HEARTS: Multi-task Fusion of Dense Retrieval and Non-autoregressive Generation for Sponsored Search

Bhargav Dodla *†
dodla.bhargav@gmail.com
Indian Institute of Technology Madras
Chennai, India

Akash Kumar Mohankumar*
makashkumar@microsoft.com
Microsoft
Bengaluru, India

Amit Singh
siamit@microsoft.com
Microsoft
Bengaluru, India

ABSTRACT
Matching user search queries with relevant keywords bid by advertisers in real-time is a crucial problem in sponsored search. In the literature, two broad set of approaches have been explored to solve this problem: (i) Dense Retrieval (DR) - learning dense vector representations for queries and bid keywords in a shared space, and (ii) Natural Language Generation (NLG) - learning to directly generate bid keywords given queries. In this work, we first conduct an empirical study of these two approaches and show that they offer complementary benefits that are additive. In particular, a large fraction of the keywords retrieved from NLG haven’t been retrieved by DR and vice-versa. We then show that it is possible to effectively combine the advantages of these two approaches in one model. Specifically, we propose HEARTS: a novel multi-task fusion framework where we jointly optimize a shared encoder to perform both DR and non-autoregressive NLG. Through extensive experiments on search queries from over 30+ countries spanning 20+ languages, we show that HEARTS retrieves 40.3% more high-quality bid keywords than the baseline approaches given queries. In addition, we demonstrate that HEARTS objective is significantly better than those trained with the standard contrastive loss functions. Finally, we show that our HEARTS objective can be adopted to short-text retrieval tasks other than sponsored search and achieve significant performance gains.

CCS CONCEPTS
• Computing methodologies → Natural language generation: • Information systems → Sponsored search advertising: Retrieval models and ranking.

KEYWORDS
sponsored search, dense retrieval, natural language generation

1 INTRODUCTION
Overview: Sponsored search plays a critical role in supporting free access to web search offered by search engines that have rapidly become essential to Billions of web users worldwide. In sponsored search, search engines allow advertisers to target the search queries their ad can be shown to by allowing them to bid on keywords of different match types. For instance, in the Exact Match (EM) type, a bid keyword is matched to search queries with the same search intent. A more relaxed alternative is Phrase Match (PM), where queries that contain or include the meaning of the keyword are matched. Such matching between search queries and relevant EM or PM keywords is nuanced and challenging for several reasons. First, search queries/bid keywords are often short, and understanding the actual intent of the user/advertiser is a difficult task. For example, the query "amazon microsoft 365 single" is a phrase match to "microsoft 365 personal" but not to "microsoft 365 login". Further, the queries/keywords could be from a wide range of topics or domains, e.g., apparel, travel, health, etc., (ii) languages, and (iii) countries. Another challenge is a query must be matched with all possible high-quality EM/PM keywords from a massive bid keyword library, not just one. In addition, the matching must be done in realtime, within a few milliseconds, and be computationally efficient to serve all the search traffic.

Existing Methods: Various approaches have been explored in the literature to solve the above-mentioned query to keyword matching problem. One set of methods views this as an Information Retrieval task where queries and keywords are represented in a shared vector space with the objective that similar query keyword pairs are close to each other while dissimilar ones are farther apart. Early works \cite{4, 5, 33} in sponsored search predominantly relied on sparse representations using bag-of-words features such as tf-idf for queries and keywords/ads. However, recent works \cite{2, 7–9, 11} have shown superior performance by learning dense representations for queries and keywords using various learning objectives and negative mining strategies. Concurrently, another direction of work \cite{19, 25, 31} has proposed to model this problem as a sequence-to-sequence constrained generation task. In particular, such approaches train language models to generate query rewrites and constrain the generations during inference using a prefix tree (trie) of bid keywords. However, a key drawback of these methods is that they adopt Autoregressive (AR) NLG models, which have high inference latency and do not meet the strict requirements for online inference. Therefore, in this study, we focus on non-autoregressive (NAR) models that allow parallel generations with significant speed-ups since they assume the target (keyword) tokens are conditionally independent of each other, given the source sentence (query).

DR & NLG Comparison: Given these two distinct approaches to the same problem, an obvious question is: How do these two methods compare, and how different are the retrieved keywords for

\footnote{Both authors contributed equally to this work}
\footnote{Work done during internship at Microsoft}

1 https://support.google.com/google-ads/answer/7478529?hl=en
2 https://help.ads.microsoft.com/#apex/en/90822/1
which would double our compute cost. While ANN or trie-based
words from NGAME and CLOVER-NAR are unique for a given
we propose CLOVER-NAR, an encoder-only NAR variant of the
(40%) of the high-quality keywords retrieved by HEARTS is
98% of the CLOVER-NAR + NGAME ensemble but at half
the GPU cost. Further, we show the DR component of HEARTS
problems beyond sponsored search. We perform experiments on
two publicly available short-text benchmarks, viz., AmazonTitles-
131K and WikiSeeAlsoTitles-320K from the extreme classification
repository [3]. We compare HEARTS with the leading extreme clas-
sification algorithms and show that HEARTS outperforms them in
all the metrics. To summarize, our key contributions are as follows:

• We conduct various empirical experiments to benchmark the
performance of Dense retrieval and non-autoregressive NLG
models on the problem of retrieving EM and PM bid key-
words given queries.

