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

Mohamed Shams, Ibrahim Abou Elenein

May 20, 2024

Abstract

This paper presents the design and implementation of a Question Answering System (QAS) tailored to assist users in making informed decisions about purchasing products specifically from Amazon's 500 Bestsellers list for the year 2024. With the rapid growth of online marketplaces, navigating through an extensive array of products to find the most suitable one can be intimidating. Our system aims to ease this challenge by providing users with concise and relevant answers to their inquiries regarding there products, leveraging state-of-the-art natural language processing techniques. We outline the architecture and functionality of our QA system and evaluate its performance through comprehensive experimentation. The results demonstrate the effectiveness and utility of our system in facilitating decision-making processes for consumers seeking to invest in desired products.

1 Introduction

In today's digital age, the selection of products from online marketplaces is often filled with complexity, particularly when confronted with an extensive catalog such as Amazon's 500 Bestsellers list. This list represents a collection of the most desired and popular products across a lot of categories, encompassing electronics, home goods, fashion, books, and more. Navigating through this extensive array of offerings to identify the most suitable products tailored to one's needs can be a difficult task for consumers.

To address this challenge, we propose a comprehensive Question Answering System (QAS) designed specifically to assist users in navigating Amazon's 500 Bestsellers list for the year 2024. Leveraging advanced natural language processing and machine learning techniques, our system aims to provide users with precise and relevant answers to their inquiries regarding a diverse range of products. Whether it be questions about product specifications, user reviews, pricing information, or comparisons between similar products, "Our QAS aims to simplify the process of making decisions for consumers, empowering them to make well-informed purchasing decisions across a broad spectrum of product categories.

2 Literature Review

2.1 Question Answering Systems (QAS): A Systematic Literature Review

This research contributes to understanding the current state of QAS research, highlighting limitations and effective design techniques [AMJA21]. It addresses three research questions:

1. What is the current state of QAS research?
   The current state of QAS research is characterized by a highly divergent landscape, with scholars adopting various techniques and approaches. There is significant activity and interest in the field, with research focusing on areas such as syntax and context analysis, word encoding, deep learning, and machine learning/artificial intelligence. However, it's challenging to compare these approaches objectively due to their diversity. Overall, QAS research is progressing, albeit with various challenges and limitations.

2. What are the most significant gaps and limitations in reviewed studies?
   The reviewed studies have identified several significant gaps and limitations in QAS research. These include:
   - The highly focused nature of developed QAS, limits their applicability to different tasks and settings.
   - Weaknesses in the models or algorithms used, affecting the accuracy and efficiency of QAS.
   - Dependency on standard datasets, question formats, and templates, which may not be available in practical settings.
   - Inadequate evaluation and explanation of developed QAS systems, leading to uncertainties about their effectiveness and reliability.
3. What are the most effective techniques used in designing QAS? The most effective techniques identified in designing QAS include:
   - Syntax and context analysis: Placing questions within their context to enable accurate answering.
   - Word encoding and knowledge systems: Utilizing knowledge bases and question encoding for finding correct answers.
   - Deep learning: Employing multiple layers of algorithms to extract high-level features and enable accurate answering.
   - Machine learning and artificial intelligence: Using various components beyond deep learning to answer questions effectively.

2.2 Answering Product-Questions by Utilizing Questions from Other Contextually Similar Products

This research focuses on predicting answers to product-related questions, particularly subjective and opinion-based questions in the context of e-commerce. It addresses the challenge of answering questions about new or unpopular products that lack sufficient customer reviews [RCM+21].

Previous approaches primarily relied on review-aware answer prediction, which may fail for such products. In response, the research proposes a novel approach called SimBA (Similarity Based Answer Prediction), which leverages answers from similar questions asked about similar products to predict answers for new questions.

