

Question Answering System that would help the user decide on a product to buy.

By: Dina Tamer and Malak Amr

May 19, 2024

1 Introduction

Question answering is a critical NLP problem and a long-standing artificial intelligence milestone. QA systems allow a user to express a question in natural language and get an immediate and brief response. QA systems are now found in search engines and phone conversational interfaces, and they're fairly good at answering simple snippets of information. On more hard questions, however, these normally only go as far as returning a list of snippets that we, the users, must then browse through to find the answer to our question.

In response to this problem, our project endeavors to introduce a groundbreaking Question Answering System (QAS) empowered by the latest advancements in Natural Language Processing (NLP). Our system is designed to revolutionize the way consumers interact with e-commerce platforms, offering a personalized and intuitive approach to product discovery. By harnessing the power of NLP, our QAS aims to understand and interpret user queries in natural language, enabling seamless communication between the user and the system.

2 Motivation

The motivation behind our project stems from a deep-seated desire to alleviate the pain points commonly associated with online shopping, particularly the overwhelming nature of product selection. Traditional search engines and recommendation systems, while effective to some extent, often fall short in providing tailored recommendations that align with the unique preferences and requirements of individual users. Recognizing this gap, we seek to leverage the capabilities of NLP to develop a QAS that not only understands the nuances of user queries but also extracts relevant information from product descriptions, reviews, and specifications to deliver concise and informative responses.

Our objectives are multi-faceted, aiming to address various challenges inherent in the process of product selection. Firstly, we strive to develop a robust and efficient QAS capable of accurately interpreting user queries and providing relevant answers in real-time. Secondly, we aim to enhance the adaptability of our system by integrating machine learning techniques, allowing it to evolve and improve its understanding of user preferences over time. Lastly, we seek to empower consumers by offering a seamless and personalized shopping experience that simplifies the decision-making process and fosters confidence in their purchasing decisions.

Throughout this paper, we will delve into the methodology and architecture of our QAS, outlining the techniques and algorithms employed for natural language understanding, information extraction, and machine learning integration. We will also present the results of our system's performance evaluation, demonstrating its effectiveness in assisting users with product selection. By offering insights into the implications of our work and potential applications in the realm of e-commerce optimization, we aim to contribute to the advancement of consumer-centric technologies and pave the way for a more seamless and enjoyable online shopping experience.

3 Literature review

In the realm of QAS, various techniques and approaches have been explored to enhance the user experience and decision-making process. From traditional rule-based systems to advanced machine learning models, researchers have delved into diverse strategies to improve the accuracy, efficiency, and user-friendliness of QAS for product recommendation. This literature review will traverse the landscape of methodologies, encompassing natural language processing (NLP), machine learning algorithms, sentiment analysis, and user modeling. By examining prior works, we aim to identify the strengths and limitations of each approach, paving the way for the synthesis of a novel QAS that excels in assisting users in making informed decisions when choosing products to purchase.

The paper [1] focuses on addressing the challenge of providing quick responses to user questions in e-commerce websites by automatically identifying plausible answers from product reviews. The motivation behind the study is to enhance user experience by reducing the waiting time for responses and improving the efficiency and trustworthiness of online shopping. To tackle this problem, the researchers proposed a novel multi-task deep learning method named QAR-net, which incorporates carefully designed attention mechanisms. This method leverages large-scale user-generated QA data and manually labeled review data to train an end-to-end deep model for answer identification in review data. The model consists of three sub-networks: Q-subnet, A-subnet, and R-subnet, which utilize Bidirectional Gated Recurrent Unit (Bi-GRU) for feature extraction and attention techniques for generating high-level embeddings of texts. Experiments conducted on data collected from Amazon demonstrated the effectiveness of QAR-net, showing that it outperformed baseline methods. The results of the implementation indicated that the proposed method successfully identified plausible answers from product reviews for user questions, showcasing its potential to improve the user experience in e-commerce platforms. In conclusion, the researchers found that QAR-net, the multi-task attentive model they developed, was effective in leveraging user-generated QA data to enhance question-answer matching in e-commerce settings. The study highlighted the importance of utilizing deep learning techniques and attention mechanisms to automate the process of extracting answers from reviews, ultimately providing users with timely and relevant information during their online shopping experience.

