Conversational Commerce with Hybrid Recommendation Model Using Neo4j

Jismy Xaviar[1], Najma K C[2], Nishy Ann Tomy[3], Aby Abahai T[4]

[1]Student, Dept. of CSE, Mar Athanasius College of Engineering, Kerala, India
[2]Student, Dept. of CSE, Mar Athanasius College of Engineering, Kerala, India
[3]Student, Dept. of CSE, Mar Athanasius College of Engineering, Kerala, India
[4]Associate Professor, Dept. of CSE, Mar Athanasius College of Engineering, Kerala, India

---***---

Abstract - *Ecommerce is the buying and selling activities over digital media. It's a booming business, especially since the COVID pandemic broke out. Consumer adoption of eCommerce has been accelerated by social distancing and lockdowns. As a result, eCommerce is more competitive than it has ever been. To thrive, the company must adopt an omnichannel strategy that incorporates product and customer data to enhance the user experience. In a current way, eCommerce displays products or services in a web shopping style interface on visual websites or mobile interfaces. Customers require assistance in resolving their questions, comprehending their preferences, guiding them, and recommending the best alternative for them. Customer satisfaction, personalization, and product discovery can all be improved by providing timely product recommendations. Hence, we propose a new form of e-commerce where brands and consumers communicate through a conversational bot. Artificial intelligence is used to automate conversations, and interactions between brands and customers feel very human. It provides customer service, interprets the needs of buyers, and aids them as needed. To provide enhanced recommendations, it employs a hybrid recommendation algorithm that blends content-based and collaborative filtering techniques. As the recommendation system was designed based on both similarity of products and user behaviors, the Neo4j graph database was used which facilitates the merging of various recommendation techniques. By involving the customers better and as a result, attract and keep hold of customers, product chatbots can benefit both customers and marketers.*

***Key Words*: Online Shopping, Chatbot, Hybrid Recommendations, Neo4j, Conversational Commerce, Named Entity Recognition**

1. INTRODUCTION

E-commerce has ushered in a new era in business and marketing, providing a better experience for both sellers and buyers while also saving time and money. They are an effective lead generation strategy for online merchants. They engage passive visitors on a retailer's website, app, or other digital touchpoints with intelligent suggestions and convert them into engaged prospects. Customers need someone who can answer their questions, understand their preferences, lead them, and offer the best solution for them to go to the final stage of the sales funnel. To boost sales, you must establish a rapport with your prospects and demonstrate that you are available to assist them. As a result, today's firms must tap into digital traffic and engage with customers through discussions, learning about their preferences. As well as recommending your options. Product chatbots can help with this.

A conversational bot is presented as a new kind of e-commerce in which businesses and consumers connect. It makes use of a hybrid recommendation methodology that incorporates both content-based and collaborative filtering techniques. It makes more precise product suggestions. In addition, it provides customer service, interprets buyers' needs, and assists them as needed.

2. RELATED WORK

E-commerce is growing in popularity by the minute, and more and more people are turning to it for their buying needs. Our motivation for the project stems from the fact that online buyers are likely to experience the problem of being unable to find the products they are seeking for owing to poor searching skills. Tidio Customer Service, for example, is a conversational bot, which exclusively answer customer service questions and do not help users find products. On a different note, existing e-commerce recommender systems use either content-based filtering or collaborative filtering, but not both. What we propose is a hybrid recommendation engine that uses both content-based and collaborative approaches stacked on top of each other to provide users with better precise product recommendations.

