CADENCE: Offline Category Constrained and Diverse Query Generation for E-commerce Autosuggest

Abhinav Anand
Flipkart Internet Pvt. Ltd.
Bengaluru, India
abhinav.a@flipkart.com

Surender Kumar
Flipkart Internet Pvt. Ltd.
Bengaluru, India
surender.k@flipkart.com

Nandeesh Kumar*
Indian Institute of Science
Bengaluru, India
nandeeshk@iisc.ac.in

Samir Shah
Flipkart Internet Pvt. Ltd.
Bengaluru, India
samir.shah@flipkart.com

ABSTRACT

Query AutoComplete (QAC) or AutoSuggest is the first place of user interaction with an e-commerce search engine. It is critical for the QAC system to suggest relevant and well-formed queries for multiple possible user intents. Suggesting only the historical user queries fails in the case of infrequent or new prefixes. Much of the recent works generate synthetic candidates using models trained on user queries and thus have these issues: a) cold start problem as new products in the catalogue fail to get visibility due to lack of representation in user queries b) poor quality of generated candidates due to concept drift and c) low diversity/coverage of attributes such as brand, color & other facets in generated candidates.

In this paper, we propose an offline neural query generation framework - CADENCE - to address these challenges by a) using both user queries and noisy product titles to train two separate neural language models using self-attention memory networks, b) adding category constraints during the training and query generation process to prevent concept drift c) implementing customized dynamic beam search to generate more diverse candidates for a given prefix. Besides solving for cold start and rare/unseen prefix coverage, CADENCE also increases the coverage of the existing query prefixes through a higher number of relevant and diverse query suggestions. We generated ~700K new offline queries, which have resulted in significant improvement in recall, reduction in product cold start, and increased coverage of attributes. Online A/B tests also show a significant impact on QAC usage, downstream search click-through rates, and product conversion.

CCS CONCEPTS

• **Applied computing** → **Online shopping**; • **Computing methodologies** → **Natural language generation**; **Neural networks**.

*Work done as an intern at Flipkart Internet Pvt. Ltd.

KDD '23, August 6–10, 2023, Long Beach, CA, USA

ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599787

KEYWORDS

e-commerce search, generative autosuggest, autocomplete, neural language model, diverse beam search, constrained DNN

ACM Reference Format:
Abhinav Anand, Surender Kumar, Nandeesh Kumar, and Samir Shah. 2023. CADENCE: Offline Category Constrained and Diverse Query Generation for E-commerce Autosuggest. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3580305.3599787

1 INTRODUCTION

As a user starts typing a search query, a Query AutoComplete(QAC) or AutoSuggest(AS) system kicks in to help her select a relevant query from a drop-down list of several matching query suggestions. In an e-commerce setting, it enhances user experience by a) reducing typing effort, which helps in reducing the user-induced spell mistakes and helps in quick product discovery [4, 5, 34], b) shaping the user's perception of the product catalogue by making her aware of different categories and products, and c) helping a user to narrow her intent by adding more facets: 'Nike shoes' → 'Nike shoes *for basketball*'. Table 1 demonstrates the impact of a QAC system on the overall clicks and cart adds of our e-commerce search engine.

In a traditional QAC system, a query's frequency and click-through rate in the last 'k' days is used to mine popular queries from the search query logs. Such popular queries also called organic candidates are stored either in an inverted index (e.g., Lucene [1]) or a trie-based data structure. These organic candidates are then matched and served as QAC suggestions in real-time to a user as she begins to type her query.

Search queries follow a skewed distribution where more than one-third of the search volume of the queries do not repeat or repeat very infrequently [2, 15, 33]. Hence, organic candidates are insufficient as the QAC system cannot match all user search prefixes, which contain a lot of tail prefixes. Statistical language model [17] based approaches [23, 32] overcome this problem by mining popular n-gram phrases from the query logs. To generate suggestions, phrases that match the last few words in the user prefix are appended. Since statistical methods fail to utilize the complete contextual information, recent works have used RNN-based sequential language models as they have shown better results in sequence

	AS used	AS shown but not used	AS not shown
% Searches	77.30	13.18	9.52
Search CTR*	1	0.88	0.81
Cart Adds *	1	0.76	0.73

Table 1: Normalized* performance with and without AutoSuggest (AS)

modeling, especially for short sequences [22]. Many neural network-based methods have been applied in the QAC systems [18, 25, 31] where a language model is trained to measure the probability of a sequence of tokens/characters/subwords. However, the above candidate generation approaches fall short in several key aspects:

- **Product cold start** due to lack of visibility of new products for a given prefix. In an e-commerce setting, there are many tokens in the product catalogue that are absent or infrequent in the historical search query logs. While products can be discovered through regular search, a QAC system provides indirect discoverability of products in the catalogue. For example, the new model of the iPhone recently added to the e-commerce platform will take time to surface in QAC based on historical queries. Thus, the lack of suggestions in QAC can give a wrong first impression of the product being unavailable and hurt customer engagement.
- **Concept drift** during query generation results in poor quality queries being shown. For example, queries 'water bottle washing brush' and 'washing machine' belong to 'General Merchandise' and 'Home & Kitchen' categories, respectively. However, the query 'water bottle washing machine' is generated with very high confidence for the' water bottle washing' prefix, leading to category drift. Such malformed queries need to be pruned out as much as possible in the generation phase itself.
- **Low diversity** of generated candidates. Showing diverse suggestions helps in shaping a user's experience of the catalogue by making her aware of different products. Current methods use popular n-grams/top-k completions via phrase-based completion or vanilla beam search, which result in low diversity and coverage of attributes such as brand, color and other facets in the generated candidates.

In our work, to address these problems we propose a candidate generation framework called **CADENCE** (**C**onstrained **A**nd **D**iver**s**E **N**eural **C**andidate Gen**E**rator). Preventing concept drift by enforcing category constraints during training and generation is one of the contributions of our work. When a user searches for products, she already has a category in mind. Thus, honoring this category constraint in the sequence or n-gram models helps generate more relevant and quality suggestions. Another contribution of our work is developing a product Catalogue Language Model (besides a Query Language Model) from the products' short and noisy title text to address the cold start problem. In web-search, this has been addressed by learning a language model from the web pages or documents [6, 21]. However, unlike web-search, documents in an e-commerce catalogue are structured as key-value pairs of attributes (*brand: Nike, type:running, category: shoes ....*); hence are inherently noisy. The product description field is verbose and not always available, while