• We show that a large fraction (40%) of the bid keywords
retrieved by the two approaches for a given query are distinct
and do not overlap with each other, pointing to the fact that
their gains are additive.

• We propose HEARTS, a multi-task fusion framework to com-
bine DR and NLG methods in one single model and show that
a single HEARTS model with one forward pass is as good as
two different DR and NLG models.

• Lastly, we show that using the HEARTS objective as an al-
ternative to the standard DR contrastive loss significantly
improves retrieval performance. We show that HEARTS (DR)
outperforms DR baselines on both sponsored search and other
short-text retrieval benchmarks.

2 RELATED WORK

Information Retrieval: Before the advent of Deep Learning, sev-
eral works viewed query to keyword/ad matching as a sparse re-
trieval task where queries and ads to static features based on the

Figure 1: Venn Diagram of different keyword sets for a given
query. Diagram illustrates the benefit of HEARTS in combining
the high-quality keywords from NLG and DR in one model.

the same query? To answer this question, we conduct various empir-
ical experiments on three diverse and high-quality query-keyword
datasets curated from the logs of a large commercial search engine in
30+ countries. We adopt the NGAME algorithm [8], which provides
state-of-the-art performance on the extreme classification bench-
mark, as our Dense Retrieval (DR) baseline. As the NLG baseline,
we propose CLOVER-NAR, an encoder-only NAR variant of the
CLOVER model proposed by [25]. While we show that NGAME
(DR) tends to perform better than CLOVER-NAR (NLG), a signif-
icant fraction of the high-quality keywords retrieved by these two
methods for a given query are distinct and do not overlap with each
other. In particular, around 40% of the high-quality retrieved key-
words from NGAME and CLOVER-NAR are unique for a given
query. We hypothesize that this happens because the two techniques
have very different training objectives that offer complementary
advantages. Specifically, since we train NLG models to predict the
target (keywords) at the word/token level, we notice that the trained
models are better at identifying word-level patterns between the
source (query) and the target (keyword). For instance, as shown in
table 1 (1st example), CLOVER-NAR is able to infer that the tokens
\textit{microsoft} and \textit{ms} (acronym) can be interchangeably used without
affecting the intent of the query/keyword. On the other hand, in DR
methods, the model is encouraged to learn abstract representations
that capture the semantics of queries/keywords. As a result, we no-
tice (see table 1) that DR models are better at retrieving lexically
diverse keywords for a given query, which is a well-known short-
coming in NLG models [25].

HEARTS: With the NLG and DR models retrieving significantly
different high-quality keywords for the same query, it is natural
to wonder if we can ensemble these two models and combine
the benefits of the two approaches. However, the challenge with an en-
semble is we need to severe queries online with two different models,
which would double our compute cost. While ANN or trie-based
beam search can run efficiently on CPU cores (relatively cheap),
we require hardware accelerators such as GPUs, FPGAs, or TPUs
to perform forward-pass over transformer encoders such as BERT
within a few milliseconds. Doubling the number of GPUs/FPGAs
needed to serve queries at the web-scale can be very expensive.
In this work, we show that its possible to blend the advantages of
NLG and DR methods into one single model. We propose HEARTS:
Hybrid of gEneration And rEntrieval for Sponsored search, a novel
multi-task learning framework to learn shared representations for DR
and NLG methods. In HEARTS, we jointly optimize (i) a contrastive
triplet margin loss with hard negative sampling for DR and (ii) a
non-autoregressive log-likelihood objective for NLG using a shared
encoder. Note that HEARTS is different from standard multi-task
learning settings, where we usually learn to solve two related prob-
lems using a shared model. In HEARTS, we solve the same problem
(bid keyword retrieval) but when viewed as two different tasks - DR
and NLG in one model. As illustrated in figure 1, a single HEARTS
model can retrieve almost all the high-quality keywords that were
retrieved by either the NLG or DR baselines. In particular, we show
that the number of high-quality keywords retrieved by HEARTS is
around 98% of the CLOVER-NAR + NGAME ensemble but at half
the GPU cost. Further, we show the DR component of HEARTS
is significantly better than NGAME. For example, as illustrated in table
, HEARTS (DR) is able to retrieve diverse bid keywords with the
same search intent as the query, which NGAME missed. We show
that HEARTS (DR) achieves 5-7% higher good keyword density
drastically

