Generalizing Discriminative Retrieval Models using Generative Tasks

Binsheng Liu
RMIT University
Melbourne, Australia
binsheng.liu@rmit.edu.au

Hamed Zamani
University of Massachusetts Amherst
Amherst, United States
zamani@cs.umass.edu

Xiaolu Lu
Microsoft
Melbourne, Australia
xiaolu_lu@yahoo.com

J. Shane Culpepper
RMIT University
Melbourne, Australia
shane.culpepper@rmit.edu.au

ABSTRACT

Information Retrieval has a long history of applying either discriminative or generative modeling to retrieval and ranking tasks. Recent developments in transformer architectures and multi-task learning techniques have dramatically improved our ability to train effective neural models capable of resolving a wide variety of tasks using either of these paradigms. In this paper, we propose a novel multi-task learning approach which can be used to produce more effective neural ranking models. The key idea is to improve the quality of the underlying transformer model by cross-training a retrieval task and one or more complementary language generation tasks. By targeting the training on the encoding layer in the transformer architecture, our experimental results show that the proposed multi-task learning approach consistently improves retrieval effectiveness on the targeted collection and can easily be re-targeted to new ranking tasks. We provide an in-depth analysis showing how multi-task learning modifies model behaviors, resulting in more general models.

CCS CONCEPTS

• **Information systems → Retrieval models and ranking**.

KEYWORDS

discriminative models, generative models, multi-task learning

ACM Reference Format:
Binsheng Liu, Hamed Zamani, Xiaolu Lu, and J. Shane Culpepper. 2021. Generalizing Discriminative Retrieval Models using Generative Tasks. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3442381.3449863

1 INTRODUCTION

Developing efficient and effective retrieval models has been at the core of information retrieval (IR) research since the 1960s [8]. Early models, such TF-IDF [40], relied on heuristics derived from statistical properties of documents contained in a collection. The well-known probability ranking principle (PRP) of Robertson [38] provided a theoretical foundation to these early ideas using probability theory. PRP frames a retrieval model as being an estimate of the probability of a document D being relevant (R) to the query Q, i.e., $\Pr(R|Q,D)$, where R is a binary random variable. This probability can be estimated using a *discriminative model* or a *generative model.*

Discriminative models for retrieval were first proposed by Fox [15] and later adopted in a number of successful learning to rank models, including SVM-based and decision tree-based modeling approaches [5, 19]. More recent neural ranking models in this category have also be proposed [12, 17, 29]. Generative models have also been used by a number of well-known retrieval models. For example, classic probabilistic retrieval models such as BM25 [37] are based on a document generation assumption. In contrast, the statistical language modeling, which is the basis of models such as query likelihood [35], are based on a query generation assumption.

Therefore, it is clear both discriminative and generative approaches have merits, and can potentially surface valuable signals which can be used to improve retrieval effectiveness. In this work, we show how both of these approaches can be used together to produce more effective retrieval models. More specifically, we are interested in verifying the following hypothesis:

Joint discriminative and generative retrieval modeling leads to more generalized, and hence more effective retrieval models.

To validate our hypothesis, we introduce a novel multi-task learning framework in which the tasks include both discriminative and generative relevance models. In the discriminative tasks, the model maximizes the likelihood of predicting the relevance labels given queries and documents, while in the generative tasks, the model is optimized to generate queries/questions given documents.

Developing a neural network architecture for both discriminative and generative modeling of relevance is challenging, because generative models rely heavily on encoder-decoder based architectures, while discriminative models typically require only an encoding component. We investigate two approaches to joint modeling: (1) an encoder-only architecture in which the generative tasks are modeled using input masking. For example, query generation is modeled by masking the query in the input of the network and using a maximum likelihood objective to predict the query tokens associated with the masked tokens; (2) an encoder-decoder Transformer architecture [42] in which the discriminative relevance modeling is implemented by feeding documents and queries to encoder and decoder inputs respectively, and using the cross-attention mechanism between encoder and decoder components (also known as the encoder-decoder attention) to learn a relevance score for a query-document pair.