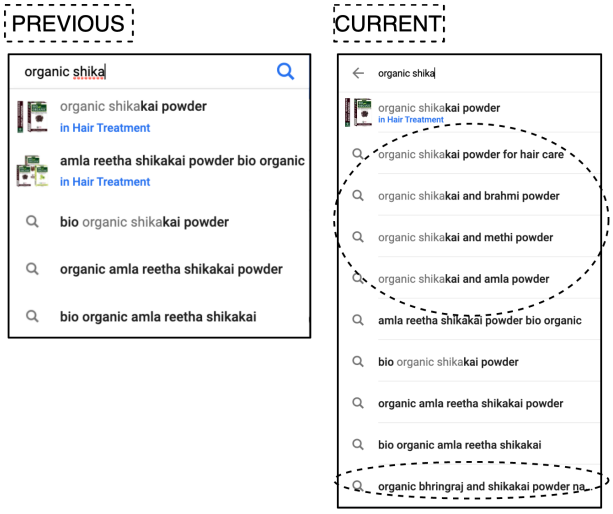

Figure 1: Increased recall through improved QAC model

the title field is short and concise. Building a Catalogue Language Model helps address the cold-start problem and shape users' perceptions of available product selection. The third contribution is to bring diversity to our query candidates. Category constraints help in increasing diversity by reducing popular category dominance. To further increase diversity, we implement a custom dynamic beam search. Figure 1 shows an example of increased recall in our results in the deployed production system where CADENCE has been in use for a year.

In Section 2, we highlight related work. In Section 3, we define some basic terminologies. In Section 4, we describe various stages of CADENCE. Section 5 presents the experiments and our results. Finally, we conclude in Section 6 with a discussion of future work.

2 RELATED WORK

The field of QAC has evolved beyond suggesting the historical user search queries. Mining the user queries at sub-query/n-gram level with Statistical Language Modelling (SLM) [17] based approaches [6, 23, 32] have improved the quality of query suggestions. Modified language models [8] leverage the large-scale search data to improve upon the traditional language models by better back-off handling of the sparse n-grams. Neural language models [22] improve over the SLM models particularly in the sequence learning tasks. Attention [3], self-attention [29] learn better the intra-sequence semantics. Thus, neural methods have become the default and state of the art for AS task [18, 25, 31]. Large pre-trained models like BERT [12] and GPT [26] learns deeper relationship between the query tokens and have been used for text completion/generation tasks.

Traditional beam search is known to generate less diverse suggestions. This has been addressed by dividing beams into groups of constant width and then adding a dissimilarity component between the groups [30].

To address the cold-start in QAC, candidates are generated directly from documents by mining popular n-grams from documents [6] or by constructing a statistical language model from documents that match a user's prefix [21]. E-commerce queries

are transactional [27] and product-specific unlike the navigational and informational queries of web-search [9]. Also the e-commerce catalogue product documents i.e. product description & title are inherently noisy and pose learning challenges. Text summarization [14, 20, 35] has been used for refining the product titles for smaller product catalogues using supervised neural network-based models. However, getting labels on a large scale of hundreds of millions of products for supervised learning is infeasible and hence we develop an self-supervised solution.

The mentioned QAC approaches neither make any attempt to prevent intra-query category drift nor do they handle the diversity of candidates in the autosuggestions. They also suffer from poor coverage of less searched products. We address cold-start through a catalogue language model after extractive summarization of the short text titles in an unsupervised manner.

3 DEFINITIONS

We define and use the following notations/terminology.

- **Vertical** - It is a collection of products based on similar catalogue attributes like jeans_length, print_type, etc. E.g., 'shoes' and 'jeans' are two different verticals.
- **Category** - It is a collection/parent of related verticals. For example, the 'Lifestyle' category is a parent of verticals 'jeans' and 'shoes' while the 'Electronics' category is a parent of verticals 'TV', 'Speakers' etc.
- **QAC Usage** - Fraction of all the search queries where user chose a QAC query suggestion.
- **Search CTR** - Search CTR (Click Through Rate) is defined as a ratio of the number of clicks on a search results page to the total number of search requests.
- **CABN** - Cart Add(CA) & Buy Now(BN) events.

4 PROPOSED FRAMEWORK - CADENCE

CADENCE is composed of multiple stages - Data generation stage, Modelling stage, Candidate generation stage, and finally, Ingestion (into the QAC index) stage - and is described below in detail. Fig.2 illustrates our proposed framework.

4.1 Data Generation

In the Data Generation stage, we collect and process historical query data and noisy product titles to train separate Query LM and Catalogue LM. Details are as follows:

4.1.1 ***Query Data*** : For training the **Query language model**, unique user queries from the past 60 days were collected, and queries appearing less than five times were filtered out. These queries were then pre-processed with the following actions:

- *Spell Correction*- Each query was first passed through a high confidence spell correction step.
- *Stopword Removal* - We removed stopwords and prepositions from the queries except for a few joiners like 'for', 'to', 'with', 'below', 'under' etc. These are retained as they help in better query understanding. Offline query analysis showed that queries with joiners are ∼ 16% more and with CTR higher by ∼ 1%.
- *Singularization* - We further singularize our queries to reduce vocab size, data sparsity, and noise. For example, queries 'washing machines' and 'washing machine' indicate the same user intent.
- *Unit Correction* - Queries like 'refrigerator 192l', 'refrigerator 192 litres' or 'refrigerator 192 litre', all refer to the same refrigerator capacity. To handle this, we created a category domain specific dictionary mapping that canonicalizes multiple such forms of units to a single one using regex. Sample dictionary is of the form: 'watt':['w', 'watts', 'watt'], 'kg':['kgs', 'kg', 'kilo', 'kilogram'].

4.1.2 ***Catalogue Data*** : In an E-commerce catalogue, product documents are structured as a list of key-value attributes, with only product title and product description being longer texts. Since the product description field is verbose and not mandatory to be filled by sellers, it contains noisy and irrelevant information. The title field is mandatory but is short and concise with some noise. We use product title text to learn a catalogue language model, but we first need to cleanse titles to remove noise from them. In some cases with smaller catalogues [14, 20, 35], text summarization has been used for refining the product titles using supervised neural network-based models. For a scale of 100s of millions of catalogue products, getting labels is expensive and time-consuming. Hence we develop an unsupervised method to cleanse the product title texts. We partly make use of the previous work [24] along with two statistical measures-based methods to cleanse our short title and learn the **Catalogue language model**. Details of principled cleansing of the short text titles are as follows:

a) Title Preprocessing - Unlike user queries, product titles do not adhere to a syntactic structure. Since sellers cannot upload more than one title and have restricted space, they add additional information using parenthesis or delimiters. Table 2 shows a few such titles and also outlines the basic pre-processing rules.

b) Extractive Summarization - After initial pre-processing steps, we remove irrelevant tokens that do not contribute to key information content in the title. For example, in a title like 'amalth boswellia serrata extract top selling veg capsules,' tokens 'top selling' can be removed without affecting the information content of the title. We use entropy and PMI (pointwise mutual information) as two statistical measures to identify the non-informative tokens. To compute these measures, all titles in each vertical are grouped to create a per vertical document set. Table 4 shows document matrix representation for a vertical. Filtering methods based on the above statistical measures are used for extractive summarization. Details are as follows:

(1) **Entropy based filtering** - Entropy-based filtering assumes tokens with less entropy carry less information and are pruned out [19]. The entropy score for each token w using its count c is calculated as below:

$$Entropy(w_j) = \frac{-\sum p_{ji} * log p_{ji}}{n}; p_{ji} = \frac{c_{j,i}}{\sum_{k=1}^{n} c_{j,k}}$$

Using log-likelihood on the held-out validation set, the mean of each vertical's entropy score distribution was chosen as an optimal threshold to filter out non-informative tokens.

