Product Query Recommendation for Enriching Suggested Q&As

Soomin Lee
lepsoomi@amazon.com
Amazon
Haifa, Israel

Eilon Sheetrit*
eilon_sheetrit@intuit.com
Intuit
Petah Tikva, Israel

Omar Alonso
omralon@amazon.com
Amazon
Palo Alto, USA

Avihai Mejer
amejer@amazon.com
Amazon
Haifa, Israel

ABSTRACT

To help customers who are still in the exploration phase, Web search engines and e-commerce websites often provide relevant Q&As in widgets, such as 'People Also Ask' and 'Customers Also Ask Alexa', with additional information. In this work, we propose to enrich this customer experience by rendering related products under each Q&A based on an automated online query recommendation. We define what are the tenets for high-quality query recommendations and explain why this challenge is different from the existing query re-writing, query expansion and keyphrase generation methods. We describe a data collection method which uses customer co-click information on a proprietary website in order to successfully guide our model into generating query recommendations that satisfy all tenets. Offline and online evaluation results demonstrate that our proposed approach generates superior query recommendations and brings much more customer engagement over strong baselines.

CCS CONCEPTS

• **Information systems** → **Query reformulation**.

ACM Reference Format:
Soomin Lee, Eilon Sheetrit, Omar Alonso, and Avihai Mejer. 2024. Product Query Recommendation for Enriching Suggested Q&As. In *Proceedings of the 2024 ACM SIGIR Conference on Human Information Interaction and Retrieval (CHIIR '24), March 10–14, 2024, Sheffield, United Kingdom.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3627508.3638314

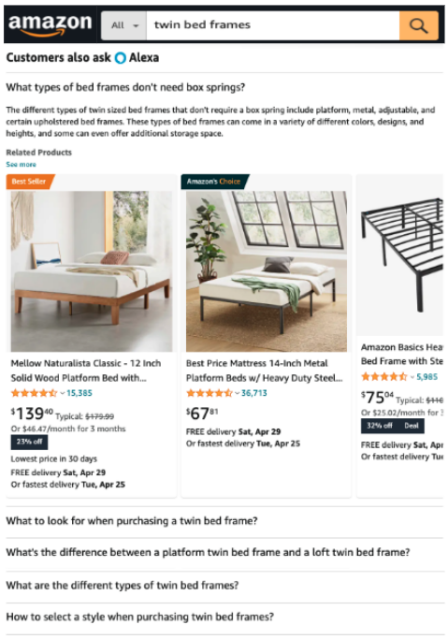

Figure 1: Customers Also Ask Alexa (a.k.a. Suggested Q&A) widget enriched by relevant product search results. A query recommendation *"platform bed frames"* is used for the products that appear in the 'Related Products'.

1 INTRODUCTION

A common feature in Web and e-commerce search engines is to present a set of Q&As in a box that is topically related to the user's search query. Examples are 'People Also Ask' functionality in Google and Bing, and 'Customers Also Ask Alexa' widget in Amazon, respectively. Figure 1 shows an example of the widget that appears upon user's product search request on `amazon.com`, which suggests 4-5 Q&A pairs with additional product information that users may be looking for. When the information in the Q&As leads to a new product discovery, customers often start a new search session by reformulating the initial query. Then, the updated list of retrieved results may lead the customers to a chain of shopping actions that include clicks, add-to-carts and purchases.

Our goal is to provide a shortcut to this long chain of shopping actions, by automatically reformulating the initial search query and rendering the returned product search results directly under each Q&A. In Figure 1, the customer's initial search query is *"twin bed frames"*, and the first suggested question is *"what types of bed frames don't need box springs"*. Upon clicking this question, the box gets expanded to reveal its answer, along with a "See more" link and top-k related products, where clicking the "See more" link further opens the full related products page.

In order to present such a list of related products, we require a product query recommendation (*"platform bed frames"*) that is not redundant with the initial query, but contains additional product information from the Q&As. Due to the e-commerce nature, further business requirements include: i) the query recommendation should

*Portions of this work were done while this author was working at Amazon.