A survey paper on Product Question Answering system In [2] provides a clear framework for understanding the different approaches and methodologies employed in addressing product-related questions. This categorization allows for a detailed analysis of the methods and evaluation protocols used in each problem setting, offering insights into the strengths and limitations of various techniques in the context of PQA. Furthermore, the paper delves into the unique challenges that characterize PQA and differentiate it from traditional Question Answering (QA) systems. By focusing on the subjectivity, reliability, multi-type resources, and low-resource issues specific to PQA in E-Commerce platforms, the paper sheds light on the complexities involved in effectively answering product-related queries. Understanding these challenges is crucial for developing robust solutions that can handle the diverse nature of user-generated content and the dynamic landscape of E-Commerce platforms. By systematically analyzing existing research efforts and discussing potential future directions, the paper aims to provide a comprehensive overview of the state-of-the-art in PQA and guide researchers towards addressing key challenges in this domain.

In [3], they introduced a comprehensive framework for addressing user queries on e-commerce product pages, with a specific emphasis on handling both factoid and non-factoid questions. The framework is designed to leverage a combination of deep learning-based distributional semantics and structured semantics imposed by a domain-specific ontology. It encompasses various components such as question category classification, question-answer annotators, deep learning-based sentence embedding, and question-answer matching models. The proposed system aims to address the challenges of identifying question intent, product attribute-value extraction, semantic matching, and maintaining high precision in providing accurate answers. Additionally, the system is designed to overcome the scarcity of training data and other resources typically encountered in domain-specific question answering systems. The paper also includes an evaluation of different components of the framework and an ablation study to analyze their contribution to the system’s performance. The evaluation demonstrates that the proposed system achieves a 66% higher precision compared to a baseline model, indicating its effectiveness in addressing user queries on e-commerce product pages. Overall, the paper provides a detailed and systematic approach to developing a question-answering system for e-commerce product pages, highlighting the interplay of various components and their impact on system performance.

The article [4] delves into the intricate task of generating accurate and meaningful answers for product-related questions within e-commerce platforms. The researchers introduce the innovative Meaningful Product Answer Generator (MPAG) model as a solution to the challenges faced in this domain. The methodological approach employed in the study involves a comprehensive framework comprising a review clustering algorithm, a review reasoning module utilizing a write-read memory architecture, an attributes encoder based on a key-value memory network, a prototype reader for extracting answer skeletons, and an answer generator incorporating an editing gate to fuse reasoning results with product attributes dynamically. The evaluation of the MPAG model is conducted using various metrics, including BLEU for lexical unit overlapping, embedding-based metrics for assessing semantic similarity, and the distinct metric for evaluating answer diversity. The dataset utilized in the research consists of a substantial collection of information from 469,953 products spanning 38 categories, with each product having an average of 59.1 reviews and 9.0 attributes associated with it. The results of the experiments showcase the superior performance of the MPAG model compared to existing baselines, demonstrating its capability to consistently generate specific and proper answers for product-related questions in e-commerce settings. The conclusion emphasizes the model’s effectiveness in providing reasonable explanations for the generated answers, mitigating the safe answer problem, and enhancing reasoning abilities in generating meaningful responses, thereby establishing the MPAG model as a valuable tool for e-commerce question-answering tasks.

4 Methodology

In this section, we outline the comprehensive methodology employed to predict the price of products based on their descriptions and brands. The methodology comprises three primary phases: data analysis, data preprocessing, and model implementation. Each phase is integral to the overall process, ensuring that the data is meticulously prepared and the models are effectively trained. Our approach to solving the problem involved two distinct strategies: creating a custom model from scratch and leveraging a pre-trained BERT model. By exploring these two methodologies, we aimed to gain insights into the advantages and limitations of a bespoke model compared to a state-of-the-art pre-trained language model in the context of price prediction based on textual data.

4.1 Data Analysis

Before conducting the data analysis, some data preprocessing had to be done on the dataset, as there were null values in several rows which might have caused inaccurate results. In preprocessing the dataset, several steps were undertaken to enhance data quality and completeness. Initially, missing descriptions were filled by leveraging the corresponding titles, as the titles section contain some sort of description of the product. Next, missing values in the price/value column were imputed using a K-nearest neighbors (KNN) approach, where the model was trained on the entire dataset to predict missing prices based on other relevant features. Similarly, missing stars ratings were imputed using KNN imputation, ensuring that the imputed values were rounded to two decimal places for consistency. Additionally, missing reviewsCount values were imputed using KNN imputation and subsequently rounded to the nearest integer. Finally, any missing entries in the price/currency column were filled with the string '\$'. These preprocessing steps collectively contributed to a more comprehensive and reliable dataset suitable for further analysis and modeling.

4.1.1 Data Analysis Results

Data analysis was the initial phase of the project. We began by examining the dataset, which consisted of three columns: 'description', 'brand', and 'price/value'. The analysis included checking for missing values, understanding the distribution of prices, and identifying any patterns or anomalies in the data. This preliminary analysis was crucial for informing subsequent data preprocessing steps and model selection.