3. PROPOSED SYSTEM

3.1 Data Sources

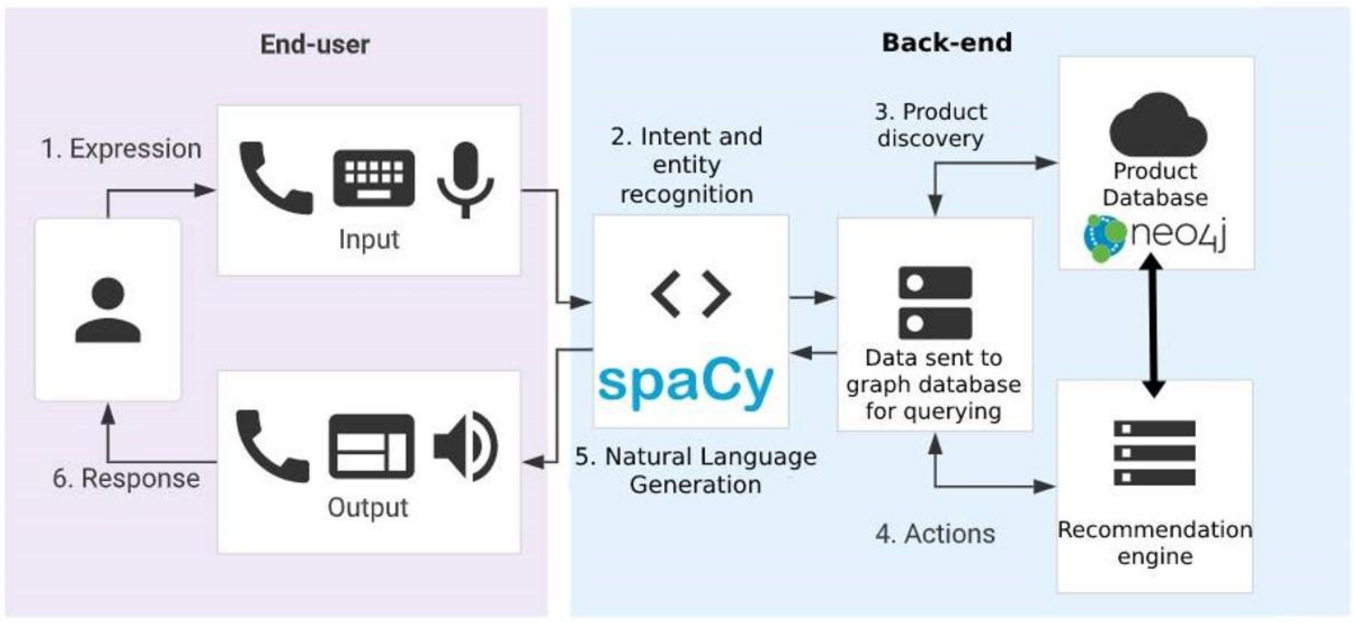

Fig -1: System Workflow

The user chats dataset, the products dataset, and the user rating dataset are all utilized. For training, the user conversations dataset consists of question-answer pairs. Product ids, product names, and product categories are all included in the attributes of the products dataset. The user rating dataset contains the user's user id, the product id of the product purchased by that user, the user's rating of the product, and the purchase date.

3.2 Text Pre-processing

Text pre-processing is carried out on user chats, which includes tokenization, stemming, lemmatization, and labeling. Tokenization is the process of breaking down a large text object into smaller tokens. Punctuation marks, words, and integers are examples of tokens. The process of reducing word inflection to its base forms, such as mapping a set of words to the same stem, is known as stemming. Lemmatization properly reduces the inflected word, ensuring that the base word is a language word. Lemma is the root word of lemmatization. The canonical form, dictionary form, or citation form of a set of words is called a lemma. These procedures transform the text into a format that machine learning algorithms can comprehend.

3.3 Predictions

The predictions model is applied to the pre-processed text after the cleaned text is created. Intent recognition and entity extraction models are the prediction models. They're used to extract data from the user's text and provide the relevant response. The intent of a user's input is represented by intents. Entities are metadata about the intents. Using PyTorch, the intent classifier analyses text and categorizes it into several categories such as "greetings," "buy," and so on. A named entity is a real-world object, such as a product, a person, or an organization, that can be identified by a proper name. Named Entity Recognition is a supervised learning algorithm. Named Entity Recognition detects named entities like props, organization, etc, and predicts the entities based on which was trained using labeled data. It takes a string of text(sentence or paragraph) as input and identifies relevant nouns mentioned in that string. A NER model aids information extraction by detecting and categorizing a named entity. This was done with the help of Spacy, a Python module for creating text processing and interpretation tools.

3.4 Similarity Detection

Similarity discovery between products is accomplished using the products dataset and the user rating dataset, and similar products are linked using relations. Similar objects and products are identified using cosine similarity. The cosine similarity score can be anywhere between 0 and 1, with 1 being the most similar.

3.5 Hybridization

To produce hybrid product suggestions, content-based filtering and collaborative filtering procedures are piled on top of each other after identifying related users and products. The Neo4j database, which contains user and product data, is then queried using these methods, and the results are shown to the user.