(2) **PMI based filtering** - After entropy based filtering, we further remove tokens that are incompatible with other tokens in the

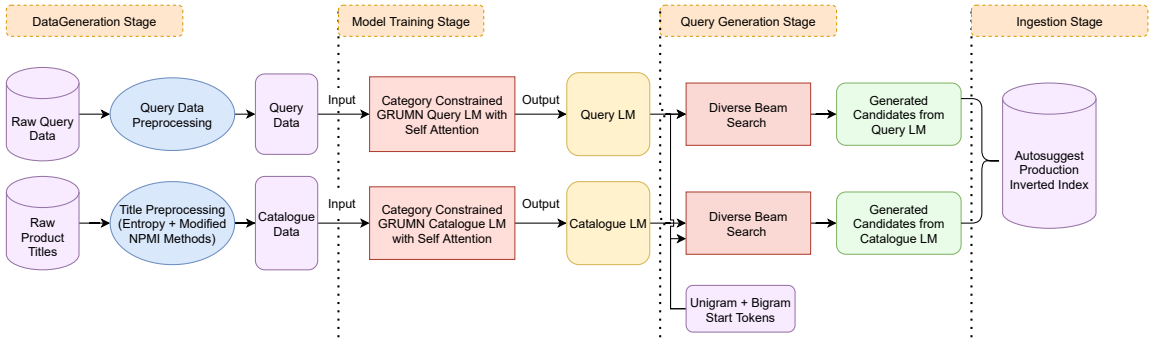

Figure 2: CADENCE Framework

Raw Title	Processing Rule	Processed Title
N2s Back Cover for Mi Redmi Y1 Lite Back Case & BackCover (Silicon, Printed, designer, Soft)	Parenthesis can contain relevant tokens (product specification). Create new titles by adding these tokens inside parenthesis to title part.	Silicon N2s Back Cover for Mi Redmi Y1 Lite Back Case & BackCover Printed N2s Back Cover for Mi Redmi Y1 Lite Back Case & Back Cover
IHC 9W BATTERY holder without cover Heavy Duty PANTHERZ 4Watts Long Life Battery	Standardize different unit formats in the title.	IHC 9 watt BATTERY holder without cover Heavy Duty PANTHERZ 4 watt Long Life Battery
TQH Solid Men Round Neck White, Blue T-Shirt	Multiple entities in the title is separated by delimiters like ',' , '|' , '+' etc. Split these delimiters to create multiple titles.	TQH Solid Men Round Neck White T-Shirt TQH Solid Men Round Neck Blue T-Shirt

Table 2: Product Title preprocessing

Processed Title	Entropy Filtered Title	PMI Filtered Title
bueno different flavor instant nail polish remover wipes	bueno flavor instant nail polish remover wipes	bueno instant nail polish remover wipes
greenspa active charcoal quick shine scrubowash face wash	greenspa active charcoal shine face wash	greenspa active charcoal face wash
scrazy stainless steel coconut scraper ex-584 metal kitchen tool set	scrazy stainless steel coconut scraper metal kitchen tool set	scrazy stainless steel coconut scraper kitchen tool set
l'oreal paris dermo expertise pearl perfect relighting whitening facial foam	l'oreal paris dermo pearl perfect whitening facial foam	l'oreal paris pearl whitening facial foam

Table 3: Product Title Extractive Summarization: Titles obtained after basic preprocessing –> entropy filtering –> PMI filtering stages. Titles obtained after the PMI filtering stage are used for training Catalogue LM.

	$title_1$	$title_2$	...	$title_i$	...	$title_n$
...	...	..	...	...	...	...
w_{j-1}	$c_{j-1,1}$	$c_{j-1,2}$	...	$c_{j-1,i}$	...	$c_{j-1,n}$
w_j	$c_{j,1}$	$c_{j,2}$	...	$c_{j,i}$	...	$c_{j,n}$
w_{j+1}	$c_{j+1,1}$	$c_{j+1,2}$	...	$c_{j+1,i}$	...	$c_{j+1,n}$
...	...	..	...	...	...	...

Table 4: Document matrix representation for a vertical

title. We use PMI, which measures the degree of association between a token and its neighbors. We used a modified version of normalized PMI, [7] as normalized PMI is dominated by bigrams which have low occurrence and high association.

$$modified - npmi = \frac{\frac{p(w_1,w_2)}{p(w_1)p(w_2)}}{-logp(w_1,w_2)} + logp(w_1,w_2)$$

Bi-grams are created using skip-gram i.e for token sequence $t1, t2, t3, t4, t5$, for a window size 2 and token $t3$ following bigrams are created $(t1, t3)$, $(t2, t3)$, $(t4, t3)$, $(t5, t3)$. Their counts are discounted with their distance (discounting factor = 0.4). Association between bigram is directly proportional to PMI score. Using log-likelihood on the held-out validation set, the mean of each vertical's PMI score distribution was chosen as an optimal threshold to filter out non-informative and incoherent tokens.

Table 3 lists a few examples of extractive summarization.

4.2 Models

Given a starting prefix token w_1, we generate a complete query token by token until the end-of-sentence(EOS) token is encountered or a pre-defined maximum query length is reached. Given a prefix word sequence $w_1, w_2, .., w_{m-1}$ for a sequence length m, word vocabulary V a language model (LM) is learnt such that:

$$w_m = argmax_{w \in V} P(w|w_{m-1}, ...w_{m-i+1})$$

Neural language models (NLM) are the latest state of the art models that perform well for language modeling [22] and QAC tasks [18, 25, 31]. In the CADENCE framework, we create a deep constrained GRU-MN (Gated Recurrent Unit Memory Network) based RNN neural network by adding category constraints to each layer.