After applying the data preprocessing, We started our analysis by extracting the main keywords in the products' descriptions. Figure 1 shows the output of this analysis.

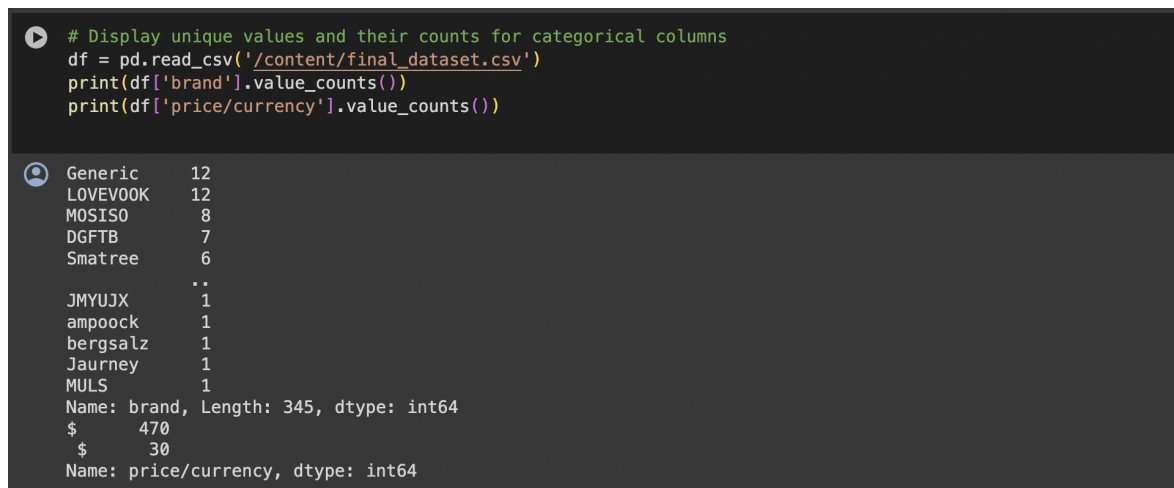


Figure 1: Descriptions Keywords

Then we checked the star ratings of all the products in the amazon dataset and here are the results. Figure 2 shows that the average star ratings of all products was 4.5.