Bhargav Dodla, Akash Kumar Mohankumar, and Amit Singh
A few approaches like Siamese-XML [7] and ANCE [39] mine hard strict online latency constraints. We instead adopt non-autoregressive variants to meet the but do not use autoregressive models due to their high latency and a prefix trie. For our NLG baseline, we follow a similar approach since, the generations are constrained to the bid keyword set using such a multi-stage approach (query/keyword rewriting + retrieval) could result in a gradual accumulation of errors, resulting in lower performance [19]. More recently, many works [18, 19, 25, 31] have adopted NGAME [8], a negative mining technique that curates mini-batches for training such that in-batch sampling by itself provides uninformative negatives or through in-batch sampling can provide uninformative negatives and lead to slow convergence [39]. To alleviate these issues, we adopt NGAME [8], a negative mining technique that curates mini-batches for training such that in-batch sampling by itself provides hard negative samples and leads to faster convergence. Specifically, NGAME periodically clusters the queries and chooses queries within a cluster to create mini-batches. Given a set of triplets of the form \( \{ q, k, l \} \in D \), where \( q \) is a query, \( k \) is its relevant keyword, \( l \) is the irrelevant keyword obtained from NGAME, we can minimize the triplet margin loss given by:

\[
L^\theta(\theta, D) = \frac{1}{T} \sum_{q,k,l \in D} \left[ E_{\theta}(l)^T E_{\theta}(q) - E_{\theta}(k)^T E_{\theta}(q) + y \right],
\]

where \( y \) is the margin hyperparameter. After training, we obtain all the keyword embeddings \( \{ E_{\theta}(k) : k \in K \} \) and index them on a

| Query                              | NGAME (DR)          | CLOVER-NAR (NLG)    | HEARTS (DR)      | HEARTS (NLG)     |
|------------------------------------|---------------------|---------------------|------------------|------------------|
| amazon microsoft 365 single        | ms 365 single       | office 365 ms       | microsoft licenses |                   |
| student houses for rent nottingham | rent student nottingham | unini nottingham accommodation |                   |
| telephone number for amazon returns | amazon support phone | amazon refund support number |                   |
| gaming pc inter 1300               | gaming pc best      | gaming pc build     | pc gamer 700 euro |                   |

Table 1: Samples of retrieved keywords from CLOVER-NAR, NGAME and our proposed HEARTS approach on four search queries

3 METHODOLOGY

In this section, we discuss the various components of our proposed HEARTS framework in detail. We provide an outline of the same in figure 2. Given a search query \( q \), our objective is to retrieve all exact and phrase match keywords \( k^p \) and \( k^f \) respectively from a set of valid bid keywords \( K \). We begin by discussing the details of the NGAME dense retrieval model in section 3.1. Later in 3.2, we describe our non-autoregressive NLG model - CLOVER-NAR. Lastly, in section 3.3, we discuss our proposed HEARTS framework with its multi-task learning objective.

3.1 Dense Retrieval: NGAME

To learn dense representations for queries and keywords, we adopt a standard siamese architecture consisting of two towers: query-encoder and keyword-encoder with shared parameters. Specifically, given any query \( q \) and keyword \( k \), we embed them into \( E_{\theta}(q), E_{\theta}(k) \in \mathbb{R}^d \) respectively where \( E_{\theta} \) represents the encoder with shared parameters \( \theta \). We adopt a transformer model [37] similar to the BERT-base architecture (12 layers, hidden size of 768, 12 attention heads) as our encoder with a vocabulary of size 250k tokens. To effectively train the encoder, it is critical to carefully choose a small set of irrelevant keywords for each query. This is because we cannot train on all irrelevant keywords for a query since both the number of training points and bid keywords are in millions. Further, randomly choosing the negative keywords either drawn uniformly or through in-batch sampling can provide uninformative negatives and lead to slow convergence [39]. To alleviate these issues, we adopt NGAME [8], a negative mining technique that curates mini-batches for training such that in-batch sampling by itself provides hard negative samples and leads to faster convergence. Specifically, NGAME periodically clusters the queries and chooses queries within a cluster to create mini-batches. Given a set of triplets of the form \( D = \{ (q^{(i)}, k^{(i)}, l^{(i)}) \}_{i=1}^L \) where \( q^{(i)} \) is a query, \( k^{(i)} \) is its relevant keyword, and \( l^{(i)} \) is the irrelevant keyword obtained from NGAME, we can minimize the triplet margin loss given by:

\[
L^\theta(\theta, D) = \frac{1}{T} \sum_{q,k,l \in D} \left[ E_{\theta}(l)^T E_{\theta}(q) - E_{\theta}(k)^T E_{\theta}(q) + y \right],
\]

where \( y \) is the margin hyperparameter. After training, we obtain all the keyword embeddings \( \{ E_{\theta}(k) : k \in K \} \) and index them on a
Our proposed HEARTS framework consists of a shared encoder that is trained with a combination of the non-autoregressive NLG (NLL) and Dense Retrieval (Triplet margin loss) objectives. During inference, we perform inference using DiskANN search (DR) and trie-based beam search (NLG) in parallel.