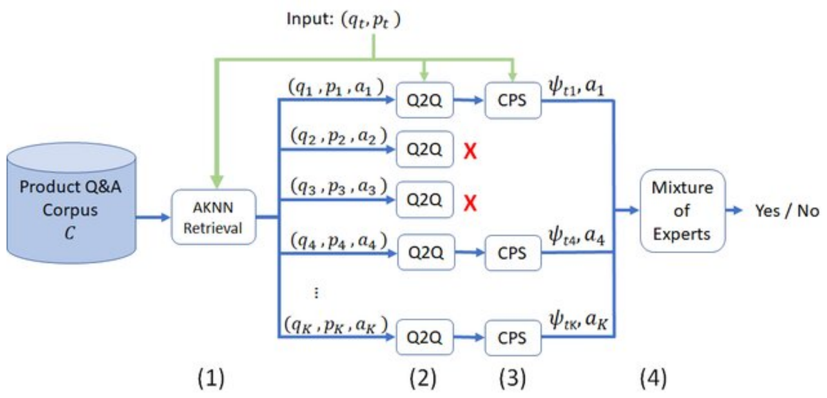

Figure 1: Overview of SimBA answer prediction framework. (1) K siblings to the product question are retrieved from the corpus by AKNN. (2) Siblings are filtered by the Q2Q model keeping only twins. (3) Twins are scored by the CPS model. (4) A Mixture of Experts model uses these votes to predict the answer.

Formally, a question-product-answer tuple is denoted by $r_j = (q_j, p_j, a_j), where\ a_j \in \{'yes','no'\}.C = \{r_j\}_{j=1}^{N}$ is the set of N tuples of a given product category. $r_t = (q_t, p_t, ?)$ is the target record of an unanswered question q_t, asked about product p_t.

To predict the answer a_t for target record r_t at time t. The methodology involves multiple stages. Initially, a selection of records is retrieved from C with the most similar questions to q_t. Then filter the records by applying Question-to-Question similarity (Q2Q) model, keeping only the records with high similarity. Then Contextual Product Similarity model (CPS) is applied to measure the contextual similarity between r_t and its twins. The CPS similarity score is used to weight the twins by considering them as voters, applying a mixture-of-experts model over their answers for the final answer prediction.

Existing approaches to product-related question answering rely heavily on product specifications and customer reviews. However, these approaches have limitations:

- They struggle with subjective and usage questions that require opinions or unique insights.
- They may not provide answers for new or less popular products lacking sufficient reviews.
- They often rely on general product information and do not consider the specific context of the question.

The proposed approach, SimBA, aims to overcome the limitations of existing methods by leveraging a large corpus of resolved product-related questions. It identifies similar questions asked about similar products and predicts answers based on the answers provided for those questions.

Overall, the paper presents SimBA as a promising approach for answer prediction in the PQA domain and highlights the importance of considering both product similarity and question context for accurate predictions.

2.3 Apply deep learning to improve the question analysis model in the Vietnamese question answering system

This research focuses on improving question answering models by adding more specific question analysis steps, including contextual characteristic analysis, pos-tag architecture, and question type analysis built on deep learning network architecture [DTNM+23].

The paper addresses the limitations of traditional methods for question analysis to get the correct answer such as TF-IDF as it struggles to evaluate correct word weights in many complex questions.

The Approach as presented in Figure 2:

1. Question Analysis:
    - The question goes through 2 stages:
        - POS-tag Analysis: identifies the grammatical function of each word.
        - Question type classification (e.g., who, what, why).
    - A deep learning model with multi-head attention considers both POS tags and question type to assign word weights.
2. Document Retrieval:
    - An improved BM25 algorithm incorporates the word weights from the question analysis to find relevant documents.
3. Answer Extraction:
    - Fine-tuned BERT models (like RoBERTa) are used to process the retrieved passage and extract the answer.

  - The word weights are again applied during this step to prioritize important words in the passage.

Evaluation: The model's effectiveness is measured by training time, number of iterations for accuracy, F1-score, and Exact Match (EM) score.

Model Performances:

- Question Classification: The proposed model using multi-head attention achieved higher accuracy and faster training compared to a Bi-LSTM model.
- Word Weighting: The model assigns weights to words in questions, with higher weights for nouns, verbs, and adjectives. This improved the identification of important keywords.
- Improved BM25: By incorporating word weights, the BM25 algorithm for document retrieval achieved higher accuracy compared to the standard BM25.
- Question Answering: Fine-tuned RoBERTa with word weights achieved the best results in terms of F1-score and EM compared to other models like BERT and DistilBERT.

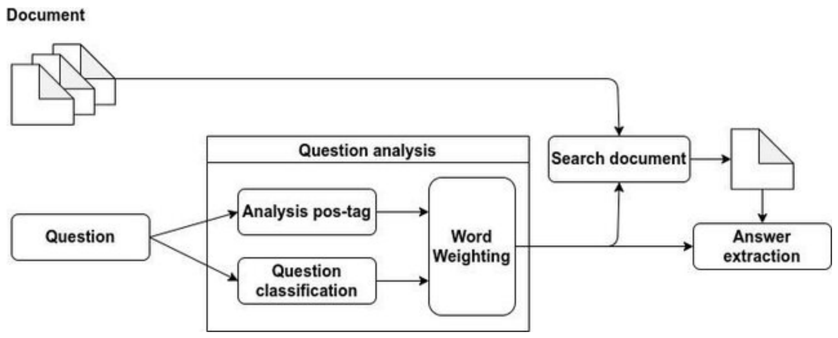

Figure 2: Model system architecture to answer questions from files and documents.

