used OpenAI GPT-Turbo 3.5 LLM for the same. You must use Your OpenAI Key for the same.

Code Snippet

```
import openai

# Set your OpenAI GPT-3.5 Turbo API key
api_key = "YOUR_API_KEY"
openai.api_key = api_key

# Sample data: Replace this with your actual data
product_data = [
    {"description": "Immerse yourself in the invigorating scent of our Refreshing Citrus Burst
    {"description": "Indulge in the luxury of Silky Smooth Coconut Infusion Shampoo. Imbued wi
    {"description": "Dive into the world of indulgence with our Decadent Dark Chocolate Deligh
    {"description": "Experience pure bliss with our Irresistible Milk Chocolate. Silky-smooth
    {"description": "Immerse yourself in the symphony of flavors with our Hazelnut Praline Cho
    {"description": "Quench your thirst with the Tropical Paradise Mango Juice. Bursting with
    {"description": "Awaken your senses with the Zesty Citrus Burst Orange Juice. Packed with
]

# Initialize a dictionary to store product descriptions under each topic
topics_dict = {f"Topic {i}": [] for i in range(3)}
```

```
# Group product descriptions by topic
for entry in product_data:
    topics_dict[entry["topic"]].append(entry["description"])

# Generate topic header names using GPT-3.5 Turbo
generated_headers = {}
for topic, descriptions in topics_dict.items():
    prompt = f"Generate a topic header for the following product descriptions:\n{', '.join(descriptions)}\nHeader:"
    response = openai.Completion.create(
        engine="text-davinci-003",  # You can use "text-davinci-003" for GPT-3.5 Turbo
        prompt=prompt,
        max_tokens=50,  # Adjust the max_tokens parameter as needed
        n=1,  # Number of completions to generate
    )
    header = response["choices"][0]["text"].strip()
    generated_headers[topic] = header

# Print generated headers
for topic, header in generated_headers.items():
    print(f"Topic: {topic}\nGenerated Header: {header}\n")

# Print product descriptions under each topic
for topic, descriptions in topics_dict.items():
    print(f"Topic: {topic}\nProduct Descriptions:\n{', '.join(descriptions)}\n")
```

Output Snippet

```
Topic: Topic 0
Generated Header: Shampoo Collection
Product Descriptions:
Immerse yourself in the invigorating scent of our Refreshing Citrus Burst Shampoo. Specially formulated to cleanse and revitali
Indulge in the luxury of Silky Smooth Coconut Infusion Shampoo. Imbued with the natural goodness of coconut, this shampoo trans

Topic: Topic 1
Generated Header: Chocolate
Product Descriptions:
Dive into the world of indulgence with our Decadent Dark Chocolate Delight. Crafted from the finest cocoa beans, each bite unve
Experience pure bliss with our Irresistible Milk Chocolate. Silky-smooth and mouthwateringly sweet, this chocolate bar is a sym
Immerse yourself in the symphony of flavors with our Hazelnut Praline Chocolate. Each piece is a masterpiece of creamy hazelnut

Topic: Topic 2
Generated Header: Tropical Juice
Product Descriptions:
Quench your thirst with the Tropical Paradise Mango Juice. Bursting with the exotic flavor of sun-ripened mangoes, this juice i
Awaken your senses with the Zesty Citrus Burst Orange Juice. Packed with the tangy freshness of handpicked oranges, this juice
```

Appropriate and relevant topic names have been assigned for each Topic.

Train Naïve Bayes Text Classification Model

Considering the above tags as labels assigned to each product description for the training data we will build a text classification model using Multinomial Naïve Bayes Model.

Code Snippet

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, roc_auc_score

# Separate data into features (X) and labels (y)
X = [item[0] for item in data]
y = [item[1] for item in data]

# Create a CountVectorizer to convert text into a matrix of token counts
vectorizer = CountVectorizer()
X_vectorized = vectorizer.fit_transform(X)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_vectorized, y, test_size=0.2, random_state=42)

# Build and train a Naive Bayes classifier
nb_classifier = MultinomialNB()
nb_classifier.fit(X_train, y_train)
```

Test Naïve Bayes Model

A test description “Delight in the harmonious blend of rich, creamy milk chocolate that defines the essence of Cadbury indulgence. Each square is a testament to the brand's legacy, enveloping your taste buds in a velvety embrace of sweetness.” is used to check the model classification and performance using AUC.

Code Snippet

```
# Predict the topic for the test description
predicted_topic = nb_classifier.predict(test_description_vectorized)[0]

# Print the predicted topic
print("Predicted Topic:", predicted_topic)

# Evaluate the model's performance
y_pred = nb_classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, nb_classifier.predict_proba(X_test), multi_class='ovr')

# Print model evaluation metrics
print("Model AUC:", roc_auc)
```

Output Snippet

```
Predicted Topic: Chocolate
Model AUC : 0.803
```

The model correctly predicts the product category with an AUC of **0.803**

Future Directions

The current scope of this utility is to just handle any data that comes in an unstructured text format. But this capability can be further expanded where any text that is a part of an image or video can be extracted using OCR and then the usual defined process in the paper can be applied.

There is plenty of scope for process automation as well to avoid any manual efforts or repetitive intervention.

Conclusion

We have been able to establish a step wise analytical process that will help any business better understand their customer needs with their reviews to effectively plan their marketing strategies. Or even where they want to better organize and group their similar products with tag for better administrative purpose. Or in general any list text documents that needs to be grouped as per their similarities to ease the process of handling huge text data.

This will not only be a huge saving on the manual efforts of reviewing each document before categorization but also save on time and other resources which will save huge costs of any business that deals with a lot of unstructured text data.

References

https://monkeylearn.com/text-classification/#:~:text=Automatic%20text%20classification%20applies%20machine,effective%2C%20and%20more%20accurate%20manner.

https://arxiv.org/abs/2310.10449

https://osf.io/kn5f2/download

https://tedboy.github.io/nlps/generated/generated/gensim.models.LdaMulticore.html

https://github.com/RaRe-Technologies/gensim/blob/develop/gensim/models/ldamulticore.py

https://siddharth1.medium.com/text-cleaning-in-practice-real-world-examples-and-best-practices-b4f52a45c6f8#:~:text=In%20summary%2C%20text%20cleaning%20is,and%20reliability%20of%20your%20results.