CHIIR '24, March 10–14, 2024, Sheffield, United Kingdom

ACM ISBN 979-8-4007-0434-5/24/03.
https://doi.org/10.1145/3627508.3638314

return enough relevant products to fill all the spots dedicated for the widget; ii) the enriched widget should bring more customer engagement; and iii) a strict latency constraint should be met in order to serve the feature online. Our goal is to develop a lightweight ML model that automatically generates a query recommendation given an initial query and Q&A, which satisfies these requirements. Our contributions are summarized as follows:

- We present a new challenge named "recommending product query for suggested Q&A", which begins from the initial query and provides a query recommendation that retrieves the relevant products described in the suggested Q&A. We define what are the tenets for a high-quality query recommendation and explain why this challenge is different from existing query re-writing, query refinement, query expansion and keyphrase generation methods.
- We provide a novel data collection method which uses customer co-click information on a proprietary website. We provide a detailed explanation on how to filter and prepare the training data in order to successfully guide our model into generating query recommendations that satisfy all tenets.
- We provide offline performance evaluation of our model against baselines including state-of-the-art keyphrase generation models and an LLM with 17B parameters.
- We provide A/B/C (default/our method/human) test results on a real customer traffic to demonstrate that our proposed method does improve customer engagement.

2 PROBLEM DEFINITION

We aim to reformulate the initial query based on the additional product information provided in the suggested Q&A (SQ&A). We address this novel task "recommending product queries for suggested Q&As". More formally, given an initial query I, question text Q and answer text A, we aim to generate a query recommendation R such that R retrieves the relevant products described in Q and A but is not a rephrase of I.

We now turn to explain the tenets of high-quality query recommendations:

1. Not redundant with the initial query. The recommended query should not be redundant with the initial query in terms of its returned search results. For example, *"battery powered fan"* might be a valid query rewrite of *"battery operated fan"*, but it is not a valid recommendation for our task.

2. Aligned with the Q&A. Not all products in the Q&A text are suggested products. In the question *"What types of iPhone cases are wireless charging compatible?"* and answer *"Typically, iPhone cases made of plastic, silicone, rubber, and leather are wireless charging compatible, but cases that are made of metal or aluminum will disrupt the connection between your phone and a wireless charger."*, *"plastic iphone case"* and *"silicone iphone case"* are properly query recommendations, but *"metal iphone case"* and *"aluminum iphone case"* are not. Therefore, the model needs to exactly understand what is the customer problem mentioned in the Q&A.

3. Specific product name. Q&As often list specific product names based on which customers would like to narrow down their search results. For the question *"What's considered a practical gift?"* and answer *"Practical gifts include a travel steamer, electric toothbrush ..."*, existing keyphrase generation models tend to generate a generic query such as *"practical gift"* as they are trained to summarize input text. For our purpose, specific product names such as *"travel steamer"* or *"electric toothbrush"* are better recommendations.

Other business motivations include:

4. Enough returned products. The query recommendation should return at least 5-10 highly relevant products to fill in the widget. For example, *"plywood shelves"* might sound like a valid recommendation from *"shelves"* but since no relevant product exists in the inventory, it is not considered a proper query recommendation.

5. Customer Engagement. A high-quality query recommendation should be helpful for customers, which can be indirectly measured by customer engagement metrics. Achieving better engagement is not a direct requirement, but rather an outcome if a query recommendation successfully meets all the other tenets.

3 DATA COLLECTION

Data generated by humans not only takes more time, but can be subjective and biased by a small team of experts. Instead, we leverage customer created product co-clicks, which are more objective and reflects actual customer behavior. We then apply a filter approach to collect training data that satisfy the previously mentioned tenets.