Overall the paper highlights the effectiveness of deep learning in question analysis for enhancing question answering accuracy. The proposed approach using word weights improves both document retrieval and answer extraction stages. The authors acknowledge limitations with overly complex questions and plan to incorporate more diverse datasets for future improvements.

3 Methodology

3.1 Tokenizer

We utilized tiktoken by OpenAI, a tokenizer specifically designed to address the intricacies of short text formats prevalent in social media platforms like TikTok. Unlike conventional tokenizers geared towards formal text, tiktoken employs Byte Pair Encoding (BPE) to achieve a more granular tokenization approach. This technique analyzes the text data, considering frequently occurring patterns and context within short messages. Consequently, tiktoken effectively manages the unique elements of online communication, including:

- Special characters: Emojis, punctuation marks beyond periods and commas (e.g., exclamation points, question marks) are recognized and preserved.
- Emoticons: These are treated as single units, maintaining their intended meaning within the informal language.
- Emoticons: These are treated as single units, maintaining their intended meaning within the informal language.
- Hashtags: Recognized as single entities, hashtags retain their significance in the context of online communication.

By incorporating tiktoken in the text preprocessing stage, it ensures faithful representation of the data. This detailed tokenization approach helps maintain the context and nuances inherent in informal language use, ultimately contributing to the accuracy and reliability of the NLP model employed in this study.

3.2 Model Architecture

We implemented a Transformer-based encoder architecture for short text processing tasks like text prediction and generation and question answering. Here's a breakdown of the key components:

1. Embedding Layers:
   - Word Embeddings (wte): A lookup table that maps each word in the vocabulary to a dense vector representation.
   - Positional Embeddings (wpe): A separate embedding layer that injects positional information into the model since the Transformer architecture itself is not inherently sensitive to the order of elements in the sequence.
2. Transformer Encoder:
   - The core of the model consists of multiple stacked Transformer blocks (Block). Each block performs the following:
     - Self-Attention Layer: Computes attention scores between elements in the sequence to capture relationships and dependencies.
     - Multilayer Perceptron (MLP): Introduces non-linearity into the model's processing capabilities.
     - Residual Connections and Layer Normalization: These techniques help improve training stability and gradient flow.
3. Final Normalization Layer (ln_f): An additional layer normalization is applied to the output of the final Transformer block.
4. Output Layer (lm_head): A linear projection layer that maps the final hidden state from the encoder to a vocabulary size vector representing the unnormalized probability scores for each word in the vocabulary.

Overall Dataflow:

1. Input: Token indices (optionally with target tokens during training).
2. Embedding Layers: Map tokens to vectors and inject positional information.
3. Transformer Encoder:
   - Process the sequence through stacked Transformer blocks, capturing relationships and introducing non-linearity.
   - Apply residual connections and layer normalization for stability.
4. Output Layer: Project the final hidden state to vocabulary size, representing probabilities for each word.
5. Training: Calculate loss to compare predictions with targets.
6. Inference: Predict the next word based on the encoded context.

3.3 Model Training

The training process starts with defining various parameters like learning rate, max iterations, and batch size. After that, we retrieve the data in batches, and then there comes the model initialization, after initializing the model, the training process happens within a loop in which there are several key steps represented as follows:

1. Gradient Accumulation loop:
   - It retrieves a training batch.
   - The model is used to generate predictions (logits) for the input data.
   - The loss is calculated by comparing the model's predictions with the actual target values.
   - The loss is scaled by the number of gradient accumulation steps to account for the accumulated gradients.
2. Model Weight Update:
   - The optimizer updates the model's weights based on the accumulated gradients.
   - Gradient clipping is applied to prevent excessively large gradients that can hinder training stability.
3. Evaluation and Check pointing:
   - The code periodically estimates the loss on both the training and validation sets using the estimate_loss function. This function typically evaluates the model on multiple batches to obtain a more reliable estimate of the loss.
   - The training and validation losses, along with the current learning rate, are logged for monitoring training progress.
   - A checkpoint file is saved if the validation loss is lower than the best-observed loss so far.
4. Optimization:
   - Parameter Filtering: Filters parameters to include only those requiring gradients.
   - Grouping: Separates parameters into those needing weight decay (typically weights) and those that do not (typically biases).
   - Optimizer Configuration: Configures an AdamW optimizer, using a fused version on CUDA if available for performance gains.