3.2 Non-Autoregressive NLG: CLOVER-NAR

Here, we model retrieving bid keywords given queries as a token-level constrained generation task. Specifically, given any query \( q \), our objective is to predict the individual tokens \( \{ k_1, \ldots, k_m \} \) of the relevant keyword \( k \). However, predicting the tokens one at a time in an autoregressive fashion is very slow as we require \( m \) sequential forward passes over the model (\( m \) is the keyword length). Further, when performing beam search with beam size \( B \), each forward pass requires \( O(B) \) compute. To overcome these issues, we focus on non-autoregressive models, which assume keywords tokens \( K \) are conditionally independent of each other given the query, i.e., the probability distribution can be decomposed as follows:

\[
p(K_1, \ldots, K_m|Q, \theta) = \prod_{i=1}^{m} p(K_i|Q, \theta)
\]

This decomposition allows us to predict the distributions \( p(K_i|Q, \theta) \) individually in parallel. In CLOVER-NAR, we encode the query using a transformer encoder model and then project the last hidden states to the output vocabulary space. Similar to AR models, we treat the generation of a special token \(<s>\) as the end-of-sequence signal. Further, to ensure that we can generate target sequences longer than the input query, we pad the query with a special token \(<pad>\) repeated \( \tau \) times. The hyperparameter \( \tau \) essentially controls the maximum keyword length relative to the query length.

Training: Given a training dataset of query and relevant keyword pairs \( D = \{ (q^{(i)}, k^{(i)}) \}_{i=1}^{L} \), we minimize the negative log-likelihood (NLL) loss, given by:

\[
L_{\text{c}}(\theta, D) = -\frac{1}{L} \sum_{q,k \in D} \sum_{t=1}^{m} \log p(K_t = k_t|Q = q, \theta)
\]

In practice, we find that using label smoothing [26] along with the NLL loss improves performance.

Inference: Our objective is to find the top-\( k \) probable keywords from the bid keyword set \( K \), given a query \( q \) and a trained model \( p(K|Q, \theta) \). However, enumerating all possible bid keywords and computing the actual top-\( k \) is infeasible since \( |K| \) is very large, in the order of millions. Further, using standard approximations such as beam search is ineffective since several of the decoded sequences may not be a bid keyword, i.e., not present in the set \( K \). We instead use trie-based beam search [19, 25], a modified beam search algorithm that guarantees that the generations belong to a given closed set. We first build a trie \( T_K \) consisting of all the bid keywords from \( K \). Later, during inference, we constrain each beam to traverse from the root to any leaf on the trie \( T_K \). Specifically, during decoding with a partially generated prefix \( k_{<t} \) at time \( t \), we only consider the children of \( k_{<t} \) in \( T_K \) as possible next tokens and explicitly set the probability of other tokens to zero. In effect, trie-based beam search is equivalent to performing vanilla beam search on the (unnormalized) constrained probability distribution:

\[
\tilde{p}(K_t = k_t|k_{<t}, q, \mathcal{K}) = \begin{cases} 
p(K_t = k_t|k_{<t}, q), & \text{if } k_t \in \text{child}_{T_K}(k_{<t}) \\ 0, & \text{otherwise}
\end{cases}
\]
We now discuss HEARTS, our multi-task learning framework that combines the previous two approaches, discussed in section 3.1 and 3.2, in one single model. We use a single encoder in HEARTS to learn shared representations for Dense Retrieval and Non-autoregressive NLG. Given a query $q = (q_1, ..., q_n)$, we prepend a [CLS] token and append a <pad> token. We then encode the query using a transformer encoder $E_q$ to get a sequence of hidden states from the last layer: $h_q^q = (h_1^q, h_2^q, ..., h_{n+1}^q)$. We use the first hidden state corresponding to the [CLS] token, $h_1^q$, as the dense representation of the query. We follow the same approach to obtain a representation $h_k^q$ for any keyword $k$. Further, we project the other query hidden states as the dense representation of the query, relevant and irrelevant keywords, respectively. The pseudocode for the implementation of trie-based beam search that is parallelized across multiple CPU cores. However, to meet the online latency constraints, we further add two pruning strategies. We remove any hypothesis $k_c$ from the beam if either (i) any of the individual token log probability, i.e., $\log p(k_c)\{q\}$ for any $s < t$, is below a token log threshold, or (ii) sentence log probability, i.e., $\sum_{i=1}^{t+1} \log p(k_c|q)$, is below a threshold. The token and sentence log thresholds act as levers to control the online latency, in addition to the beam size.