4.2.1 *Constrained Neural Language Model*

Since GRUs have shown comparable performance as LSTM in sequence modeling tasks [28], we have developed a word-level GRU based deep neural model.

Transformer[29] based models like GPT-2 [26] and BERT [12] truly excel in capturing context in longer text sequences. Given the short nature of user queries (avg. length is 4 tokens), we have brought in the goodness of Transformers (self attention) by using memory units in our modified GRU Cell. Thus, these modified GRU models are able to capture context with a significantly lesser number of parameters without overfitting while enforcing better autoregressive sequence dependency. Also the smaller inference time per token (the model is invoked multiple times as the user incrementally types in a search query) of our architecture allows us to trivially use this model in the online user path in the future. Category constraints are enforced by passing category as side information at each layer. Self-attention is incorporated by modifying the GRU cell(GRU-MN) [10]. Fig. 3 illustrates our model architecture and details of category constraints and cell level self-attention are described below.

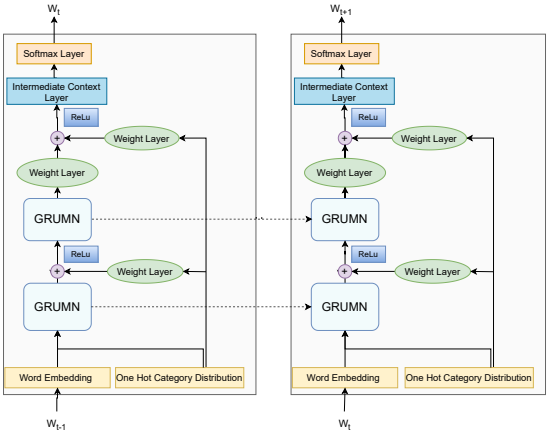

Figure 3: Category constrained model architecture

(1) **Enforcing Category Constraints -** To prevent the category drift as well as popular category dominance within a query, we predict the next token conditioned on not only the previous context but also the category i.e.

$$w = argmax_{w \in V} P(w_t|h_t, \textbf{category})$$

We have the query category available in our training dataset from an in-house category classifier. Category constraints are enforced with an n-dimensional (n = number of categories) one-hot encoded vector.

Input word at time step t(w_t) is passed through an embedding layer, to get corresponding word embedding e_t. We denote c_t as (one hot) category distribution vector at time step t. $[e_t, c_t]$ is passed through a multi-layer GRU-MN. Let h' be a single hidden layer which we refer to as intermediate context layer. We pass category information c_t along with the output of final hidden layer h_t through a trainable weight layer, followed by $tanh$ activation and projection to a required dimension. i.e,

$$h' = tanh(W_{hh'} h_t + W_{ch'} c_t + b)$$

$$x_{t+1} \sim softmax(W_{h'o} h' + b(o))$$

where $W_{h'o}, b(o)$ are the weights and biases connecting h' to output(o) layer. $W_{hh'}, W_{ch'}$ are the weights connecting h_t to h' and c_t to h' respectively.

Further, to enforce category constraints consistently across the layers of the network, c_t is passed as input at all GRU as well as the intermediate context layers. If h_t^{l-1} is the output from GRU unit at layer $l-1$, input x_t^l to GRU unit at layer l is:

$$x_t^l = relu(Wc_t + h_t^{l-1})$$

(2) **Cell Level Self Attention -** The dynamics of a standard GRU [11] cell is controlled by an input vector(x_t), update gate (z_t) and reset gate (r_t), and a hidden state (h_t) which are computed as:

$$z_t = \sigma(x_t U_z + h_{t-1} W_z + b_z)$$

$$r_t = \sigma(x_t U_r + h_{t-1} W_r + b_r)$$

$$o_t = tanh(x_t U_o + r_t \odot h_{t-1} W_o + b_o)$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot o_t$$

where h_{t-1} is hidden state from previous time step, σ is the sigmoid function and $\odot$ is the element-wise product. U_z, U_r, U_o, W_z, W_r, W_o are the weight terms with corresponding dimensions.

To enhance model's ability to capture variable context and learn relations between tokens, we use GRU Memory Network (**GRU-MN**) with attention[10]. Further, having the attention unit inside the cell provides it more flexibility than the standard attention unit. Each GRU cell has memory tape referred to as hidden layer tape (H_t^l). For a layer l at time t, let $\tilde{h}_{t-1}^l$ be the aggregated hidden unit, x_t^l be the input and $H_{t-1}^l = (h_1^l, .., h_{t-1}^l)$ be the hidden layer tape. Then attention for each unit in hidden layer tape is computed as:

$$a_{i,t}^l = v^T tanh(W_h h_i^l + W_x x_t^l + W_{\tilde{h}} \tilde{h}_{t-1}^l)$$

$$s_{i,t}^l = softmax(a_{i,t}^l)$$

aggregated hidden unit at time t ($\tilde{h}_t^l$) is computed as:

$$\tilde{h}_t^l = \sum_{i=1}^{t-1} s_{i,t}^l h_i^l$$

$\tilde{h}_t^l$ is then used to compute GRU gates and new hidden state(h_{t+1}^l). $W_h, W_x, W_{\tilde{h}}$ and v are the new weight terms of the network. Fig. 4 shows the architecture of a GRU-MN cell

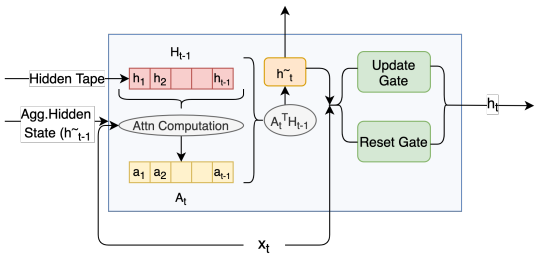

Figure 4: GRU-MN cell with attention architecture

4.3 Candidate Generation

Since we generate our suggestions offline from a set of starting tokens, we always have categories of a token available. Beam search is a greedy approach to generate tokens at every time step of a sequence model and is known to produce outputs that are not very diverse [30]. We developed a custom beam search to keep the initial diversity of tokens very high and then greedily build the sequence. To avoid generating very long query suggestions (due to poor downstream query understanding), we dynamically reduce the diversity among partial suggestions as we progress along the length of suggestions. Algorithm 1 describes our method.

At every step, we partition suggestions into groups and perform a beam search within each group. However unlike [30], we don't add any dissimilarity measure among groups because of an implicit diversification introduced in the first two levels. Also unlike [30], the number of groups changes in our case along the length of query.

We extract start tokens (unigrams and bi-grams) and their corresponding category from user queries (non-overlapping with training set). For each start token, the category is fed as side information at each time step to generate candidates.