3.1 Collecting Product Co-Clicks

Figure 2 (left) presents Alexa Instant Answer (AIA) widget that provides answers to customer questions that are submitted in the search bar on amazon.com. Under the widget, a list of products retrieved with respect to the submitted question is also rendered, and if any of the products is relevant to the question, customers click on the product. We collect the customer question (`qtext`), the answer text in the widget (`atext`), and the first clicked product ID (`product_id`) from this log. Figure 2 (right) presents an example of customer product navigation via a product search query. If one or more of the search results satisfy the original intent, customers click the products to inspect further. We collect the query (`query`) and the first clicked product ID (`product_id`) from this click log. We then join these two logs on `product_id` to collect the (`qtext, atext, query`) triplets, where the (`qtext, atext`) pair will be used as an input to our model and (`query`) as the target string that the model should aim to generate in the supervised training process.

In Figure 2, one customer types a question *"are car seat bases interchangeable"* (`qtext`) in the search box, and AIA provides an answer *"Basically, with the exception of ..."* (`atext`) to this question. This customer finds the product *"Britax Gen2 infant car seat base... compatible with all ..."* (`product_id`) under the widget is relevant to the question, and clicks it to see more information. Another customer who searches for *"universal infant car seat base"* (`query`) finds that the same product *"Britax Gen2 infant car seat base... compatible with all ..."* is relevant to the query, and clicks the product (`product_id`) to add-to-cart or purchase. Since the two customers landed on the same product using two different ways of navigation, we can collate these two pieces of information into one.

The beauty of this data collection approach is that through this co-click created by customers, we know the shopping problem mentioned in the `qtext` and `atext` is summarized and resolved in `query`, which is exactly the kind of data that we need for our task.

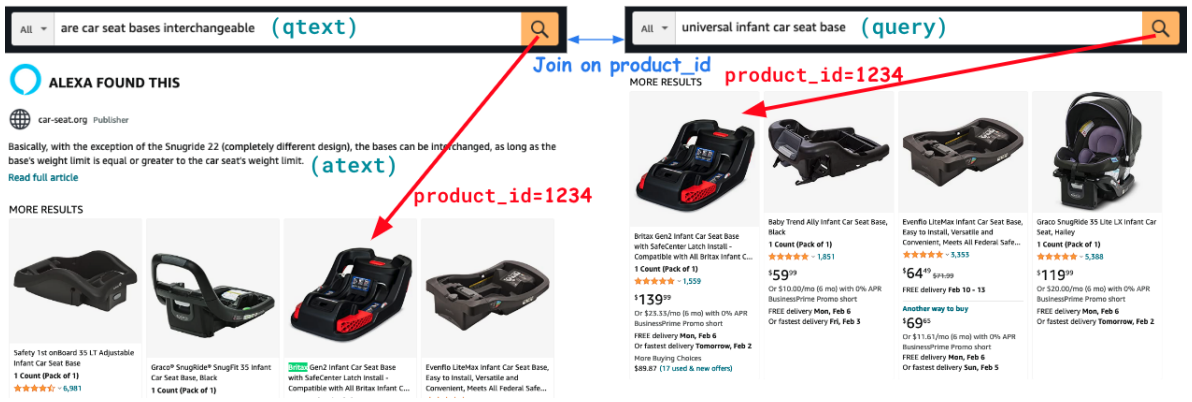

Figure 2: Product co-click between Alexa Instant Answer and Customer Search.

3.2 Data Filtering

Customer search logs ($\mathcal{S}$) consist of `session_id`, `customer_id`, `query`, `number_of_returned_products`, `product_id` of the first clicked product, and `attributed_revenue`. We apply the following filtering methods on $\mathcal{S}$ for data clean-up: First, we select queries that were issued by at least two different customers. This step is to remove queries that have typographical errors or are too personal. Second, we further filter out queries which have less than 2 or more than 6 words. This filtering logic is based on the common sense that queries that are too short (e.g., *"camera"*, *"t-shirt"*) are usually too generic, while queries that are too long (e.g., *"digital camera 2.7k 48mp compact camera"*) are too specific and narrow, and thus may yield not enough search results. Third, to ensure the widget shows enough relevant products, we filter out queries which return less than 20 products on average across multiple search sessions.