### 3.3 Multi-task Fusion: HEARTS

We now discuss HEARTS, our multi-task learning framework that combines the previous two approaches, discussed in section 3.1 and 3.2, in one single model. We use a single encoder in HEARTS to learn shared representations for Dense Retrieval and Non-autoregressive NLG. Given a query $q = \{q_1, ..., q_n\}$, we prepend a [CLS] token and append a <pad> token. We then encode the query using a transformer encoder $E_q$ to get a sequence of hidden states from the last layer: $h_q^q = \{h_1^1, h_2^1, ..., h_{n+1}^1\}$. We use the first hidden state corresponding to the [CLS] token, $h_1^q$, as the dense representation of the query. We follow the same approach to obtain a representation $h_k^q$ for any keyword $k$. Further, we project the other query hidden states $h_1^q$ into the output vocabulary space to obtain the NAR logits, i.e., $\log p(K_t|q, \theta) = Wh_1^q$, where $W \in \mathbb{R}^{V \times d}$ is a learnable weight matrix, $V$, and $d$ are vocabulary and hidden sizes.

#### Training: Given a dataset $D = \{q_i, k_i\}_{i=1}^L$, we obtain irrelevant keywords $f(k)$ for each query using the NGAME negative mining technique described in 3.1. We minimize a weighted combination of the triplet margin loss (DR) and negative log-likelihood (NLG) as follows:

$$\mathcal{L}(\theta, D) = \frac{1}{L} \sum_{q,k \in D} \left[ \left( h_k^q \right)^T h_1^q - \left( h_k^q \right)^T h_1^q + \gamma \right] $$

$$-\alpha \sum_{i=1}^{m} \log p(K_t = k_i|q, \theta)$$

where $h_1^q, h_k^q, h_1^q$ are hidden states corresponding to the query, relevant and irrelevant keywords, respectively. The pseudocode for the entire training procedure is described in Algorithm 1.

#### Inference: In HEARTS, we obtain the query representation $h_k^q$ and the NAR probabilities $p(K_t|q, \theta)$ through a single pass over the encoder $E_q$. We then parallelly retrieve bid keywords using (i) DiskANN search on the query vector $h_1^q$, and (ii) trie-based beam search on the predicted probabilities $p(K_t|q, \theta)$.

### 4 EXPERIMENTS & DISCUSSION

#### 4.1 Dataset Description

We now discuss details of the various datasets used for training and evaluation. We primarily evaluate our proposed approach on high-quality query-to-keyword datasets mined from the logs of a commercial search engine. To show the extensibility of our proposed work to tasks beyond sponsored search, we also consider two public benchmarks on short-text retrieval, popularly used to benchmark the extreme classification algorithms.

#### Sponsored Search: We curated three sponsored search datasets, viz., Query2Bid-1M, Query2Bid-5M, Query2Bid-10M, each with a varying number of bid keywords and training queries. We provide the statistics of these datasets in Table 2. The data points and labels in Table 2 represent queries and keywords, respectively. We selected queries searched by users from more than 30 countries spanning 20+ languages. We then mined query keyword pairs from various sources, including ad impression logs, offline dictionaries, and predictions from different algorithms. We then applied various relevance filtration models to obtain these high-quality datasets. Note that, during training, we consider a keyword to be relevant to a query if it’s either an exact or a phrase match. As we shall discuss in 4.3, we separately evaluate whether the retrieved keywords during inference are exact or phrase matches.

#### Short-text Benchmarks: We also evaluate our proposed approach on two short-text retrieval benchmarks from the extreme classification repository [3]: LF-AmazonTitles-131K and LF-WikiSeeAlsoTitles-320K. In AmazonTitles, the task is to predict frequently bought together items on Amazon just based on the product titles. Whereas in WikiSeeAlsoTitles, the objective is to recommend related Wikipedia articles using the page title. We observed that these datasets are not...
ideal for NLG-based retrieval since it’s difficult to predict the individual tokens of the labels given the data point. For example, consider the dataset “Calvin Klein Escape for men spray” and its label “Contradiction for Men by Calvin Klein” from LF-AmazonTitles-131K. It is tough for any NLG model to predict "Contradiction" as a possible first token given the data point unless it’s aware of all products from Calvin Klein. If the token "Contradiction" is not among the top $\beta$ probabilities, then the label will never be retrieved during trie-based beam search. On the other, it’s relatively easier for DR models to match the datapoint with the label since they share common phrases “Calvin Klein” and ‘for Men’. Due to this inherent limitation of NLG retrieval models, their performance is poor on these benchmarks. We, therefore, only evaluate the DR component of HEARTS and the DR baselines on these datasets.

4.2 Experimental Details

Sponsored Search: For all the sponsored search datasets, we adopt a 12-layer transformer model as the encoder and train it from scratch. We use a multilingual vocabulary of 250k tokens that supports all the 20+ languages we consider. We use a maximum sequence length of 16 for the queries and keywords. For each training experiment, we use 4 Nvidia 40GB A100 GPUs. In CLOVER-NAR, we train the encoder with the output projection weights for six epochs using the ADAM optimizer with a learning rate of $4 \times 10^{-5}$. For inference, we use a beam size of 100 with token-logp threshold of $-10$, sentence-logp threshold of $-50$, and the relative maximum length $u$ to 3. For NGAME training, we use a batch size of 1000 sentences with the triplet margin loss with margin $y = 0.3$. We set the cluster size $C$ as 32 and perform clustering once every $\tau_c = 5$ epochs for the 1M, 5M, and 10M dataset variants, we train the encoder for 120, 60, and 40 epochs, respectively. For inference with NGAME, we retrieve the same number of top-$k$ keywords per query as CLOVER-NAR. In HEARTS, we adopt the same training hyperparameters as NGAME. We set the loss weight $\alpha$ to 0.01 for 1M and 5M and to 0.008 for the 10M dataset.