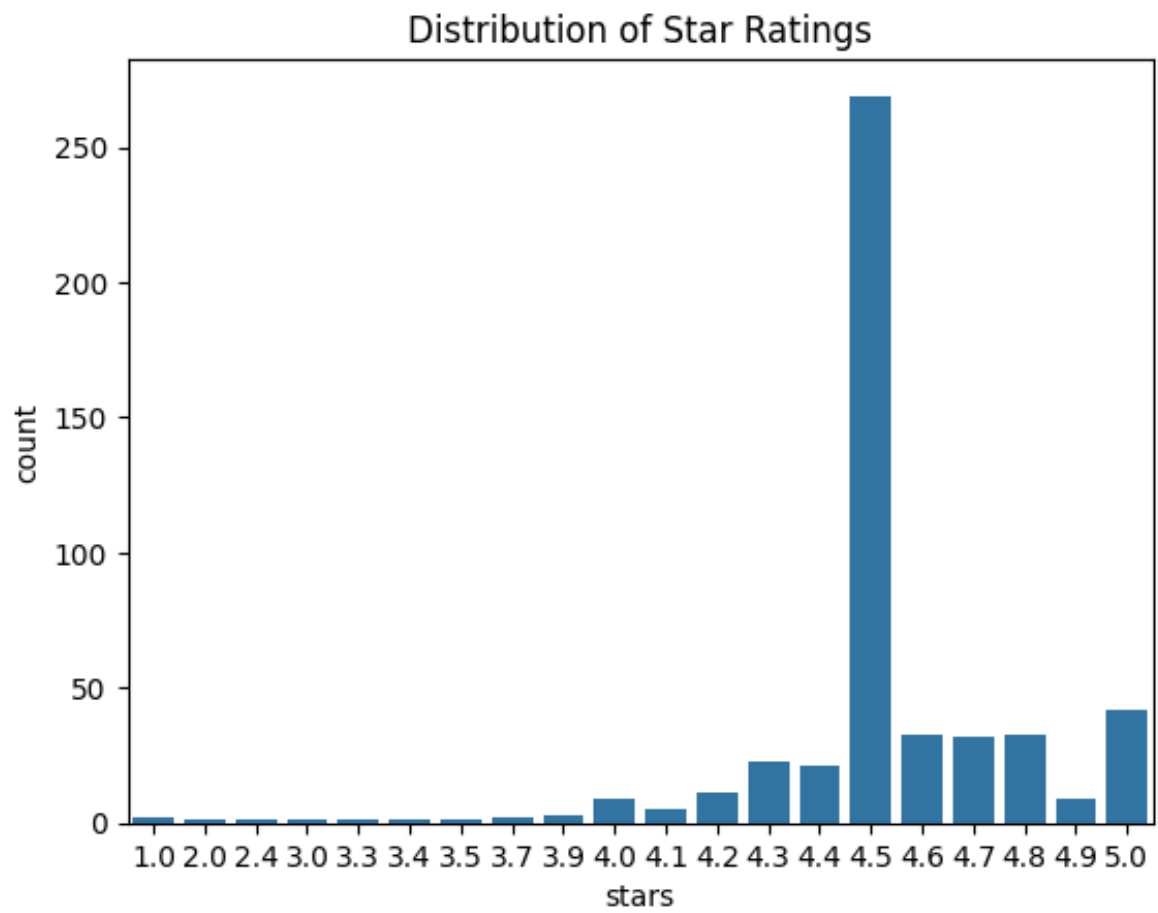


Figure 2: Star Ratings

Then we imported wordcloud to generate a word cloud of the most frequently used words in the description column. Figure 3 shows the output word cloud.



Figure 3: Word Cloud

Furthermore, we downloaded all of nltk files and conducted sentiment analysis to show the sentiment scores of the different descriptions. Figure 4 is a bar chart that shows the output of this sentiment analysis, which shows that most the sentiment scored were neutral and the positives scores were more than the negative ones.

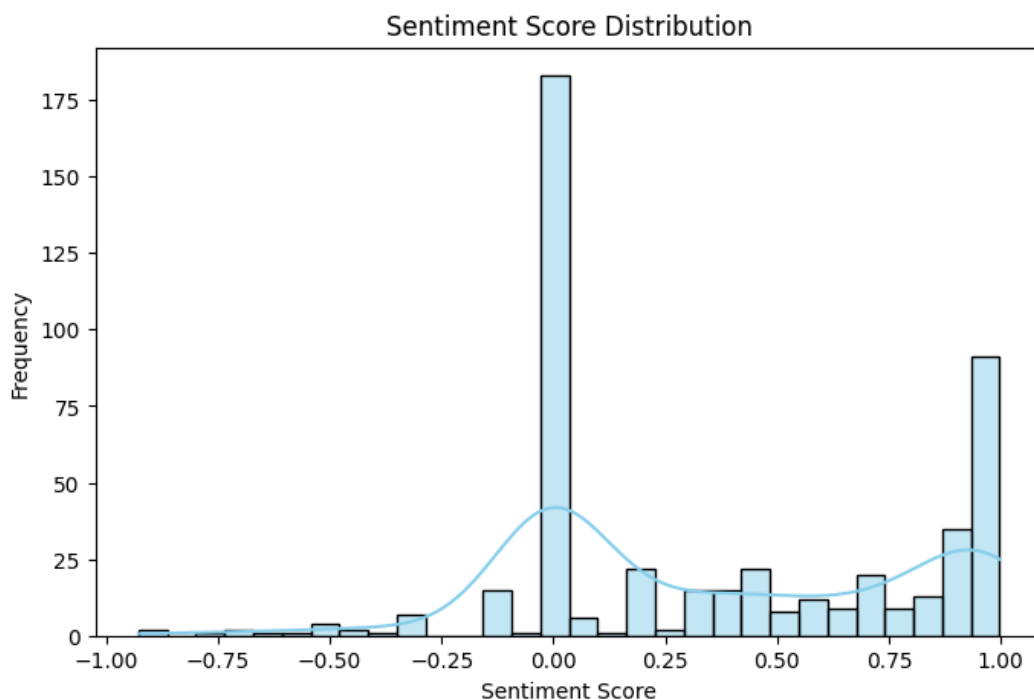


Figure 4: Sentiment Analysis

Figure 5 is another display of this sentiment analysis.