AIA logs ($\mathcal{A}$) consist of `customer_id`, `qtext`, `atext` and `product_id` of the first clicked product. We join the two logs $\mathcal{S}$ and $\mathcal{A}$ on `product_id` to create $\mathcal{S} \times \mathcal{A}$. Since there could be multiple queries which led to the click of the same product, we select up to top-k ($k \leq 100$) queries based on the `attributed_revenue` for each (`qtext, atext`) pair. Then, we measure the cosine similarity between the top-k queries and the answer text (`atext`) using all-mpnet-base-v2 [26] embedding. Queries that do not meet the threshold cosine similarity score of 0.6[1] are filtered out, and the remaining are ranked in descending order. Since the products under the AIA widget are selected based on their relevance to the question (`qtext`), the purpose of this step is to favor queries that address the additional information mentioned in the answer text (`atext`).

3.3 Data Preparation

The product co-click data collected above does not have an initial query. To prevent the model from generating queries that are redundant with the initial search query, we explore two training procedures:

[CoClick1] We extract a product name (`product_name`) from `qtext` using an industry product name extractor (*"car seat bases"* is the `product_name` from *"are car seat bases interchangeable"* in Figure 2) and use it as a proxy of an initial query. We select top-1 `query` for each (`qtext, atext`) pair based on the cosine similarity score. We prepare the training data in the following format:

Input String (I, Q, A) := (`product_name, qtext, atext`),

Target String R := `query`,

where in case `product_name` and `query` are identical, the corresponding row is discarded.

[CoClick2] We select top-2 queries (`query1` and `query2`) for each (`qtext, atext`) pair based on the cosine similarity score. We prepare the training data in the following format:

Input String (Q, A) := (`qtext, atext`),

Target String R := `query1<SEP>query2`,

where <SEP> is a special token separating the two queries. Note that `query2` might be occasionally an empty string due to the score threshold. In case both `query1` and `query2` are empty strings, the corresponding row is discarded.

4 MODEL DESCRIPTION

Our analysis on the data sets in Section 3 showed that for only 36% of the cases a target query was seen as an extractive n-gram of either `qtext` or `atext`. Therefore, rather than adopting an extractive approach, we frame this task as a sequence-to-sequence (Seq2Seq) problem. We adopt the encoder-decoder framework T5-Base, which has been shown to obtain state-of-the-art performance on several language generation tasks [25]. Furthermore, we face with a strict latency requirement for running the model online. For this particular task, the number of model parameters could not go over 300M. The biggest and the most powerful seq2seq generative model which met this requirement was the T5-Base with 223M parameters.

We fine-tune the T5 model in a supervised manner using the training data described in Section 3. Given an input text $X = (I, Q, A)$ or $X = (Q, A)$, where I, Q and A are initial query, question text and answer text, respectively, the goal is to learn a model $p(\theta)$ that can generate the target query recommendation $R = [R_1, \ldots, R_T]$ as its output, where T is the number of tokens to generate. The parameters θ are learned by maximizing the likelihood of the target query recommendation R, i.e., the learning objective is to minimize the cross-entropy loss:

$$L_{ce}(\theta) = -\sum_{t=1}^{T} \log p(R_t | R_{1:t-1}, X), \quad (1)$$

[1]Threshold was chosen based on a sample study of 200 examples.

Table 1: Rouge scores of generated queries against editor recommendations (left); Relative query acceptance rate compared to T5-CoClick2 (right).

	Rouge-1	Rouge-2	Rouge-L	Accept (%)
Extractive	45.60	22.97	44.53	-29.2
One2Seq	45.10	19.32	43.72	-31.2
One2Set	46.70	21.66	45.38	-25.2
Jurassic-2	49.01	25.87	46.46	-22.9
T5-CoClick1	50.72	25.15	48.57	-20.0
T5-CoClick2	58.66	34.89	56.43	-

where $R_{1:t}$ is the subsequence of R ranging from 1 to t.

In order to prevent the model from generating a query recommendation that is redundant with the initial query, we train the T5-Base using two differently created data sets:

T5-CoClick1. A T5-Base model trained on the [CoClick1] data described in Section 3.3. This model learns to generate one query recommendation given the initial query, question text, and answer text triplet (I, Q, A) as an input. Since the target string is by construction always different from the product name, the model indirectly learns to generate a query recommendation that does not overlap with the initial query.