Index ingestion: Before ingesting the candidates generated from query & catalogue LMs into the AS serving index, non-performing queries that lead to null search are filtered out by hitting our internal search API.

5 EXPERIMENTS AND DISCUSSION

This section describes the dataset, training procedure, evaluation metrics, and offline and online A/B results.

5.1 Dataset Details

All offline results are evaluated on logged user queries and catalogue product tiles on our e-commerce platform. Our query dataset contains unique user queries over two months. There are 42 categories and ~4000 verticals in our e-commerce catalogue. Raw product titles of all verticals were collected to create the catalogue dataset.

Algorithm 1 Diverse Query Generation

Input: A start token s, beam width, γ, max number of tokens δ
Output:Top diverse query candidates set Σ
// Assumptions:
// getNextWords(prefix, p_t) returns next words having probability greater than p_t;
// Beam(γ) return a beam of width γ with *add* operation.
// BeamIterator(BG, p_t, δ) performs beam search until numTokens > δ or EOS is encountered and return set of generated queries. BG is beam group.
p_t^l - prob. threshold for level l such that $p_t^l < p_t^{l+1}$
first_level_sug = [], second_level_sug = []
for $w_s \in$ getNextWords(s, p_t^1) **do**
    first_level_sug.append(s+w_s)
end for
for s_1 in first_level_sug **do**
    for $w_s \in$ getNextWords(s_1, p_t^2) **do**
        second_level_sug.append(s_1+w_s)
    end for
end for
for w_s in second_level_sug **do**
    beam_group = Beam(γ)
    beam_group.*add*(w_s)
    suggestion_list = BeamIterator(beam_group, p_t^3, δ)
    $\Sigma = \Sigma \cup$ suggestion_list
end for
return Σ

After data pre-processing (section 4.1), we exclude user queries with impression < 5 and product titles having a length > 10 tokens. Our final query dataset has ~6 million unique queries, and the catalogue dataset has ~4 million unique titles. Out-of-vocabulary words were mapped to the 'unk' token. We split the data in the ratio of 7:3 to create a training and validation set. We created a standard test set of 0.15 million queries to evaluate all models. Unique user queries were randomly sampled from a query dataset (non-overlapping with a training set) to have representation of each query segment (head, torso, and tail) in the test set.

5.2 Training details

1. Model Configurations: After hyperparameter tuning, we used the following parameter settings for our model training. For both LMs, we used a 3-layer GRU-MN with 2000, 1500, and 1000 hidden unit dimensions with a dropout of 0.8 between GRU layers. Query LM had attention unit dimensions of 1000, 750, and 500 while the Catalogue LM had attention units of size 500, 375, and 250 for respective layers. The intermediate context layer embedding size was set to 500 in both models. We used in-domain fasttext [16] embeddings for initialization. GRU models optimized the cross-entropy loss with gradient clipping. Multiple GPT-2 models were trained with varying layers, embedding size and attention heads. We report the selective best results. The common BERT-Base-Cased model was trained with 12 layers 12 heads. Both BERT and GPT-2 models used 256-dimensions and were pre-trained on in-domain e-commerce data. All neural models used Adam optimization with learning rate of 10^{-5}. The training was done with a mini-batch size of 128 on NVIDIA TESLA V100 GPUs.

2. Category specific EOS/BOS tokens: With global EOS and BOS tokens, we observed premature termination of queries for some

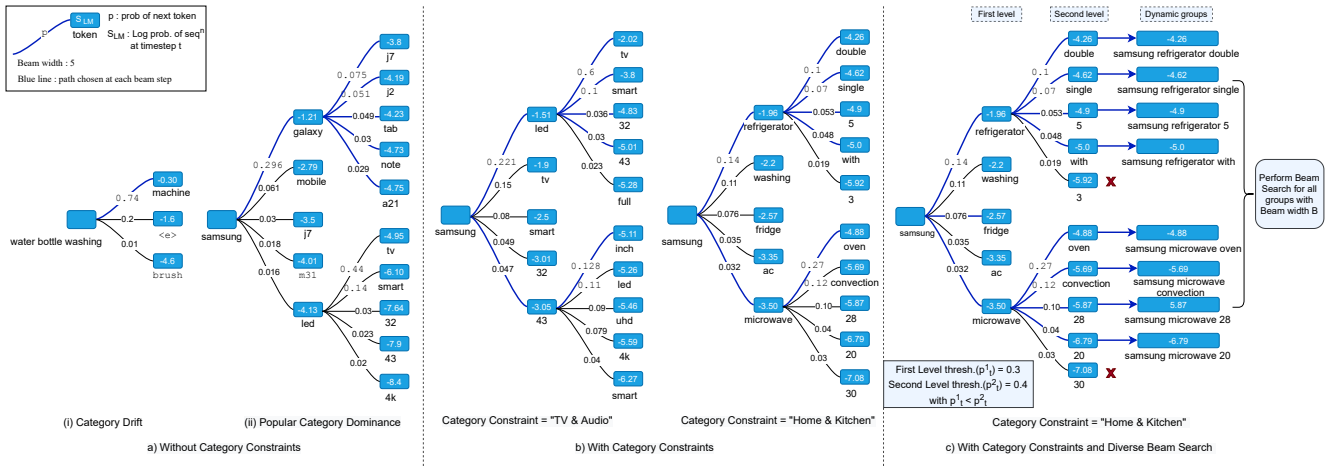

Figure 5: Category constraints & dynamic beam search - a.(i) for prefix 'water bottle washing', 'machine' gets very high score causing category drift a.(ii) suggestions related to 'samsung led' are discarded at the end of beam step due to lower frequency in training data (b) category constraint ='TV & Audio', causes 'led' to get high probability. Similarly for category constraint='Home Kitchen' but both with less diversity (c) Improvements in diversity by using dynamic beam search.

categories in GRU models. Hence, we create category specific BOS and EOS tokens. For example, for *Home & Kitchen* BOS and EOS are *<s_home_kitchen>* and *<e_home_kitchen>* respectively.

3. Inference Optimizations: Static padding impacts the run time of query generation because, for each token, the inference is made for all time steps equal to max padded length. Hence we implemented our GRU-MN unit with dynamic padding support, where each query instance is padded to the length of the longest query sequence in a batch. During query generation, padding is skipped by default as batch size is 1. We imputed all low-frequency tokens (freq<5) with a single *'rare_token'*. With dynamic padding, GPU inference time per token was reduced from **46 ms** to **9.8 ms**. These optimizations pave the way for us to use this model in the online user path in the future. GPT-2 models had per token inference time of 25ms and BERT had 30ms which is not a concern for offline candidates but are prohibitive in our future online setting and hence still needs further exploration to reduce inference time.