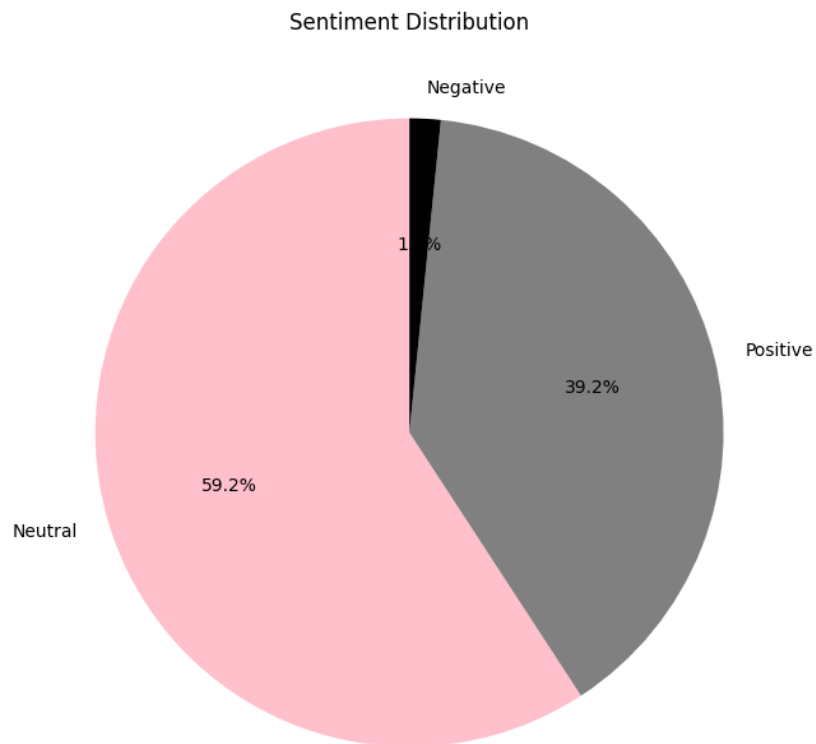


Figure 5: Sentiment Analysis pie chart

Moreover, we conducted a price analysis which shows us the frequency of the prices and the price distribution among all products. Figure 6 shows the output of this analysis.

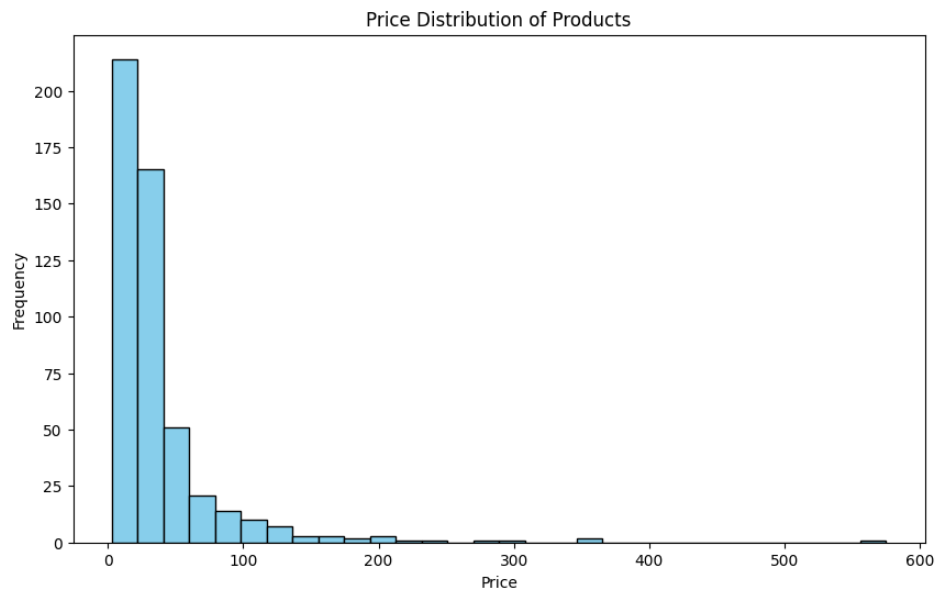


Figure 6: Price Distribution

We also conducted an analysis by grouping the brands and the number of descriptions of each brand. Figure 7 shows the output of this analysis.

```

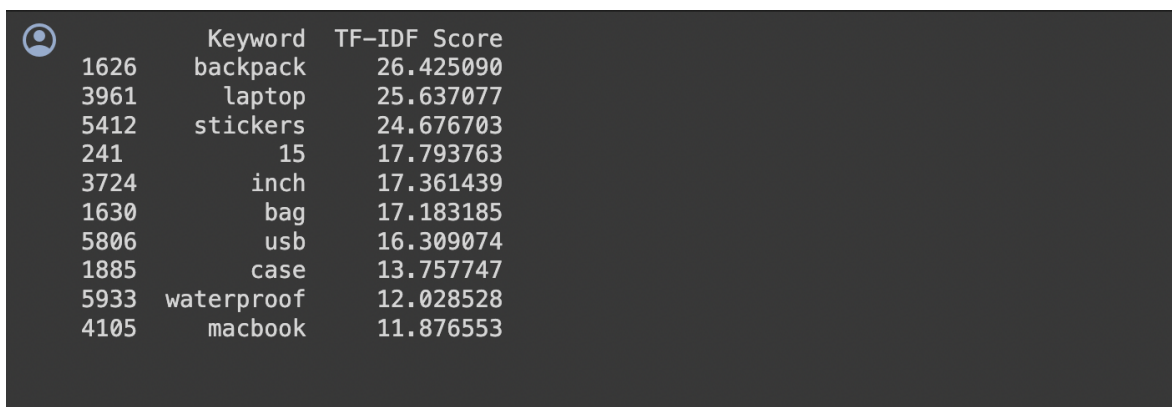
brand
LOVEV00K      12
Generic       12
MOSISO        8
DGFTB         7
Smatree        6
...           ...
JGTM           1
JESW0          1
Inateck        1
Icicrim        1
wonfurd        1

[345 rows x 1 columns]