T5-CoClick2. A T5-Base model trained on the [CoClick2] data. This model learns to generate up-to two query recommendations separated by the special token '<SEP>', given only the question text and answer text pair (Q, A) as an input. In case the first query recommendation overlaps with the initial query, we can suggest the second one as an alternative, unless it is empty.

In the training, we used Adam optimizer [16] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for optimization. We pre-trained the model with 20 epoch and a learning rate of 1e-4. The mini-batch size was fixed at 8. At test time, we used beam search of width 5 on all our models to generate our final predictions.

5 EVALUATION

5.1 Human Generated Data for Testing

We randomly selected 1K initial queries, question text and answer text triplets from the SQ&A logs in order to generate editorial data for offline quality evaluation. Human editors are provided with the tenets and requested to go over the following three steps: 1) provide a query recommendation that is not redundant with the initial query, specific and aligned with the Q&A; 2) test the recommended query against the current product inventory to check if the first 10 returned products are not significantly overlapping with those of the initial query, and aligned with the query intent; 3) check if the average price of the products returned by using the recommended query is not significantly lower than that of the initial query. If any of the above steps is not satisfactory, editors repeat the whole process starting from recommending another query.

5.2 Offline Evaluation

We report the evaluation of our trained T5-CoClick1 and T5-CoClick2 models on the 1K human generated test data set and compare the results against the following baselines:

i) Extractive: A baseline model which combines noun-phrases extracted from the question and answer text using three off-the-shelf tools, namely, spaCy [15], NLTK [7], and distilbert_openkp [35]. The noun-phrases are then ranked by their relevance to the question-answer pair, which is measured based on cosine similarity:

$$\text{cos\_sim(NP,Q\&A)} = \alpha \frac{v_{NP}^\top v_q}{\|v_{NP}\|\|v_q\|} + (1-\alpha)\frac{v_{NP}^\top v_a}{\|v_{NP}\|\|v_a\|}, \quad (2)$$

where $\alpha \in [0, 1]$ is a hyperparameter, and v_{NP}, v_q, v_a are the embedding vectors of the noun-phrase (NP), question text, and answer text, respectively, obtained using all-mpnet-base-v2 [26]. We select the first ranked noun-phrase as a query recommendation. If the first one is redundant with the initial query, the second ranked one is selected.

ii) One2Seq and One2Set: One2Seq is a training paradigm that can generate a sequence of multiple keyphrases with dynamic numbers as well as considering the dependency between keyphrases [38]. Each target string is a sequence of keyphrases that are concatenated with a delimiter <SEP> and a terminator <EOS>. One2Set is a state-of-the-art sequence-to-sequence keyphrase generation method [11]. Since the keyphrases are an unordered set rather than an ordered sequence, One2Set [37] proposes to generate a set of keyphrases in parallel rather than imposing a predefined order by concatenating target keyphrases. One2Seq and One2Set are trained on the [CoColick2] dataset. At inference time, we select the query with highest Rouge scores for the final query recommendation.

iii) Jurassic-2 (Grande): The large language model (LLM) with 17B parameters recently developed by AI21 studio [2]. We prompted the model with 3-shot in-context examples that are selected randomly from a validation set of size 20, following the previous works [32, 33]. Note that the comparison against an LLM is for offline evaluation only, as any model bigger than 300M parameters does not meet our latency requirement.

Following the literature in natural language generation, we use Rouge scores [21] after stemming against the human generated query as an automatic evaluation metric. As Rouge scores are often too strict and there might be more than one high-quality query recommendation for each case, we also conduct an editorial evaluation of the generated queries and report the percentage of acceptable cases. A query recommendation is labeled as Accept if it successfully passes the 3-step procedure described in Section 5.1.