4. IMPLEMENTATION

4.1 Chatbot

Chatbots have become a leading topic due to breakthroughs in the science of deep neural networks, particularly in natural language processing (NLP). We used a generative chatbot model for this project. Generative models [4] are built to solve the problem of responding to a non-predefined response. Generative models can handle new cases because they do not rely on any predefined response and create their response starting from the person they must respond to. These models are special in the sense that they can give the users the feeling of talking to a real human. Since there are no predefined answers, these models need to learn how to build responses using a large collection of conversations.

4.1.1 Intent Recognition

Our approach is based on a model that is rather akin to PyTorch's implementation of a two-layer feed-forward neural network. The chatbot needs to handle questions about product details, product recommendations, product delivery, and order inquiry.

Conversational intents should be defined in a chatbot framework's structure. A JSON file is a simple way to accomplish this. Each conversational intent is mainly composed of:
1. A tag - This is a name to uniquely identify the intent.
2. Patterns - These are sentence patterns for the neural network text classifier.
3. Responses - These are a set of responses out of which one will be randomly selected and generated as the response to the user.
The training of the model must be re-run whenever the JSON file is modified.

We cannot just pass the input sentence as it is to our neural net. We somehow have to convert the pattern strings to numbers that the network can understand. For this, we convert each sentence to a so-called bag of words (bow). To accomplish so, we'll need to gather some training words, i. e., all the words that our bot can have a look at in the training data. We can then determine the bag of words for each unseen sentence using all of these words. The dimensions of the bag of words array are equal to that of the all words array, and each place holds a 1 if the word appears in the incoming phrase, and a 0 otherwise.

We use two additional NLP techniques before we can calculate the bow: tokenization and stemming.
Tokenization is one of the most popular methods for working with data in natural language. Tokenization is the act of breaking down a phrase, sentence, paragraph, or even a whole text document into distinct words or phrases. Tokens are the titles given to every one of these small components. Words, numerals, or punctuation marks could be identified as tokens. The point where one word terminates and the next commences is referred to as a word boundary. These tokens are used in the stemming process as a preliminary step.

Stemming is the removal of a portion of a word or the reduction of the given word to its root. More results are returned as more forms of words are recognized, searched for, and retrieved. When a word's form is identified, it may be possible to return search results that would otherwise be missed. Stemming is essential to search queries and information retrieval because of the additional information retrieved.

For the labels, they are sorted alphabetically and the index is used as the class label. Once the tokenization and stemming processes are completed, the bag of words is generated as an array of zeroes and ones. The NLP utils are then implemented using Python's NLTK (Natural Language Tool Kit) package, which includes several useful methods.

The implementation of the neural network is straightforward with a feed-forward neural net with two hidden layers. This is followed by the implementation of the training pipeline where every module is put together. The chat is implemented by loading the trained model and making predictions for new sentences.

4.1.2 Custom Named Entity Recognition

A fundamental NLP task is named entity recognition, which can distinguish entities discussed in a text document. A model that can match this criterion is a Named Entity Recognizer. It should be able to recognize named elements such as "Google," "Sarah," "India," and so on, and classify them as ORGANIZATION, PERSON, LOCATION, and so forth. It is a highly handy tool that aids in the retrieval of information.

SpaCy is a natural language processing library, that is frequently utilized due to its advanced and versatile characteristics. The pipeline component 'ner' implements named entity recognition in spaCy. Training data is accepted as a list of tuples by spaCy [6].
The text and a dictionary should be included in each tuple. The dictionary must include the indices where the

recognized entity begins and ends in the text, as well as the named entity's class or label. For example,

("Sarah works at Google.", {"entities": [(0, 5, "PERSON"), (15,21, “ORG”)]})