Short-text Benchmarks: We use the Distil BERT base model [36] as the encoder and finetune from the pretrained checkpoint. For both NGAME and HEARTS, we train for 600 epochs with triplet margin of $y = 0.3$, cluster size $C$ of 32, cluster interval $\tau_c$ of 5 epochs. We use a batch size of 1600 for LF-AmazonTitles-131K and 800 for LF-WikiSeeAlsoTitles-320K. In the HEARTS objective, we set the loss weight $\alpha = 0.01$ for AmazonTitles and $\alpha = 0.005$ for WikiSeeAlsoTitles. Note that we do not train the additional classifier embedding described as Module 4 in NGAME [8]. We only use the encoder representation $E_q(.)$ as the embedding for the labels.

4.3 Evaluation Metrics

Sponsored Search: Evaluating the performance of a query-to-keyword retrieval model is in itself a challenge. Firstly, obtaining all high-quality keywords for a query through human annotations is impossible since the number of keywords is very large. Further, the evaluation would be biased if we only use a subset of the high-quality keywords per query as the ground truth. For instance, a model would be unfairly penalized for retrieving a new high-quality query keyword pair that hasn’t been mined yet. Also, as pointed by [25], standard NLG evaluation metrics such as BLEU [27], ROUGE [20] perform poorly at this task. To overcome these challenges, we follow the approach adopted by [25], i.e., train a custom evaluation model. We first curated a large-scale dataset of human judgments on the query-keyword match quality. We showed trained human annotators pairs of query keywords and asked them to label it as good or bad match. We obtain such annotations from multiple annotators for each query keyword pair to get the final consensus label. Since the labeling guidelines differ for EM and PM, we do this process separately for the exact and phrase match types. We then finetuned an XLM-R large model [6] with separate classification heads to predict the exact and phrase labels. Our evaluation model achieves an AUC-ROC score of 86.06 and 80.22 on the exact and phrase labeled data (test split). Let $M^e(q,k)$ and $M^p(q,k)$ denote the EM and PM scores provided by our evaluation model for a q, k pair. Given any set of retrieved query-keyword pairs $(q_i, k_i)_{i=1}^n$ on a test set with $n$ unique queries, we define:

$$\text{EM Good keyword Density} \big|_{EM} = \frac{\sum_{(q, k) \in \mathcal{M}^e(q,k)} 1}{n}$$

$$\text{PM Good keyword Density} \big|_{EM} = \frac{\sum_{(q, k) \in \mathcal{M}^p(q,k)} 1}{n}$$

Essentially, the good keyword densities are the count of high-quality keywords retrieved per query where a query-keyword pair is considered high-quality if the evaluation model score is above a threshold.

Short-text Benchmarks: We evaluate our approaches using the standard evaluation measures: Precision@k ($P(k, k \in \{1, 3, 5\}$), nDCG@k ($N(k, k \in \{1, 3, 5\}$), and Recall@k ($R(k)$). We also evaluate using propensity scored precision and nDCG (PSP@k ad PSN@k), which is an unbiased estimator for precision and nDCG.
We report our results on the sponsored search datasets in Table 3. In which CLOVER-NAR missed. For example, as shown in Table 1, (DR) are more aligned with the search intent of the query while specifically, we find that the retrieved keywords from HEARTS addition to the contrastive triplet loss improves the DR performance. Using the non-autoregressive NLG objective as an auxiliary loss in good keyword density for EM and PM, respectively. In other words, the strong NGAME baseline by an average of 5.82% and 7.73% in observation here. First, the DR component of HEARTS outperforms either its NLG or the DR component. We make an interesting ob-
serve that NGAME could retrieve more lexically diverse keywords, spect to both EM and PM good keyword density. In particular, we ob-
performs better than CLOVER-NAR across all three datasets with re-
spect to both EM and PM good keyword density. In particular, we ob-
serve that NGAME could retrieve more lexically diverse keywords, which CLOVER-NAR missed. For example, as shown in Table 1, for the query "student houses for rent nottingham", CLOVER-NAR mainly retrieves minor lexical variants, but NGAME could select more diverse alternatives. In the next two rows of Table 3, we show the results for our proposed HEARTS model when inferred using either its NLG or the DR component. We make an interesting ob-
ervation here. First, the DR component of HEARTS outperforms the strong NGAME baseline by an average of 5.82% and 7.73% in good keyword density for EM and PM, respectively. In other words, using the non-autoregressive NLG objective as an auxiliary loss in addition to the contrastive triplet loss improves the DR performance. Specifically, we find that the retrieved keywords from HEARTS (DR) are more aligned with the search intent of the query while also being diverse. For instance, in the same example in Table 1, the retrieved keywords from HEARTS-DR contain the important token nottingham which was missing in a few of the keywords retrieved by NGAME. This might be a result of the additional token-level NAR objective, which enforces the hidden representations to predict the individual keyword tokens. On the other hand, the NLG component of HEARTS does not improve over the CLOVER-NAR baseline; in fact, there is a small regression of around 2%. In many samples, we observe that the key factor limiting the performance of the NLG models is its non-autoregressive nature and not the quality of the hidden representations. As shown in Table 1, the NLG approaches tends to retrieve a few nonsensical bid keywords such as "amazon www amazon" because it does not have the context of the previously generated tokens. While the performance of both the NLG models is similar, we further analyzed if they retrieved almost the same keyword set for any given query. Surprisingly, we find that, on average 16.4% and 22.9% of the high-quality EM and PM keywords retrieved by HEARTS (NLG) are unique and do not overlap with the CLOVER-NAR. Similarly, 21.7% of the high-quality PM keywords retrieved by HEARTS (DR) do not overlap with NGAME. This points to the fact that we would still get considerable gains if we could run both the NLG or DR models in parallel.