Table 1 shows the Rouge scores and the editorial evaluation results on the test data set. We can observe the manual evaluation results are aligned with the Rouge scores, and the percentage of acceptable queries generated by T5-CoClick2 is much higher than the other models. Many of the queries generated by `Jurassic-2 Grande` were hallucinatory and included words that do not appear in the question or answer text (See Table 2 for examples). A similar pattern was often observed for T5-CoClick1, which we argue that T5-CoClick1 was forced to use other words than the initial query and diverted from the context in the training process. All results in Table 1 are statistically significant when measured using the two-tailed paired t-test at a 99% confidence level. Bonferroni correction [23] was applied for comparing against all the baselines with respect to T5-CoClick2. Table 2 shows some example queries generated by all models. In the third example, T5-CoClick2 is able to select the

Table 2: Examples of generated query recommendations. Boldfaced is the final query recommendation.

Initial Query	ice scraper	laundry baskets	grass seed
Q&A	Q: What are different types of ice scrapers? A: The main types of ice scrapers include traditional short-handle scrapers, ice scraper mitts, and heated scrapers.	Q: What is a pop up hamper? A: Pop-up hampers are collapsible laundry hampers that are often made of mesh, making them more convenient and portable.	Q: What's the difference between grass seed and sod? A: Grass seed needs to be planted, spouted, and grown. Alternatively, sod is mature grass that is transplanted into a yard.
Human	ice scraper mitt	laundry hamper collapsible	grass sod
Extractive	ice scraper	hampers	grass seed
One2Seq	**ice mitts**;heated ice mitts;scraper	**pop up laundry**;pop up laundry	grass seeds for lawn;**sod grass seed**
One2Set	scrapers ice maker;heated ice mitts; scrapers;**ice mitts**;ice mitts for home	**collapsible laundry hamper**; pop up laundry hamper;pop up hamper	**sod grass seed**;cold grass seed
Jurassic-2	ice scraper for windshield	laundry basket with wheels	grass seed for shade
T5-CoClick1	ice scraper tools	laundry room hampers	grass seed for lawn
T5-CoClick2	**ice scraper mitt**<SEP>heated ice scraper	**collapsible laundry hamper**<SEP>pop up laundry hamper	grass seed<SEP>**grass sod**

Table 3: Online test metrics of the `ML` bucket compared to `Control` and `Human`.

	`Control`	`Human`
Overall CTR	+2.50%	-0.20%
Query-level CTR Win Rate	+6.46%	+17.63%

second generated query *"grass sod"* as the first one is redundant with the initial query.

5.3 Online A/B/C Test

We have deployed our best model T5-CoClick2 on an e-commerce website and performed A/B/C testing. The experiment was configured as `Control`/`ML`/`Human` bucket which renders product search results under each Q&A based on the initial query, T5-CoClick2 generated query recommendation and human generated query recommendation, respectively. Our hypothesis for the experiment was that using T5-CoClick2 generated query recommendations would provide the benefit of online serving and scaling, while having positive impact on engagement when compared to `Control`, which by default renders the same search page that a customer is already on.

We designated a set of initial queries and their SQ&As to collect human generated query recommendations using the 3-step procedure mentioned in Section 5.1. The online traffic on this set of initial queries gets split into 1:1:1 ratio among the three buckets. In order to get statistically significant results, we observed the online experiment for 14 days. Since this is an industrial work, we only reveal relative metrics, improvement on overall CTR and query-level CTR win rate. The query-level win rate refers to which treatment 'wins' in terms of CTR when compared for each initial query.

Table 3 shows the differences in CTR and the query-level CTR win rate. We observe that `ML` CTR is 2.50% higher than that of `Control` and only slightly lower than `Human`. The query-level CTR win rate of `ML` is 17.63% higher than that of `Human`, which indicates that our proposed model generates 'more helpful' product suggestions than humans. This online experiment validates that our model brings more customer engagement and our data generation method effectively reflects actual customer behavior.

6 RELATED WORK

Recommending search queries is a popular problem in Web search. Candidate queries are extracted from a query log [8–10, 17, 19] or retrieved directly from the document text [8, 19, 28]. In the e-commerce domain, [14] performed analysis on its major differences between query recommendation in the web domain and e-commerce domain. Another related work used customer reviews for query generation [20].