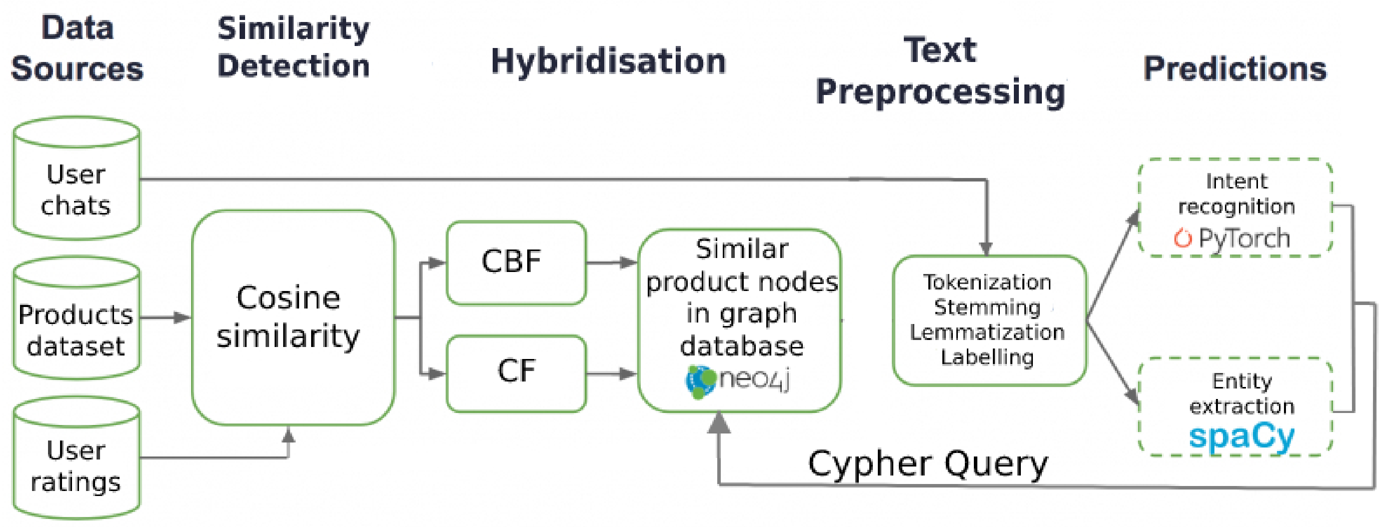

Fig -2: Architecture

The major part of the work involved lies in creating the custom entity data for the input text where the named entity is to be identified by the model during the testing period. This process is called utterance manufacturing. Then, we trained the recognizer by disabling the unnecessary pipeline except for NER. The nlp_update function was used to train the recognizer. The training consisted of 300 iterations, with the model or ner being updated after each iteration using the nlp.update() function. Finally, all of the learning was conducted in the context of the NLP model with the other pipelines disabled to avoid engaging the other components.

4.2 Recommender System

Recommender systems are automated systems to filter entities such as products, ads, movies, or songs. It can suggest items to the customers that they may be interested in. Recommender systems help businesses to attract and retain customers. Hence, they have become an integral part of businesses. There are different types of recommendation strategies.

A content-based recommender [3] is a simple recommender. Here for each item, a relevance vector is found which determines how each genre is relevant for the product. TF-IDF scores can be used here to get real-valued vectors. It identifies the most significant words from the document. Term Frequency (TF) calculates how often a particular term appears in a document. Inverse Document Frequency (IDF) gives the frequency of a term found in all documents. TF-IDF vectorizer transforms text to feature vectors. This can be then used for calculating distances, many similarity coefficients can be calculated.

The most widely used similarity coefficients are Euclidean, Cosine, Pearson Correlation, etc. Here we make use of cosine similarity as it has less complexity compared to other similarity measures.

It calculates the similarity between two documents by measuring the cosine of the angle between their vector representations. Using cosine similarity similar items are identified based on genre or categories. And for a given user, the items which are similar to items bought by the user are given as recommendations. The benefit of the content-based recommender is that items not yet rated by any user can be recommended and is useful in solving cold start problems. But it can work only with data provided by the users and recommendations are not unique.

Collaborative filtering makes use of user behavior only. It recommends items based on users with similar patterns. There are different collaborative filtering approaches. In Item-item collaborative filtering, the similarity between items is calculated using ratings given by the user for those items. Then for a particular user, we find all items highly rated by him. Items similar to these items are identified using cosine similarity. These items are given as recommendations to the user by filtering items that have already been bought. In user-item filtering, we find similarity between users from ratings, and users are paired based on similarity scores. For user1, find products highly rated by the user2 and filter which was already bought. This is the recommendation given. The problem with these two techniques is that as we are using a user-item matrix consisting of ratings and most of the users may not rate items they used. Hence the size of the matrix is large but most elements are 0. i.e. sparsity is high. So, we apply Matrix factorization (MF), an unsupervised learning method for dimensionality reduction. It removes redundant features and discovers hidden correlations in the raw data. Singular Value Decomposition (SVD) is a matrix factorization method that can express a matrix as a product of two rotation matrices and one scaling matrix. i.e., when users' rating matrix is given as input it returns a user's feature matrix, a diagonal matrix, and an item features matrix. And it predicts ratings of each item that the user may give them and returns the item with the highest predicted rating by filtering out items that user has already rated. The advantage is that user gets an extensive exposure to many items that they may be interested in. But it has the disadvantage of the cold start problem.