| Dataset       | Model                  | GPU Compute | Total Kwd Density @0.5 | EM Good Keyword Density @0.5 | PM Good Kwd Density @0.5 |
|---------------|------------------------|-------------|------------------------|-------------------------------|--------------------------|
| Query2Bid-1M  | CLOVER-NAR (NLG)       | 1x          | 50.76                  | 2.49                          | 1.19                     |
|               | NGAME (DR)             |             | 51.00                  | 3.45                          | 1.63                     |
|               | HEARTS (NLG)           |             | 50.51                  | 2.42                          | 1.16                     |
|               | HEARTS (DR)            |             | 51.00                  | 3.62                          | 1.7                      |
| Intra Ensemble| NLG Ensemble           | 2x          | 86.14                  | 2.87                          | 1.33                     |
|               | DR Ensemble            |             | 80.61                  | 3.86                          | 1.72                     |
| Cross Ensemble| CLOVER-NAR + NGAME     | 2x          | 93.69                  | 4.11                          | 1.85                     |
|               | HEARTS (Both)          | 1x          | 91.69                  | 3.94                          | 1.78                     |
| Query2Bid -5M | CLOVER-NAR (NLG)       | 1x          | 85.63                  | 5.86                          | 2.74                     |
|               | NGAME (DR)             |             | 86.00                  | 9.45                          | 4.55                     |
|               | HEARTS (NLG)           |             | 84.05                  | 5.79                          | 2.68                     |
|               | HEARTS (DR)            |             | 86.00                  | 9.93                          | 4.81                     |
| Intra Ensemble| NLG Ensemble           | 2x          | 135.13                 | 7.04                          | 4.2                      |
|               | NGAME Ensemble         |             | 131.06                 | 11.29                         | 5.05                     |
| Cross Ensemble| CLOVER-NAR + NGAME     | 2x          | 158.27                 | 11.59                         | 5.25                     |
|               | HEARTS (Both)          | 1x          | 152.57                 | 11.4                          | 5.12                     |

Table 3: Performance of NGAME, CLOVER-NAR, HEARTS and their various ensembles on the three sponsored search datasets

under missing labels [13]. Propensity weighted metrics essentially provide more weightage to correctly classifying a tail label than popular head labels.

4.4 Results & Discussion: Sponsored Search

We report our results on the sponsored search datasets in Table 3. In the first two rows within each dataset, we show the results of the baselines - CLOVER-NAR and NGAME. We observe that NGAME performs better than CLOVER-NAR across all three datasets with respect to both EM and PM good keyword density. In particular, we observe that NGAME could retrieve more lexically diverse keywords, which CLOVER-NAR missed. For example, as shown in Table 1, the NLG approaches tends to retrieve a few nonsensical bid keywords such as "amazon www.amazon" because it does not have the context of the previously generated tokens. While the performance of both the NLG models is similar, we further analyzed if they retrieved almost the same keyword set for any given query. Surprisingly, we find that, on average 16.4% and 22.9% of the high-quality EM and PM keywords retrieved by HEARTS (NLG) are unique and do not overlap with the CLOVER-NAR. Similarly, 21.7% of the high-quality PM keywords retrieved by HEARTS (DR) do not overlap with NGAME. This points to the fact that we would still get considerable gains if we could run both the NLG or DR models in parallel.
Lastly, we consider the results of ensembles between DR and NLG, i.e., cross ensembles, in table 1. While NGAME clearly outperformed CLOVER-NAR, we see that around 23.6% and 49.0% of the high-quality EM and PM retrievals from CLOVER-NAR weren’t retrieved by NGAME. In other words, if we serve queries online using both CLOVER-NAR and NGAME in parallel, we would obtain a 14.5% and 40.3% increase in good EM and PM keyword density over just using NGAME. However, the key challenge is serving two models in parallel requires 2x additional hardware accelerators such as GPUs to perform forward passes over the two models. As shown in table 1, HEARTS effectively overcomes this problem by providing similar gains without additional GPU resources. Specifically, HEARTS achieves 97.59% and 97.68% of the EM and PM good keyword densities of the CLOVER-NAR + NGAME ensemble at half the GPU cost.