5.3 Baseline Models

(1) **Statistical Language Model (SLM) with LTR:** We implemented category specific Google's n-gram (n=4 to keep vocabulary and model size manageable) statistical language model (SLM) [8]. At serving time, we score the relevant matching candidates, by a learning-to-rank(LTR) model similar to [23] and [32]. We refer to this baseline model as **SLM**.

(2) **Neural Language Model (NQLM):** We use [25] as our second baseline, which implements a word-level multi-layer LSTM LM followed by a LambdaMART model to rank candidates. We also call this **NQLM** as called by its authors [25].

(3) **GPT-2:** Given the auto-completion task, the auto-regressive GPT-2 [26] models are natural choice of large pre-trained (on in-domain data) models. We use the notation *GPT2-L(number of layers)-H(number of heads)* to represent different GPT-2 model configurations. We induced category constraints in GPT-2 versions by adding category specific start token and this configuration notation is suffixed with the "+CC" qualifier.

(4) **BERT:** Since BERT[12] is an encoder model with attention from both directions, for the QAC task, we fine-tuned the in-domain pre-trained model by masking the last token.

5.4 Offline Model Evaluation

Besides the model log-likelihood, we compared offline the top-k (k=1) accuracy of the models i.e. the actual user selected query in the logs was present in the top-k query predictions from the model for a given prefix. We compare offline numbers of the following models, first set of which are the query LMs against the above baselines.

(1) **GRU-MN Language Model with Category Constraints**: Our multi-layer GRU-MN with category constraints on user queries. We refer to this model as **NQLM+CC**.

(2) **GRU-MN Language Model with Category Constraints and Self Attention:** Same as our NQLM+CC above but with self attention in the GRU MN cell. We refer to this model as **NQLM+CC+SA**.

(3) **Catalogue Language Model on raw product titles:** Our neural LM with category constraints and self-attention on the raw catalogue product titles without any pre-processing by statistical measures of entropy and PMI.

(4) **Catalogue Language Model:** Our neural LM with category constraints and self-attention on the cleansed catalogue product titles. We call this model **NCLM+CC+SA**.

Table 5 compares offline model metrics (average log likelihood and top-1 accuracy) of different models. For SLM, we used stupid back-off smoothing [8]; hence it is an *unnormalized** log-likelihood. However, its log-likelihood scores would be further lower than the

Model	Avg. log likelihood	Top-1 Accuracy
User Query Models		
Baselines		
SLM [23, 32]	−7.53*	0.14
Neural LM* (NQLM [25])	-4.65	0.30
GPT2-L4-H8	-4.44	0.36
GPT2-L8-H12	-4.07	0.41
GPT2-L8-H12+CC	-3.49	0.42
BERT-L12-H12(Base-Cased)	-4.49	0.31
Proposed Models		
NQLM+CC	-3.55	0.40
NQLM+CC+SA	**-3.32**	**0.45**
Catalogue Title Models		
NCLM+CC+SA (raw titles)	-5.29	0.35
NCLM+CC+SA	**-4.65**	**0.41**

Table 5: Offline performance metrics for different models

current values when normalized. NQLM is an improvement over SLM showing neural LMs are better than SLMs. Our empirical results further show that enhancements like category constraints on neural LM improve the quality of the autosuggest queries, as seen in all the metrics. Adding self-attention in the GRU-cell further improves the metrics compared to using just category constraints. GPT-2 models are better than NQLM models with category constrained variant being the best GPT-2 model. However, adding category constraint to our NQLM variant brings it very close to GPT-2 models on top-1 accuracy showing the significance of dynamic category constraint. Further, self-attention in the GRU cell in our architecture outperforms GPT-2 and BERT and this is most likely due to latter being very high capacity models which is better suited for longer sequences and longer contexts. Also, BERT based QAC generated relatively poor and very short suggestions (2-3 tokens) by aggressively predicting a terminal word. Similar to our results, LSTMs have been shown to outperform the large pre-trained language models like BERT on other datasets too [13].

Observing the query language model performance comparison of GPT and NQLM+CC+SA, we use best Query LM architecture to train on Catalogue titles. Pre-processing the titles and learning another category constrained neural language model on them further improved the top-k accuracy of the AS results. The last two rows of Table 5 demonstrate the improvement brought by the extractive summarization cleansing on product titles. Training a single model on both queries and catalogue titles or fusion of two separate LMs showed a deterioration in the model metrics.

Next, we do offline evaluations to answer the following research questions:

- **Q1: Cold start and the visibility of new products?** To compute the % increase in the new products served through AS queries, we retrieved all unique products before and after ingesting new candidates generated through Catalogue LM. We saw a significant increase of **3.7%** in the new products served.

Quality Error Class	Description	Example
Non-redundancy	No tokens should be repeated in the same query, either same meaning or belonging to same class(tag).	iphone 12 cover redmi
Structure and Coherence	Query should have a proper structure/sequence of tokens. Query should not end abruptly.	air conditioner white voltas window

Table 6: Quality Guidelines for Human Assessment

- **Q2: Well-formedness of the generated queries?** While Table 5 reports the empirical improvements from the category constraints, Fig. 5 demonstrates the effect through a qualitative example. Fig. 5a shows concept drift encountered in the state-of-the-art neural baseline model [25] where a cross-category query like 'water bottle washing machine' was generated with high confidence which is a mix of 2 different categories - 'General Merchandise' and 'Home & Kitchen'.

  For offline quantitative error analysis, we defined basic quality error classes for assessment by human judges as outlined in Table 6. Table 7 captures the improvement in human judged quality of the generated candidates as we introduce the category constraints. Our work's future area is to refine the guidelines for human evaluation further and create a more robust evaluation metric of AS candidate quality.
- **Q3: Diversity and coverage of attributes?** Fig 5(a) shows the issue of popular category dominance. For the start token 'samsung', only suggestions from popular category -'Mobiles and Cases&Covers' survive at the end of the beam search. Fig 5(b) shows the issue with vanilla beam search, where it produces less diverse results. We evaluated the following three metrics to measure the impact of category constraints and diverse beam search in the diversity of generated candidates:
  - **Unique Facet Key Count** - Counts the number of unique facet keys like color, size, occasion, etc., normalised by the number of suggestions shown for a prefix.
  - **Unique Facet Value Count** - Counts the number of unique facet values normalised by the number of suggestions shown for a prefix. For eg, 'red', 'blue' and 'medium', 'small' are facet values for facet keys 'color' and 'size' respectively.
  - **Unique Verticals Count** - Counts the number of unique landing verticals normalised by the number of suggestions shown for a prefix. E.g., for prefix 'nike', suggestions 'nike shoes for men', and 'nike shorts for men' lead to two different verticals.