```

Figure 7: Brand Grouping

Lastly, TF-IDF was utilized to identify significant keywords from the dataset. TF-IDF assigns weights to words based on their frequency in individual product descriptions and rarity across all descriptions. This allowed us to extract essential terms that represent the products effectively. By calculating TF-IDF scores for each word, we pinpointed the most important keywords in the dataset. This approach enabled us to highlight key features of the products and gain insights into their characteristics, aiding in subsequent analysis and decision-making processes. Figure 8 shows the result of this analysis.

A terminal window with a dark background and a user icon in the top-left corner. It displays a table of TF-IDF scores for various keywords, sorted in descending order of score. The table has three columns: a numerical value, a keyword, and the TF-IDF score.

	Keyword	TF-IDF Score
1626	backpack	26.425090
3961	laptop	25.637077
5412	stickers	24.676703
241	15	17.793763
3724	inch	17.361439
1630	bag	17.183185
5806	usb	16.309074
1885	case	13.757747
5933	waterproof	12.028528
4105	macbook	11.876553

Figure 8: TF-IDF

4.2 Experimental Methodology

In this section, we will discuss the two approaches we mentioned above: the architecture of the models, how they were compiled and the preprocessing of the dataset we followed.

4.2.1 Approach 1: Custom Model from Scratch

The initial step of this approach involved a thorough examination of the dataset to understand its structure and content. The dataset comprised various columns such as 'title', 'brand', 'description', 'price/currency', 'stars', and 'reviewsCount'. For this approach, only the 'title' and 'price/value' columns were retained, while the rest were discarded. This decision was made to focus on predicting the price based solely on the product title. After that, some data preprocessing was made as it is essential for preparing the text data for modeling. The preprocessing steps included:

- Text Cleaning where the product titles were cleaned by removing stopwords and punctuation. This was achieved using NLTK's stopwords list and tokenization functions. Custom functions were created to automate this process for the 'title' column.
- Rounding Prices where the 'price/value' column was rounded to two decimal places to ensure consistency in the price values.
- Tokenization and Padding where the cleaned text data was tokenized using TensorFlow's Tokenizer, converting the text into sequences of integers. These sequences were then padded to a fixed length of 50 which is the maximum length we assigned for the titles to ensure uniform input size for the neural network.

Then for the model architecture A custom neural network model was built from scratch using TensorFlow's Keras API. The architecture of the model is as follows:

1. Embedding Layer which converts the input sequences into dense vectors of fixed size (64 dimensions). This layer helps in capturing the semantic meaning of the words in the titles.
2. Bidirectional LSTM Layers: Two bidirectional LSTM layers, each with 128 units, were used to capture contextual information from both directions in the text. This is crucial for understanding the meaning of the text in a more comprehensive manner.
3. Dropout Layers: Dropout layers with a rate of 0.2 were added after each LSTM layer and dense layer to prevent overfitting and improve the model's generalization ability.
4. Dense Layers: A dense layer with 64 units and ReLU activation was added, followed by another dropout layer. This layer helps in learning complex patterns in the data and gather all outputs from previous layers together.
5. Output Layer: The final output layer consisted of a single neuron with a linear activation function to predict the price of the product.

The model was compiled using the Adam optimizer and mean absolute error as the loss function. The training process involved running the model for 30 epochs on the training data.

Here is a detailed description of how the model works with any input to produce the required output, consider the following steps:

1. Input Processing: When a new product title is provided, it undergoes the same preprocessing steps as the training data. This involves cleaning the text, removing stopwords and punctuation, tokenizing the text, and padding the token sequences to the fixed length of 50.
2. Embedding: The preprocessed input sequence is then fed into the embedding layer of the model. This layer converts the sequence into dense vectors.
3. Contextual Understanding: The embedded vectors are then passed through the bidirectional LSTM layers, which captures contextual information from both directions of the text. This enables the model to understand the relationship between words in the title.

4. **Pattern Learning:** The output from the LSTM layers is processed through dense layers with ReLU activation functions, allowing the model to learn complex patterns and relationships in the data. Dropout layers are used to prevent overfitting during this process.
5. **Price Prediction:** Finally, the processed data reaches the output layer, which consists of a single neuron. This neuron produces a continuous value representing the predicted price of the product. The linear activation function ensures that the output is suitable for regression tasks.