### Table 4: Results of various extreme classification algorithms along with our proposed HEARTS approach on the WikiSeeAlsoTitles-320K and AmazonTitles-131K datasets

| Method/Metrics | WikiSeeAlsoTitles-320K | AmazonTitles-131K | WikiSeeAlsoTitles-320K | AmazonTitles-131K |
|----------------|------------------------|------------------|------------------------|------------------|
|                | PSP@1 | PSP@3 | PSP@5 | PSN@1 | PSN@3 | PSN@5 | P@1 | P@3 | P@5 | N@1 | N@3 | N@5 | R@100 |
| Astec          | 13.69 | 15.81 | 17.50 | 13.69 | 15.56 | 16.75 | 22.72 | 15.12 | 11.43 | 22.72 | 22.16 | 22.87 | - |
| AttentionXML   | 9.45  | 10.63 | 11.73 | 9.45  | 10.45 | 11.24 | 17.56 | 11.34 | 8.52  | 17.56 | 16.58 | 17.07 | - |
| XT             | 8.99  | 10.52 | 11.82 | 8.99  | 10.33 | 11.26 | 17.04 | 11.31 | 8.60  | 17.04 | 16.61 | 17.24 | - |
| DECAF          | 16.73 | 18.99 | 21.01 | 19.18 | 20.75 | 25.14 | 16.90 | 12.86 | 25.14 | 24.99 | 25.95 | - |
| ECLARE         | 22.01 | 24.23 | 26.27 | 22.01 | 24.46 | 25.93 | 29.35 | 19.83 | 15.05 | 29.35 | 29.21 | 30.20 | - |
| GalaXC         | 19.77 | 22.25 | 24.47 | 19.77 | 21.70 | 23.16 | 27.87 | 18.75 | 14.3  | 27.87 | 26.84 | 27.60 | - |
| NGAME-M1       | 27.41 | 28.86 | 30.61 | 27.41 | 30.63 | 31.33 | 31.23 | 31.19 | 32.17 | 54.08 |
| HEARTS (DR)    | 28.46 | 30.29 | 32.38 | 28.46 | 30.51 | 32.10 | 31.72 | 16.57 | 31.74 | 31.87 | 32.97 | 54.08 |

Table 4: Results of various extreme classification algorithms along with our proposed HEARTS approach on the WikiSeeAlsoTitles-320K and AmazonTitles-131K datasets

4.5 Results & Discussion: Short-text Benchmarks

We present the results on the short-text benchmarks in table 4. We compare our approach with leading extreme classification and DR methods such as ECLARE [24], GalaXC [35], Astec [9], DECAF [23], AttentionXML [40], XT [38], LightXML [15] along with NGAME [8]. We observe that NGAME performs best among all the baselines in both datasets by a significant margin. Further, we see that our proposed HEARTS-DR approach consistently outperforms the strong NGAME baseline on all the metrics and datasets. In particular, we observe greater gains over NGAME in the propensity weighted metrics, e.g., we obtain an increase of 5.78% in PSP@5 whereas a 2.39% increase in P@5 on the WikiSeeAlsoTitles dataset. This indicates that HEARTS’s gains are more prominent on the rare labels, which could be more rewarding in real-world scenarios. In addition to precision, HEARTS also improves the Recall at 100 by over 3%, pointing out that the Non-autoregressive NLG objective, when included as an auxiliary loss with standard contrastive loss, helps improve the retrieval performance.

5 CONCLUSION

In this work, we focused on the problem of retrieving high-quality EM and PM bid keywords given user search queries. We studied two different approaches to this problem, viz., Dense Retrieval and Non-autoregressive NLG. We showed that the two methods are able to retrieve different sets of high-quality keywords for the same query, i.e., their retrieval performance is additive. However, a simple ensemble of these would require twice the compute resource and can be expensive to serve queries online at the web-scale. We, therefore, proposed HEARTS, a multi-task framework that solves this problem using the two approaches (DR and NLG) in one model. We showed that HEARTS achieves around 40% higher good bid keyword density than the baseline approaches while performing on-par with the ensemble of DR and NLG that requires twice the GPU compute. Finally, we showed that the DR component of HEARTS performs better than the DR baselines on both public and private datasets.