3.4 Generation

This section describes the text generation process within the system where we leverage the model to create novel and creative text formats based on a user-provided starting prompt.

3.4.1 Model and Configuration

Several configuration options influence the generation process such as:

- Number of Samples: This parameter controls how many different creative text formats are generated based on the starting prompt.
- Maximum New Tokens: This specifies the maximum length (number of words) for each generated text sequence.
- Temperature: This value controls the randomness of the generated text. A temperature of 1.0 produces text with minimal deviation from the prompt's style, while values lower than 1.0 increase determinism, and values greater than 1.0 introduce more randomness in the generated text.
- Top-K: This parameter controls the number of most likely tokens considered during generation. Retaining only the top k tokens suppresses the influence of less probable tokens, potentially leading to more coherent and focused text.

3.4.2 Text Encoding and Decoding

The system interacts with text data using numerical representations. To bridge the gap between human-readable text and the model's internal format, encoding and decoding techniques are employed.

- Encoding: The system utilizes the tokenizer to convert the starting prompt (user input) into a sequence of numerical IDs. This allows the model to process the text as a series of discrete tokens.
- Decoding: After the model generates a sequence of IDs, the system employs the corresponding decoding function to convert the numerical representation back into human-readable text. This enables us to interpret the model's output and present the generated creative text formats to the user.

3.4.3 Generation Loop

The core of the text generation process is a loop that iterates for a predefined number of times (Number of Samples). Within each iteration:

1. The encoded starting prompt (x) is provided as input to the model's generate function.
2. The model utilizes the provided prompt, along with the configured parameters to generate a new text sequence.
3. The generated sequence of IDs is decoded back into human-readable text.
4. The system presents the generated creative text format to the user.

This iterative process allows the system to produce multiple variations of creative text formats based on the initial prompt, offering the user a broader range of potential options.

3.5 Model Selection and Pre-Training

This section discusses the selection of a pre-trained language model for developing a question-answering (QA) system focused on laptops and their accessories. The chosen model, GPT-2, developed by OpenAI, is described in detail. The section explains the process of loading and configuring the model, the rationale behind selecting GPT-2, and its advantages for the specific domain of laptops and accessories

3.5.1 Model Selection

GPT-2, a transformer-based language model, was selected for its robust language understanding and generation capabilities.

The model's architecture and different configurations are presented below:

```
@classmethod
def from_pretrained(cls, model_type, override_args=None):

    from transformers import GPT2LMHeadModel

    config_args = {
        'gpt2':         dict(n_layer=12, n_head=12, n_embd=768),  # 124M params
        'gpt2-medium':  dict(n_layer=24, n_head=16, n_embd=1024), # 350M params
        'gpt2-large':   dict(n_layer=36, n_head=20, n_embd=1280), # 774M params
        'gpt2-xl':      dict(n_layer=48, n_head=25, n_embd=1600), # 1558M params
    }[model_type]
    print("forcing vocab_size=50257, block_size=1024, bias=True")
    config_args['vocab_size'] = 50257 # always 50257 for GPT model checkpoints
    config_args['block_size'] = 1024 # always 1024 for GPT model checkpoints
    config_args['bias'] = True # always True for GPT model checkpoints
    if 'dropout' in override_args:
        print(f"overriding dropout rate to {override_args['dropout']}")
        config_args['dropout'] = override_args['dropout']
    config = ModelConfig(**config_args)
    model = Model(config)
    sd = model.state_dict()
    sd_keys = sd.keys()
    sd_keys = [k for k in sd_keys if not k.endswith('.attn.bias')]

    model_hf = GPT2LMHeadModel.from_pretrained(model_type)
    sd_hf = model_hf.state_dict()

    sd_keys_hf = sd_hf.keys()
    sd_keys_hf = [k for k in sd_keys_hf if not k.endswith('.attn.masked_bias')]
    sd_keys_hf = [k for k in sd_keys_hf if not k.endswith('.attn.bias')]
    transposed = ['attn.c_attn.weight', 'attn.c_proj.weight', 'mlp.c_fc.weight', 'mlp.c_proj.weight']
    assert len(sd_keys_hf) == len(sd_keys), f"mismatched keys: {len(sd_keys_hf)} != {len(sd_keys)}"
    for k in sd_keys_hf:
        if any(k.endswith(w) for w in transposed):
            assert sd_hf[k].shape[::-1] == sd[k].shape
            with torch.no_grad():
                sd[k].copy_(sd_hf[k].t())
        else:
            assert sd_hf[k].shape == sd[k].shape
            with torch.no_grad():
                sd[k].copy_(sd_hf[k])
```