Then for the model training and evaluation, the model was trained using the preprocessed training data, and the training process was monitored to ensure proper learning. After training, the model's performance was evaluated on the validation set using mean absolute error as the evaluation metric. This metric provided an indication of how accurately the model could predict the prices of products based on their titles.

4.2.2 Approach 2: BERT Model with Additional Regression Layer

The dataset consisted of various columns, but for this approach, we focused on using the 'description', 'brand', and 'price/value' columns. The descriptions provided textual information about the products, while the 'price/value' column contained the numerical price values. Brands were also included to provide additional context to the descriptions.

Preprocessing the text data involved several steps to ensure it was suitable for input to the BERT model:

- **Text Cleaning:** Although BERT's tokenizer can handle a variety of textual formats, basic cleaning was performed to remove any extraneous whitespace and ensure consistency in the data.
- **Tokenization:** The BERT tokenizer was used to convert the product descriptions and brands into sequences of tokens. This involved concatenating the brand and description into a single input string, splitting the text into subwords, and mapping them to integer indices based on BERT's vocabulary.
- **Padding and Truncation:** The tokenized sequences were padded to a maximum length of 300 tokens to ensure uniform input size. Sequences longer than 300 tokens were truncated.

Then for the model architecture A pre-trained BERT model was utilized for its strong performance in natural language processing tasks. BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained language model developed by Google. It has set new benchmarks in various natural language processing (NLP) tasks due to its ability to understand the context of words in a sentence by considering the words that come before and after them. BERT is based on the Transformer architecture, which utilizes self-attention mechanisms to process input text in parallel, making it highly efficient. BERT takes tokenized text as input and processes it through multiple layers of transformers. Each word is represented as a token, and special tokens like [CLS] and [SEP] are added to denote the start and separation of sentences. Unlike traditional models that read text sequentially, BERT reads text bidirectionally, meaning it considers both the left and right context of each word to understand its meaning. BERT uses multiple layers of transformers, which apply self-attention mechanisms to weigh the importance of each word in the context of others. The output is a set of contextualized word embeddings that capture the nuanced meaning of each word in the given context.

In this approach, an additional regression layer was added on top of the BERT model to predict the price based on the contextualized embeddings generated by BERT. The architecture of the model is as follows:

1. **BERT Model:** A pre-trained BERT base model (bert-base-uncased) was used to obtain contextualized embeddings of the input text. BERT's transformer layers effectively capture the relationships between words in the descriptions and brands.
2. **Dropout Layer:** A dropout layer with a probability of 0.1 was added to prevent overfitting by randomly setting some of the output values from BERT to zero during training.

3. **Regression Layer:** A linear layer was added on top of BERT’s pooled output to perform regression. This layer maps the high-dimensional embeddings to a single continuous value representing the predicted price.

The model was compiled using the AdamW optimizer and mean absolute error as the loss function.

Here is a detailed description of how the model works with any input to produce the required output, consider the following steps:

1. **Tokenization and Encoding:** The input description and brand are combined into a single text string. This string is then tokenized using the BERT tokenizer, which splits the text into subwords and maps them to unique integer indices. The tokenizer also adds special tokens ([CLS] at the beginning and [SEP] at the end) to the input sequence.
2. **Padded Sequences:** The tokenized sequence is padded to a fixed length of 300 tokens to ensure uniformity in the input size.
3. **BERT Embeddings:** The padded and tokenized input sequence is fed into the BERT model. BERT processes the input through its multiple transformer layers, generating embeddings for each token in the sequence. These embeddings capture the contextual meaning of each token, taking into account the surrounding words in the input text.
4. **Pooled Output:** BERT produces an output embedding for each token, as well as a special pooled output that represents the entire input sequence. This pooled output is typically the embedding of the [CLS] token, which is designed to capture the aggregate meaning of the sequence.
5. **Dropout Layer:** The pooled output from BERT is passed through a dropout layer. Dropout is a regularization technique that randomly sets a fraction of the input units to zero during training, helping to prevent overfitting.
6. **Regression Layer:** The output from the dropout layer is fed into a linear regression layer. This layer is a simple fully connected layer that maps the high-dimensional BERT embeddings to a single continuous value, which represents the predicted price of the product.
7. **Prediction:** During inference, the model processes new input text in the same way as during training. The final output of the regression layer is the predicted price, which can be compared to the actual price for evaluation.