  For a start token, candidates were generated from different model configurations. Our e-commerce NER (named entity recognition) system was used to assign the facet/attribute to each token of the generated query to compute the above metrics. Fig 6 reports the average of these metrics (over 10K start tokens). It can be seen that category constraints and diverse beam search have a significant impact on attribute diversity. Also, due to high information content(attributes) in the Catalogue LM, queries generated from Catalogue LM are more diverse than Query LM.

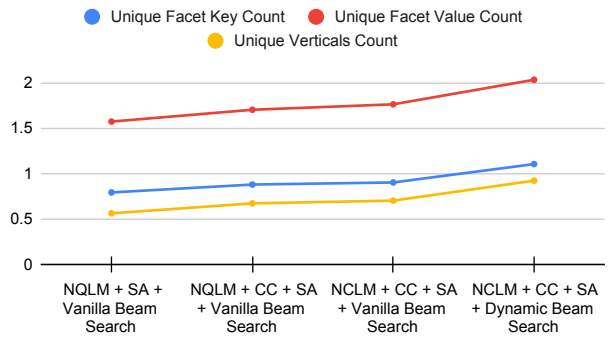

Figure 6: Ablation study of different model configurations w.r.t Diversity

	SLM	NQLM	NQLM+CC
Relative #quality issues w.r.t SLM	1	0.7	0.1

Table 7: Relative ratio of human judged quality issues w.r.t SLM baseline

5.5 Online Evaluation

To observe the actual feedback at scale from the users of our e-commerce platform, we ran online A/B experiments with **SLM**, **NQLM** and **NQLM+CC+SA** against the control bucket of historically popular queries(referred as **Hist Pop** in the A/B results Table 8). The A/B test results are evaluated at 5% significance level. In our production system, we first launched **NQLM+CC+SA** on 100% user traffic replacing the existing **Historically popular queries** model. In the following online A/B, this became the new control against the treatment bucket of the Catalogue Neural model **NCLM+CC+SA**. We found that the catalogue model's superiority on offline metrics also held in the online A/B test. Table 8 summarizes the A/B test comparison and reports AS Usage, Search CTR, and product conversion (purchase) metrics. **1 bps** increase in purchase rate corresponds to an increase of US $50-100M in GMV (Gross Merchandise Value) on our e-commerce platform. Hence, the business improvements of Search Page CTR and purchase rates from our neural variants of Query and Catalogue language models are significant besides the positive movement in AS Usage. CADENCE has been in production for almost a year now, and the models are refreshed every week to generate new candidates.

Treatment	Control	AS Usage	Search CTR	Search Purchase Rate
SLM[23, 32]	Hist Pop	-44 bps	-8 bps	-2 bps
NQLM	Hist Pop	+6 bps	-23 bps	-1 bps
NQLM+CC+SA	Hist Pop	**+5 bps**	**+2 bps**	**+2 bps**
NCLM+CC+SA	NQLM+CC+SA	**+3 bps**	**+6 bps**	**+2 bps**

bps or basis points is a standard & significant unit to measure change in metrics in e-commerce industry. 1bps = 0.01%

Table 8: Online A/B test comparison of different AutoSuggest LM models at 5% significance level

It is interesting to note from SLM and NQLM model's online performance that while they improve offline metrics like top-k accuracy, it is tough to beat the down-the-funnel performance of historically popular queries. Candidates from both these models negatively impacted downstream online metrics of Search CTR and purchase rate. NQLM generated better suggestions than SLM, resulting in increased AS Usage. However, because of the presence of quality issues (Table 7) like redundancy, bad and incoherent query structure suggestions (as defined in our human assessment guidelines, Table 6), query suggestions from these models produced less clickable product results leading to drop in both the search CTR and conversion metrics. On the other hand, our neural variants improved the AS quality issues significantly, which is reflected in the business metrics.

We also did a comparison of AS coverage (number of relevant suggestions or recall) against user prefixes of **Hist Pop** only candidates versus the final QAC which indexes all candidates (**NQLM+CC+SA**, **NCLM+CC+SA**, **Hist Pop**). Fig.7 shows % decrease in prefix count which have lesser (0,1,2 etc.) suggestions in the combined bucket indicating increased recall. This is due to the increased diversity of candidates from the dynamic beam search as well as new candidates generated by both the Query and Catalogue LMs.

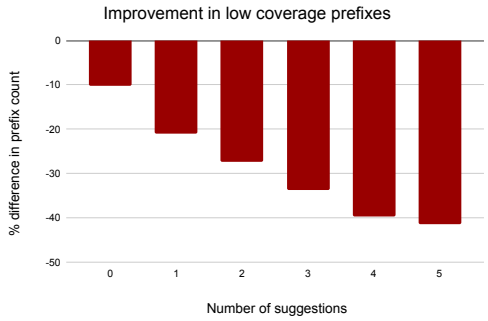

Figure 7: % drop in the number of prefixes having lesser suggestions in the final combined QAC against the popular suggestions. Decrease in prefix count indicates increased recall

6 CONCLUSION

We demonstrated that given the short-length nature of e-commerce queries, our auto-regressive GRU-MN (with variable self attention) deep network with deep category constraints improves the quality and performance of our e-commerce QAC system. It is better than the state of the art sequential neural models as well as large pre-trained models like BERT, GPT-2. Secondly, our customized dynamic beam search algorithm brings diversity in the query suggestions. And finally, the catalogue language model trained after product titles addresses the product cold start problem in AutoSuggest. As our current solution generates the AS candidate offline in bulk, our next step is to use this model to generate queries on the fly as the user types in and improve the AS coverage further. Also, we would like to try open source LLMs like OPT [36] in the offline setting.

7 ACKNOWLEDGEMENTS

We thank our colleagues Akash Khandelwal, Rahul Saxena, Divya Rai, Krishan Goyal, Charu Agrawal and the entire AutoSuggest engineering team for stimulating discussions, critical feedback and helping with deployments. We also thank our human annotators for their help in labelling and evaluation.