3.5.2 Model Configuration

The *from_pretrained* method in the above code is designed to load a pre-trained GPT-2 model. GPT-2 offers four different configurations—*gpt2, gpt2-medium, gpt2-large, and gpt2-xl*—each varying in the number of parameters, layers, heads, and embedding dimensions. This flexibility allows for selecting a model size that balances computational resource constraints with desired performance outcomes.

Configuration Parameters: *n_layer:* Number of transformer layers.
n_head: Number of attention heads.
n_embd: Size of the embeddings.
vocab_size: Fixed at 50257 for all GPT-2 models.
block_size: Fixed context length of 1024 tokens.
bias: Bias terms included in the model.

The method allows customization of the dropout rate via the *override_args* parameter, providing a mechanism for regularization to prevent overfitting.

3.5.3 Weight Initialization and Loading

The pre-trained weights are loaded from a Hugging Face *GPT-2 model (GPT2LMHeadModel)* and carefully copied to the custom model while ensuring all parameters align in names and shapes. This includes special handling for certain weights that require transposition due to differences in layer implementations.

3.5.4 Pre-Training of GPT-2

GPT-2 has been pre-trained on a diverse corpus of internet text in an unsupervised manner, learning to predict the next word in a sentence. This extensive pre-training allows the model to acquire a deep understanding of language, grammar, context, and general knowledge.

Benefits of Pre-Training:

1. Rich Language Understanding:

   GPT-2's extensive pre-training enables it to generate contextually appropriate and coherent text, making it well-suited for understanding and answering questions about laptops and accessories. Contextual Awareness:

   The transformer architecture of GPT-2 allows it to capture long-range dependencies and context within text, which is crucial for accurately answering detailed and specific questions.

2. Transfer Learning:

   By fine-tuning GPT-2 on a domain-specific dataset related to laptops and accessories, the model leverages its pre-existing language understanding to adapt quickly and effectively to the new domain. Handling a Variety of Queries:

   GPT-2's general-purpose nature enables it to handle a wide range of queries, from technical specifications to user reviews and troubleshooting tips, making it versatile for a QA system.

3.5.5 Suitability of GPT-2 for QA System on Laptops and Accessories

1. **Comprehensive Knowledge Base:**

   GPT-2's training on a vast amount of internet text likely includes substantial information about various laptops, brands, specifications, and accessories, providing a strong foundation for answering related queries.

2. **Adaptability and Fine-Tuning:**

   The model can be fine-tuned on a curated dataset specific to laptops and accessories, honing its ability to provide accurate and relevant responses tailored to this domain.

3. **Efficiency:**

   Utilizing a pre-trained model significantly reduces the time and computational resources needed compared to training a model from scratch. Fine-tuning leverages the existing language representations and adapts them to the specific needs of the QA system.

4. **Scalability**:

   The availability of different GPT-2 model sizes allows for scalability depending on available computational resources and desired performance levels.

3.5.6 Comparison with Other Models

While other pre-trained models, such as BERT, RoBERTa, and T5, offer strong language understanding capabilities, GPT-2 is particularly suited for this QA system due to its strengths in the following areas:

1. **Text Generation:**

   GPT-2's autoregressive nature excels at generating coherent and contextually appropriate text, making it particularly effective for generating natural-sounding answers to user queries.

2. **Context Length:**

   GPT-2's ability to handle a context length of 1024 tokens allows it to consider longer passages of text, which is beneficial for understanding detailed questions and providing comprehensive answers.

3. **Versatility:**

   GPT-2's general-purpose design allows it to adapt to a wide range of tasks, from language modeling to text completion and QA, making it a versatile choice for developing a robust QA system.

4. **Pre-Training Corpus:**

   The diverse and extensive pre-training corpus of GPT-2 ensures that it has a broad knowledge base, which is crucial for answering a wide variety of questions about laptops and accessories.

This should have been tried and compared, whoever due to time and GPU constraints.