Research work that is different than ours is lexical signatures and query expansion. Lexical signature aims to find keywords from a Web document which is sufficient for finding the document later, even if its URL has changed [4, 24]. Query expansion is a long studied approach to address the vocabulary mismatch between queries and relevant documents [5]. Common techniques are based on relevance feedback provided for documents [27], passages [3, 36], terms [13, 31] and entities [29].

The problem of suggesting search tokens from a product Q&A pair is broadly related to the task of summarizing a document into keyphrases. Previous works that use deep learning techniques include [1, 6, 12, 18, 22]. A pointed out by [30, 34], there is no work on domain specific applications. To the best of our knowledge there is little work in the context of e-commerce which tries to help the customer information need, especially by recommending a follow-up search query recommendation.

7 CONCLUSIONS

We studied the problem of recommending product queries for enriching the SQ&A widget and explained why the problem itself is a unique challenge. We proposed to collect a large-scale training dataset using the co-click information between AIA and user search log, then filter and prepare the data accordingly in order to guide our model into generating desirable query recommendations. We trained a neural generative model T5 on the collected data set which generates a query recommendation given a Q&A pair as an input within the strict latency requirement. Offline and online evaluations attest to the clear merits of using our trained model, by comparing against state-of-the-art keyphrase generation models and a prompt based LLM, and by running our model on a live customer traffic.

REFERENCES

[1] Wasi Ahmad, Xiao Bai, Soomin Lee, and Kai-Wei Chang. 2021. Select, Extract and Generate: Neural Keyphrase Generation with Layer-wise Coverage Attention. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Online, 1389–1404. https://doi.org/10.18653/v1/2021.acl-long.111

[2] AI21-Studio. 2023. Jurassic-2 Models. https://docs.ai21.com/docs/jurassic-2-models.

[3] James Allan. 1995. Relevance Feedback With Too Much Data. In *Proc. of SIGIR*. ACM Press, 337–343.

[4] Omar Alonso, Sushma Bannur, Kartikay Khandelwal, and Shankar Kalyanaraman. 2015. The World Conversation: Web Page Metadata Generation From Social Sources. In *Proc. of WWW*. 385–395.

[5] Hiteshwar Kumar Azad and Akshay Deepak. 2019. Query expansion techniques for information retrieval: A survey. *Inf. Process. Manag.* 56, 5 (2019), 1698–1735.

[6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[7] Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

[8] Ilaria Bordino, Gianmarco De Francisci Morales, Ingmar Weber, and Francesco Bonchi. 2013. From Machu picchu to "Rafting the Urubamba River": Anticipating Information Needs via the Entity-Query Graph. In *Proc. of WSDM*. 275–284.

[9] David Carmel, Yaroslav Fyodorov, Saar Kuzi, Avihai Mejer, Fiana Raiber, and Elad Rainshmidt. 2019. Enriching News Articles with Related Search Queries. In *The World Wide Web Conference*. 162–172.

[10] Zhicong Cheng, Bin Gao, and Tie-Yan Liu. 2010. Actively Predicting Diverse Search Intent from User Browsing Behaviors. In *Proc. of WWW*. 221–230.

[11] K.-W. Chang D. Wu, W. U. Ahmad. 2022. Pre-trained Language Models for Keyphrase Generation: A Thorough Empirical Study. arXiv:2212.10233 [cs.CL]

[12] J. Gu et al. 2016. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In *Proc. of ACL*. Association for Computational Linguistics, 1631–1640.

[13] Donna Harman. 1988. Towards Interactive Query Expansion. In *Proc. of SIGIR*. ACM, 321–331.

[14] Mohammad Al Hasan, Nish Parikh, Gyanit Singh, and Neel Sundaresan. 2011. Query Suggestion for E-Commerce Sites. In *Proc. of WSDM*. 765–774.

[15] Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. (2017). To appear.

[16] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR*, Yoshua Bengio and Yann LeCun (Eds.).