REFERENCES

[1] 2005. Apache Lucene. (2005). https://lucene.apache.org/core

[2] Luca Aiello, Ioannis Arapakis, Ricardo Baeza-Yates, Xiao Bai, Nicola Barbieri, Amin Mantrach, and Fabrizio Silvestri. 2016. The Role of Relevance in Sponsored Search. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management* (Indianapolis, Indiana, USA) *(CIKM '16)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2983323.2983840

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

[4] Holger Bast, Debapriyo Majumdar, and Ingmar Weber. 2007. Efficient Interactive Query Expansion with Complete Search. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management* (Lisbon, Portugal) *(CIKM '07)*. Association for Computing Machinery, New York, NY, USA, 857–860. https://doi.org/10.1145/1321440.1321560

[5] Holger Bast and Ingmar Weber. 2006. Type Less, Find More: Fast Autocompletion Search with a Succinct Index. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Seattle, Washington, USA) *(SIGIR '06)*. Association for Computing Machinery, New York, NY, USA, 364–371. https://doi.org/10.1145/1148170.1148234

[6] Sumit Bhatia, Debapriyo Majumdar, and Prasenjit Mitra. 2011. Query Suggestions in the Absence of Query Logs. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Beijing, China) *(SIGIR '11)*. Association for Computing Machinery, New York, NY, USA, 795–804. https://doi.org/10.1145/2009916.2010023

[7] G. Bouma. 2009. Normalized (pointwise) mutual information in collocation extraction.

[8] Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large Language Models in Machine Translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Association for Computational Linguistics, Prague, Czech Republic, 858–867. https://aclanthology.org/D07-1090

[9] Andrei Broder. 2002. A Taxonomy of Web Search. *SIGIR Forum* 36, 2 (2002), 3–10. https://doi.org/10.1145/792550.792552

[10] Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long Short-Term Memory-Networks for Machine Reading. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (2016). https://doi.org/10.18653/v1/d16-1053

[11] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation* (2014). https://doi.org/10.3115/v1/w14-4012

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[13] Aysu Ezen-Can. 2020. A Comparison of LSTM and BERT for Small Corpus. (2020). https://doi.org/10.48550/ARXIV.2009.05451

[14] Yu Gong, Xusheng Luo, Kenny Q. Zhu, Wenwu Ou, Zhao Li, and Lu Duan. 2019. Automatic Generation of Chinese Short Product Titles for Mobile Display. *Proceedings of the AAAI Conference on Artificial Intelligence* 33 (Jul 2019), 9460–9465. https://doi.org/10.1609/aaai.v33i01.33019460

[15] Dustin Hillard, Eren Manavoglu, Hema Raghavan, Chris Leggetter, Erick Cantú-Paz, and Rukmini Iyer. 2011. The Sum of Its Parts: Reducing Sparsity in Click Estimation with Query Segments. *Inf. Retr.* 14, 3 (2011). https://doi.org/10.1007/s10791-010-9152-6

[16] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomás Mikolov. 2016. Bag of Tricks for Efficient Text Classification. *CoRR* abs/1607.01759 (2016). http://arxiv.org/abs/1607.01759

[17] Dan Jurafsky and James Martin. 2021. Statistical Language Processing. (2021). https://web.stanford.edu/~jurafsky/slp3/

[18] Gyuwan Kim. 2019. Subword Language Model for Query Auto-Completion. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (2019). https://doi.org/10.18653/v1/d19-1507

[19] A Klose, A Nürnberger, R Kruse, G Hartmann, and M Richards. 2000. Interactive text retrieval based on document similarities. *Physics and Chemistry of the Earth, Part A: Solid Earth and Geodesy* 25, 8 (2000), 649–654. https://doi.org/10.1016/S1464-1895(00)00100-9 DUST-2: An Improved Interactive Access to Information about the Earth atmosphere.

[20] Fu Y. Li Y. et al Lin, Y. 2020. Self-attention-based neural networks for refining the overlength product titles. In *Multimed Tools Appl (Multimed Tools Appl)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1007/s11042-021-10908-x

[21] David Maxwell, Peter Bailey, and David Hawking. 2017. Large-Scale Generative Query Autocompletion. In *Proceedings of the 22nd Australasian Document Computing Symposium* (Brisbane, QLD, Australia) *(ADCS 2017)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3166072.3166083

[22] Tomas et al Mikolov. 2017. Recurrent neural network based language model *(INTERSPEECH 2010)*.

[23] Bhaskar Mitra and Nick Craswell. 2015. Query Auto-Completion for Rare Prefixes. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management* (Melbourne, Australia) *(CIKM '15)*. Association for Computing Machinery, New York, NY, USA, 1755–1758. https://doi.org/10.1145/2806416.2806599

[24] Ajinkya More. 2016. Attribute Extraction from Product Titles in eCommerce. (2016). arXiv:1608.04670 [cs.CL]

[25] Dae Hoon Park and Rikio Chiba. 2017. A Neural Language Model for Query Auto-Completion. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Shinjuku, Tokyo, Japan) *(SIGIR '17)*. Association for Computing Machinery, New York, NY, USA, 1189–1192. https://doi.org/10.1145/3077136.3080758

[26] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. (2019).

[27] Parikshit Sondhi, Mohit Sharma, Pranam Kolari, and ChengXiang Zhai. 2018. A Taxonomy of Queries for E-Commerce Search. In *The 41st International ACM SIGIR Conference on Research Development in Information Retrieval* (Ann Arbor, MI, USA) *(SIGIR '18)*. Association for Computing Machinery, New York, NY, USA, 1245–1248. https://doi.org/10.1145/3209978.3210152

[28] Yuanhang Su and C.-C. Jay Kuo. 2019. On extended long short-term memory and dependent bidirectional recurrent neural network. *Neurocomputing* 356 (Sep 2019), 151–161. https://doi.org/10.1016/j.neucom.2019.04.044

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[30] Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R. Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2016. Diverse Beam Search: Decoding Diverse Solutions from Neural Sequence Models. (2016). arXiv:1610.02424 [cs.AI]

[31] Po-Wei Wang, Huan Zhang, Vijai Mohan, Inderjit S Dhillon, and J Zico Kolter. 2018. Realtime query completion via deep language models. In *eCOM@ SIGIR*.

[32] Sida Wang, Weiwei Guo, Huiji Gao, and Bo Long. 2020. *Efficient Neural Query Auto Completion.* Association for Computing Machinery, New York, NY, USA, 2797–2804. https://doi.org/10.1145/3340531.3412701

[33] Zhongyuan Wang, Haixun Wang, Ji-Rong Wen, and Yanghua Xiao. 2015. An Inference Approach to Basic Level of Categorization. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management* (Melbourne, Australia) *(CIKM '15)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2806416.2806533

[34] Ryen W. White and Gary Marchionini. 2007. Examining the Effectiveness of Real-Time Query Expansion. *Inf. Process. Manage.* 43, 3 (May 2007), 685–704. https://doi.org/10.1016/j.ipm.2006.06.005

[35] Joan Xiao and Robert Munro. 2019. Text Summarization of Product Titles. *SIGIR 2019 eCom* (Jul 2019). https://sigir-ecom.github.io/ecom2019/ecom19Papers/paper36.pdf

[36] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. OPT: Open Pre-trained Transformer Language Models. arXiv:2205.01068 [cs.CL]