To keep the strengths of these models and to get rid of their weaknesses, individual models can be pipelined for better results. Content-based filtering can be used as the first recommender to determine items to be recommended to the user. This is then passed to the SVD based collaborative model which filters and sort these recommendations based on predicted ratings.

5. RESULT ANALYSIS

Content-based filtering for product recommendations is based on genres. We have clustered products based on categories with the KNN classifier for evaluation. The classifier label returned by KNN classifier to the products recommended by the content-based filtering is compared to the classifier label of the product on which the model recommends. The hit and error are calculated accordingly to measure accuracy. Accuracy is measured using the hit ratio. It is the ratio of the number of hits to the total recommendation ratio. In content-based filtering, we got a model accuracy of 0.932. The hit ratio of User-User and Item-Item filtering models are 0.7 and 0.8 respectively. SVD model is evaluated using Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). MAE is the sum of all errors divided by the number of data points. RMSE difference between the model predictions and training data. The difference between SVD predicted ratings and the actual rating given by the user is calculated using these metrics. MAE and RMSE of these models were found to be 0.67 and 0.87 respectively. The comparison of these models is shown in the graph. The hit ratio is high for the content-based model. Since the hit ratio is higher for content-based we can choose the recommendation as to the combination of content-based and SVD.

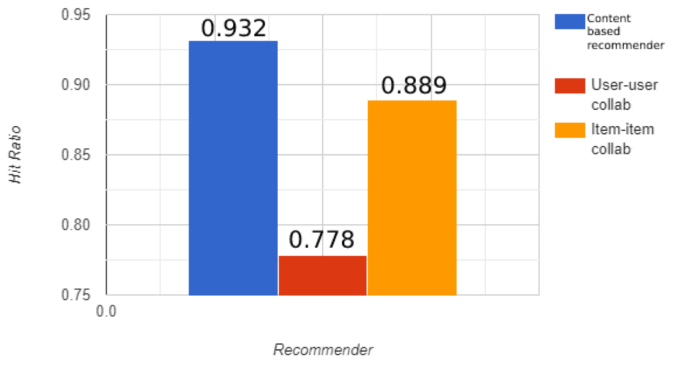

Chart -1: Comparison of the Hit Ratios of different recommenders

6. CONCLUSIONS

We were able to successfully design and implement an e-commerce chatbot with product recommendations. The output of the custom-named entity recognition model is key-value pairs of entity names and the part of the input string that match the entity. The hybrid recommendation engine sequentially combines content-based and collaborative filtering approaches to generate a hybrid recommendation. This approach may be extended and utilized in any e-commerce platform to improve the customers' purchasing experience, create customer trust and loyalty, improve brand recognition, scale support to numerous consumers, and help you achieve higher brand recognition.

REFERENCES

[1] Basu, C., Hirsh, H. and Cohen W.: 'Recommendation as Classification: Using Social and Content-Based Information in Recommendation', 1998.

[2] Eeuwen, M. van Mobile conversational commerce: messenger chatbots as the next interface between businesses and consumers, 2017.

[3] Rohini Nair and Kavita Kelkar "Implementation of Item and Content based Collaborative Filtering Techniques based on Ratings Average for Recommender Systems", International Journal of Computer Applications (0975-8887), Volume 65- No.24, March 2013.

[4] Nguyen, T., & Shcherbakov, M. A Neural Network based Vietnamese Chatbot. 2018 International Conference on System Modeling & Advancement in Research Trends (SMART), 2018.

[5] https://chatbotsmagazine.com/contextual-chat-bots-with-tensorflow-4391749d0077

[6] https://towardsdatascience.com/train-ner-with-custom-training-data-using-spacy-525ce748fab7