[17] Weize Kong, Rui Li, Jie Luo, Aston Zhang, Yi Chang, and James Allan. 2015. Predicting Search Intent Based on Pre-Search Context. In *Proc. of SIGIR*. 503–512.

[18] Mayank Kulkarni, Debanjan Mahata, Ravneet Arora, and Rajarshi Bhowmik. 2022. Learning Rich Representation of Keyphrases from Text. In *Findings of the Association for Computational Linguistics: NAACL 2022*. Association for Computational Linguistics, 891–906.

[19] Chia-Jung Lee and W. Bruce Croft. 2012. Generating Queries from User-Selected Text. In *Proc. of IIIX*. 100–109.

[20] Yen-Chieh Lien, Rongting Zhang, Max Harper, Vanessa Murdock, and Chia-Jung Lee. 2022. Leveraging customer reviews for e-commerce query generation. In *Proc. of ECIR 2022*.

[21] Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*. ACL, Barcelona, Spain, 74–81.

[22] Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep Keyphrase Generation. In *Proc. of ACL*. Association for Computational Linguistics, 582–592.

[23] Ron Mittelhammer, George Judge, and Douglas Miller. 2002. *Econometric Foundation*. Vol. 97.

[24] Seung-Taek Park, David M. Pennock, C. Lee Giles, and Robert Krovetz. 2004. Analysis of Lexical Signatures for Improving Information Persistence on the World Wide Web. *ACM Trans. Inf. Syst.* 22, 4 (oct 2004), 540–572.

[25] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *CoRR* abs/1910.10683 (2019).

[26] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. https://arxiv.org/abs/1908.10084

[27] Gerard Salton and Chris Buckley. 1990. Improving retrieval performance by relevance feedback. *J. Am. Soc. Inf. Sci.* 41, 4 (1990), 288–297.

[28] Eilon Sheetrit, Yaroslav Fyodorov, Fiana Raiber, and Oren Kurland. 2021. Recommending Search Queries in Documents Using Inter N-Gram Similarities. In *Proc. of ICTIR*. 211–220.

[29] Eilon Sheetrit, Fiana Raiber, and Oren Kurland. 2023. Entity-Based Relevance Feedback for Document Retrieval. In *Proc. of ICTIR*. 177–187.

[30] Mingyang Song, Yi Feng, and Liping Jing. 2023. A Survey on Recent Advances in Keyphrase Extraction from Pre-trained Language Models. 2153–2164.

[31] Amanda Spink. 1994. Term Relevance Feedback and Query Expansion: Relation to Design. In *Proc. of SIGIR*. 81–90.

[32] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. arXiv:2203.11171 [cs.CL]

[33] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, Quoc Le, and Denny Zhou. 2022. Chain of Thought Prompting Elicits Reasoning in Large Language Models. *CoRR* abs/2201.11903 (2022).

[34] Di Wu, Wasi Uddin Ahmad, and Kai-Wei Chang. 2022. Pre-trained Language Models for Keyphrase Generation: A Thorough Empirical Study. arXiv:2212.10233 [cs.CL]

[35] Lee Xiong, Chuan Hu, Chenyan Xiong, Daniel Campos, and Arnold Overwijk. 2019. Open Domain Web Keyphrase Extraction Beyond Language Modeling. In *Proc. of EMNLP*. 5175–5184.

[36] Jinxi Xu and W. Bruce Croft. 1996. Query Expansion Using Local and Global Document Analysis. In *Proc. of SIGIR*. 4–11.

[37] Jiacheng Ye, Tao Gui, Yichao Luo, Yige Xu, and Qi Zhang. 2021. One2Set: Generating Diverse Keyphrases as a Set. In *Proc. of ACL*. 4598–4608.

[38] Xingdi Yuan, Tong Wang, Rui Meng, Khushboo Thaker, Peter Brusilovsky, Daqing He, and Adam Trischler. 2020. One Size Does Not Fit All: Generating and Evaluating Variable Number of Keyphrases. In *Proc. of ACL*. 7961–7975.