This paper addresses the following research questions by conducting extensive experiments with the MS MARCO Passage Re-Ranking [4] and the 2019 TREC Conversational Assistance Track (CAsT) [11] datasets:

RQ1 Do generative tasks improve discriminative retrieval models?
RQ2 Which neural network architectures (i.e., encoder-only or encoder-decoder) produce more effective joint discriminative and generative relevance models?
RQ3 Are the resulting models easily transferable to other retrieval tasks?

In summary, our experimental results support our hypothesis that discriminative neural ranking models can be generalized using generative tasks and importantly, does not rely on a specific architectural configuration. This generalization leads to significant improvement across models and collections for both precision- and recall-oriented metrics. Our experiments also highlight the significant impact of such a joint modeling on other task-transfer scenarios.

2 RELATED WORK

In this section, we first review combining generative and discriminative retrieval models, followed by neural ranking models derived from transformers. Finally we review multi-task learning in IR.

Generative and Discriminative Retrieval Models. The probability ranking principle (PRP) proposed by Robertson [38] formally describes the retrieval and ranking task as estimating the probability a document being relevant to a query. Both discriminative [5, 15, 19] and generative models [35, 37] have been explored in the literature to solve this problem previously. Classic models such as BM25 [37] and query likelihood [35] can be categorized as a generative model. BM25 is based on the document generation assumption and the query likelihood model is based on a generative query assumption. Models fall in this category often estimate relevance indirectly using Bayesian rule ($Pr(R|Q,D) \propto P(Q|D,R)$). Discriminative models estimate the relevance based on the query and document, which directly learn the decision boundary between relevant and non-relevant. For example, classic learning-to-rank models such as Rank-SVM [19] and recently developed neural ranking architectures can be considered as discriminative models [12, 17, 29].

Most retrieval approaches focus primarily on improving ranking effectiveness by optimizing either a discriminative or a generative retrieval model. IRGAN [44] also includes a discriminative and generative component which improves retrieval effectiveness using a Generative Adversary Network (GAN) [16]. The IRGAN model formalizes the retrieval problem as a min-max game where the discriminative and generative models compete with each other – the generative model estimates the relevance distribution over signals using the discriminative model and then the discriminative model uses the output from the generative model to produce a better estimate of document rankings. In the follow-up work, Zou et al. [45] derive a theoretical analysis that connects the query reformulation and the document ranking problem using a game-theoretical approach, which models the two tasks as a general-sum game and a partnership game, respectively. Although related, this line of work differs from ours in two key aspects: first, we rely primarily on multi-task learning where auxiliary tasks are often considered as complementary to the main task, instead of a "competitor" in a GAN; second, tasks in GAN-based approaches are strongly coupled. The discriminator must be carefully designed to compete against the generator and the tasks involved are restricted to two due to the design (one for discriminator and one for generator). In our framework, any number of loosely related tasks can be easily incorporated to improve the main task.

Neural Ranking Models. Neural ranking models, which are capable of learning representations of both query and documents, have been studied extensively in recent years.See Mitra et al. [27] for instance for a detailed review on this topic. In this work, we rely heavily on transformer-based language architectures [13, 22, 42]. Recent work on transformers pretrained on large corpora have led to remarkable progress in a variety of natural language tasks and are now routinely applied in many IR related tasks [4, 7, 11, 43]. These models are rich in general linguistic knowledge [6], and can be beneficial to IR. Using a transformer ranking model to maximize ranking effectiveness after an initial filtering stage using retrieval method such as BM25 is now common practice in IR.