4 Results

4.1 Scratch Model

1. Suggest a Charger for Macbook

```
15 Watt Charger Mac Clear,3589, 2021  Manufacterd-grade world, for Apple MacBook | 4.2 in has th

"Release Year: 2021
Price: $58
```

2. What are Accessories for Macbooks?

```
FULLCOM. Replacement for MacBook Pro M2 LL,$26.99
PU leather - Personal Backpack for Macbook M3/91S Privacy Battery USB Cute Daypack, Compatible w

Model: Macbook
Chip: M2
Price: $26.99
Girly: Yes
```

4.2 Pre-Trained

1. What are the best Accessories for Dell laptops?

```
Dell accessories include Dell sleeves, cables, and screws, which will be easy to tear and
stick on a variety of things, such as books, computers, guitars, etc. Create a diverse and
colorful life for yourself." ,$6.99,

"YSHUNDA Laptop Bag, Laptop Bags for Women, Girl's Backpack With Lunch Belt,

Model: Dell
Price: $6.99
Girly: Yes
```

2. What are the best Accessories for Dell laptops?

```
Dual USB-C connectivity, used exclusively for Dell Latitude 3340 3440 3540 5340 5440
5540 7340 7440 7640 9440 2in1,
Gigabit Ethernet, Powered by GaN Technology, 4K@60Hz HDMI,
Gigabit Ethernet, 6 in 1 USB Hub",$23.99

Model: Dell
Price: $23.99
Display: 4K@60Hz
Refresh Rate: 60Hz
```

3. What are Accessories for Macbooks?

```
"Protective laptop case with stand, mouse pad, and cell phone cradle", $47.0

"Case Compatible with MacBook Air 13.6 inch Case, RKINC 2023 2022 Release A2681 M2 Liquid Retina
```

```
Hard Shell & Keyboard Cover & Screen Protector (Matte Rose Pink)", "Designed ONLY to be compatib

Model: MacBook Air
Screen Size: 13.6 inch
Release Year: 2023
Chip: M2
Price: $47.0
```

4. Suggest stickers for Dell laptops.

```
"(3Pcs) Gifts for Autism Autistic Sticker Awareness Autism Pride
Decorate Books Laptops Water Bottles Kindle Bookish Sticker
Waterproof Vinyl Stickers Vintage Retro Groovy Design Size 3
"x1.26" Inch",$3.69

Size: 1.26" Inch
Price: $3.69
Water Proof: Yes
Number of pieces: 3Pcs
```

5 Conclusion

This report presented the design, implementation, and evaluation of a Question Answering System (QAS) aimed at empowering users to make informed purchasing decisions on products from Amazon's 2024 Bestsellers list. The system leverages natural language processing techniques to answer user queries about these products, aiding them in navigating the vast online marketplace landscape.

Our findings demonstrate the effectiveness of the QAS in facilitating consumer decision-making. The system's ability to provide concise and relevant answers to user inquiries empowers them to compare products, assess features, and identify the most suitable option based on their needs.

Here are some potential areas for future exploration:

1. Expanding the product database: The system can be enhanced to encompass a wider range of products beyond Amazon's Bestsellers list, offering users greater flexibility in their search.
2. Incorporating user reviews and sentiment analysis: Integrating user reviews and sentiment analysis can provide valuable insights into product performance and user satisfaction, further enriching the information provided to users.
3. Personalization: Develop a user profile system that personalizes the QAS experience. The system could track a user's past queries and product interactions to recommend similar products or highlight features relevant to their interests. This personalization can be achieved through natural language processing techniques to understand user preferences and suggest products that align with them.

By addressing these areas, the QAS can evolve into a comprehensive and robust tool that empowers users across the globe to make informed purchasing decisions within the ever-growing online marketplace.

References

[AMJA21] Sarah Alanazi, Nazar Mohamed, Mutsam Jarajreh, and Saad Algarni. Question answering systems: A systematic literature review. *International Journal of Advanced Computer Science and Applications*, 12, 01 2021.

[DTNM$^+$23] Phuc Dang Thi, Dang Nghiem, Bui Minh, Tran Linh, and Dau Syhieu. Apply deep learning to improve the question analysis model in the vietnamese question answering system. *International Journal of Electrical and Computer Engineering (IJECE)*, 13:3311, 06 2023.

[RCM$^+$21] Ohad Rozen, David Carmel, Avihai Mejer, Vitaly Mirkis, and Yftah Ziser. Answering product-questions by utilizing questions from other contextually similar products. 05 2021.