The training process involved fine-tuning the BERT model on the product descriptions and brands. The preprocessed training data was loaded into PyTorch DataLoader objects, which facilitated efficient batch processing and shuffling of the data. The model was trained for a certain number of epochs, during which the training loop iterated over the batches of data, performed forward passes to compute the predicted prices, calculated the loss, and updated the model parameters using backpropagation. After each epoch, the model was evaluated on the validation set to monitor its performance and ensure it was not overfitting. Once training was complete, the model was evaluated on the test set. The mean squared error was calculated to assess the model’s accuracy in predicting product prices.

5 Experimental Results

5.1 Approach 1: Custom Model from Scratch

In this approach, a custom neural network model was built from scratch using TensorFlow’s Keras API. The model was trained on the preprocessed product titles and prices, and its performance was evaluated using mean absolute error (MAE) on the validation and test sets.

The model architecture consisted of an embedding layer, bidirectional LSTM layers, dense layers, and dropout layers to prevent overfitting. The training process involved running the model for 30 epochs on the training data, and the validation loss was monitored after each epoch to ensure the model was not overfitting.

5.1.1 Training and Validation Loss

- Training Loss: The mean absolute error on the training set was observed to decrease steadily over the epochs, indicating that the model was learning from the data.
- Validation Loss: The validation loss was monitored to prevent overfitting. The model showed good generalization ability, with a relatively low validation loss.

Mean Absolute Error (MAE): The model achieved a mean squared error of 9.1 on the test set. This metric indicates the average absolute difference between the predicted and actual prices.

5.2 Approach 2: BERT Model with Additional Regression Layer

In this approach, a pre-trained BERT model with an additional regression layer was used to predict product prices based on product descriptions and brands. The model leveraged BERT’s ability to understand the context of words in a sentence to generate contextualized embeddings, which were then used for regression.

The model was fine-tuned on the preprocessed product descriptions, brands, and prices. The training process involved using the AdamW optimizer and mean absolute error as the loss function. The validation loss was monitored to ensure the model’s performance was stable.

5.2.1 Training and Validation Loss

- Training Loss: The mean absolute error on the training set was observed to decrease steadily, indicating effective learning.
- Validation Loss: The validation loss was monitored to ensure the model was not overfitting. The model showed good generalization ability with a relatively low validation loss.

Mean Squared Error (MSE): The model achieved a mean squared error of 28.18 on the test set, indicating the average squared difference between the predicted and actual prices.

5.3 Comparison of Results

5.3.1 Training Efficiency

The custom model required 30 epochs to achieve optimal performance, while the BERT-based model was fine-tuned for 40 epochs. Despite the longer training time, the BERT-based model leveraged pre-trained knowledge from the BERT model, which contributed to its performance.

5.3.2 Model Complexity

The custom model was built from scratch with multiple layers, including embedding, LSTM, and dense layers. In contrast, the BERT-based model utilized a pre-trained BERT model with an additional regression layer. The BERT-based model’s complexity in terms of architecture is higher, but it simplifies the feature extraction process by leveraging pre-trained embeddings.

6 Conclusion

Both approaches demonstrated the ability to predict product prices based on textual information, with the BERT-based model showing better performance due to its contextual understanding of the input text. The custom model from scratch, although simpler in architecture and requiring fewer training epochs, showed strong performance and can be considered a viable approach for similar tasks. However, the BERT-based model’s ability to leverage pre-trained embeddings allowed it to achieve higher accuracy, showcasing the benefits of transfer learning in natural language processing tasks.

The choice between these approaches depends on the specific requirements and constraints of the application. If computational resources and time are limited, the custom model provides a simpler yet effective solution. On the other hand, if higher accuracy is crucial and there are sufficient resources, the BERT-based model is advantageous due to its sophisticated understanding of context.

References

- [1] L. Chen, Z. Guan, W. Zhao, W. Zhao, X. Wang, Z. Zhao, and H. Sun, “Answer identification from product reviews for user questions by multi-task attentive networks,” in *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-20)*. AAAI Press, 2020.
- [2] Y. Deng, W. Zhang, Q. Yu, and W. Lam, “Product question answering in e-commerce: A survey,” 02 2023.
- [3] A. Kulkarni, K. Mehta, S. Garg, V. Bansal, N. Rasiwasia, and S. Sengamedu, “Productqna: Answering user questions on e-commerce product pages,” in *WWW 2019 Workshop on ECNLP*, 2019. [Online]. Available: <https://www.amazon.science/publications/productqna-answering-user-questions-on-e-commerce-product-pages>
- [4] S. Gao, X. Chen, Z. Ren, D. Zhao, and R. Yan, “Meaningful answer generation of e-commerce question-answering,” *ACM Transactions on Information Systems*, vol. 39, no. 2, pp. 1–26, 2021.