As an encoder-only architecture, BERT [13] has arguably been one of the most successful neural models adapted to the ranking task. It uses only the encoder component of a transformer architecture [42] pretrained with a masked language modeling objective. BERT uses a bidirectional attention mechanism which learns relationships between words and contexts. In IR, BERT based models have consistently produced competitive approaches IR tasks such as MS MARCO passage (according to disclosed models) and document ranking [4], TREC 2019 [7], and CAsT 2019 [11], and generally require only a moderate amount of fine-tuning. Fine-tuning BERT is straight-forward. For example, Nogueira and Cho [29] directly apply the fine-tuning approach described by Devlin et al. [13] to produce a competitive system. They cast ranking as sentence pair classification and fine-tune BERT using the MS MARCO training data. Dai and Callan [10] employ the same fine-tuning approach and observed significant improvements on other commonly used IR test collections such as ROBUST and ClueWeb09-B. In more recent work, Nogueira et al. [32] propose a further refinement using a three-stage re-ranking system based on BERT with a modified approach to fine-tuning. In contrast to the encoder-only approach, encoder-decoder architectures are rarely applied in ranking tasks. Nogueira et al. [30] explore the use of an encoder-decoder transformer T5 [36] for ranking. By casting relevance prediction as text generation, the model is trained to predict "True" or "False" literals given a query-document pair. The documents are then re-ranked based on the probability assigned to "True" token. However, applying encoder-decoder models generally require modifications to model training and incur additional retraining costs as the models tend to be much larger than an encoder-only architecture. It is also unclear exactly how an encoder-only architecture and an encoder-decoder architecture differ in terms of retrieval effectiveness. We include a detailed study in Section 6.4 that explores this question further.

Multi-Task Learning in IR. A large body of work [1–3, 24, 25, 28, 39] explores applying multi-task learning (MTL) to retrieval tasks. Broadly speaking, these approaches can be categorized into two types. Models in the first category tries to learn similar functions for multiple tasks [24, 28, 39]. For example, Nishida et al. [28] consider both extractive question answering and document ranking tasks, which can both be cast to classification tasks; both recommendation and retrieval tasks formulated by Liu et al. [24] are ranking tasks.

Studies in the second category learn common representations and adapt the learned model to other domain-specific problems or heterogeneous tasks [1–3, 25]. For example, Bai et al. [3] leverage the MTL approach to learn "super features" which can be applied to search tasks in different domains; the method proposed by Liu et al. [25] learns a general representation by jointly learning from both query classification and document ranking tasks; work has also been done in session-based retrieval [1, 2], where the goal is to represent query, document and users' context using MTL. In this work, the core idea is to select supplementary tasks from one category of tasks (generative) and use it to improve models in another category (discriminative). More specifically, our main focus is the *retrieval model* instead of representation learning, which may enable much more flexibility when selecting auxiliary tasks to complement the model being targeted.

3 BACKGROUND: TRANSFORMER ATTENTION

A full Transformer [42] has two independent components: encoders and decoders. The key difference is how they model dependencies which results in different attention mechanisms. In this section, we briefly review the attention mechanisms inside transformers and highlight their differences as the background of our approach.

Encoder: Bidirectional Attention. An encoder transformer is broadly used for classification, regression, and other related tasks. The core component of an encoder is the bidirectional attention mechanism as illustrated in Figure 1a. Using ranking as an example, when we feed the concatenation of a query and a document into an encoder, token embeddings are updated according to the context of the entire sequence. Bidirectional attention can take full advantage of the information not only within a query and a document but between them.

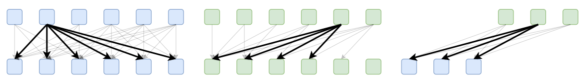

(a) Bidirectional **(b)** Unidirectional **(c)** Cross

Figure 1: An encoder implements bidirectional attention. A decoder implements unidirectional attention and cross attention. Cross attention bridges an encoder to a decoder.

Decoder: Unidirectional and Cross Attention. Encoder-decoder transformers have decoders in addition to encoders. The encoder-decoder transformers are primarily used in sequence-to-sequence (seq2seq) tasks which is an abstraction that maps one sequence of text to another. Examples are summarization, machine translation, query generation, and question answering. A source sequence is fed into the encoder. Then the decoder regressively produces new tokens conditioned on the source and on past tokens that were generated. The generation process often repeats several times until an end-of-sentence identifier is encountered or a pre-defined length limitation is reached. Formally, given the source sequence X and a target sequence Y, the probability distribution of the next token is conditioned on X and past tokens $Y_{<t}$.

$$P(Y|X)=\prod_{t}^{|Y|}P(y_t|Y_{<t},X) \tag{1}$$

The regressive nature of the model is reflected in the attention design of the decoders. Within an decoder, the self-attention is causal, or unidirectional, as in Figure 1b. A token can only attend to past tokens. This property corresponds to conditioning on $Y_{<t}$ in Eq 1. Following the unidirectional attention is cross attention (Figure 1c) where the model conditions on the source input X as in Eq 1. The various attention mechanisms illustrate the fundamental difference between an encoder transformer and a decoder transformer.

4 JOINT DISCRIMINATIVE AND GENERATIVE RETRIEVAL USING MTL

In this section, we introduce our GDMTL (Joint Discriminative and Generative Retrieval Model with Multi-Task Learning) framework for improving discriminative retrieval models using generative tasks in Sec 4.1, then in Sec 4.2 and Sec 4.3 we describe how we implement each component of that general framework using Transformer networks and unify the different requirements of discriminative and generative models.

4.1 GDMTL Framework

The GDMTL framework combines the ideas of discriminative and generative retrieval models using multi-task learning. In more detail, the framework simultaneously optimizes two different objectives: (1) one objective for a discriminative retrieval model, and (2) one set of objectives for generative tasks. GDMTL may contain multiple generative tasks. In the next section, our experiments will provide empirical evidence that two generative tasks provides better generalization for the discriminative retrieval model. We now formalize the GDMTL framework.

Problem formulation Let $\mathcal{T}=\{(q_1,D_1,R_1),(q_2,D_2,R_2),\cdots,(q_n,D_n,R_n)\}$ be the training set for a ranking task with n training queries, in which q_i denotes the i^{th} query, and $D_i=\{d_{i1},d_{i2},\cdots,d_{im_i}\}$ denotes the set of documents for q_i in the training set. R_i is a set containing ground truth relevance judgments, such that R_{ij} represents the relevance judgment for the query-document pair of (q_i,d_{ij}). This task trains a ranking model that re-ranks the documents in a given candidate document set D' for a test query q'.

The GDMTL overview The GDMTL framework consists of a discriminative retrieval model M_d (parameterized by θ_d) and a number of generative models $M_g : g \in G$ (each parameterized by θ_g), where G is a set of all generative tasks that enrich the discriminative retrieval model. Without loss of generality, we assume that relevance labels are binary, thus M_d estimates the probability of the document $d_{ij} \in D_i$ being relevant to the query q_i using $P(R=1|q_i,d_{ij})=M_d(q_i,d_{ij};\theta_d)$, where R is a binary random variable. This can be easily extended to graded relevance labels. With the ground truth relevance labels R_i, the parameters θ_d are learned by minimizing a loss $L_d(\theta_d;q_i,D_i,R_i)$. The total loss function is computed by averaging L_d for all queries in $\mathcal{T}$.

In contrast, each generative model $M_g : g \in G$ estimates the probability of a target text t being generated from a source text s: $P(t|s;\theta_g)=M_g(s;\theta_g)$. The parameters θ_g are optimized using a different loss $L_g(\theta_g;t,s)$. The question now is: How are t and s related to the training set $\mathcal{T}$? One reasonable approach, which is mainly used throughout this paper, is to use relevant documents as source texts and queries as target texts. This casts the problem to a query generation retrieval model, similar to query likelihood [35] and doc2query [31]. In summary, one way of modeling this component is to compute the query generation probability $P(q_i|d_{ij}, r_{ij}=1;\theta_g) = M_g(d_{ij};\theta_g, r_{ij}=1)$. Thus, the loss function of this part would be $L_g(\theta_g;q_i,d_{ij},r_{ij})$. As discussed previously, we have designed GDMTL such that it can optimizes multiple generative tasks. Multiple generative tasks in addition to query generation can be used, such as question generation based on QA data (e.g., generating questions from answer documents) and anchor text generation based on a hyperlink graph from Web data.

Using multi-task learning, parameters are split into shared and independent model-specific parameters. Therefore, θ_d and $\theta_g : g \in G$ share the parameters θ_{sh}. The resulting loss function is:

$$L = w_d L_d(\theta_d) + \sum_{g\in G} w_g L_g(\theta_g) \tag{2}$$

where w_d and $w_g : g \in G$ are the weights assigned to the respective losses.

Implementing the Loss The discriminator loss L_d can be modeled as either a point-wise, pair-wise, or list-wise loss function. Without loss of generality, in our experiments, Hinge loss is used for pair-wise ranking. For a pair of document candidates $d_{ij}, d_{ik} \in D_i$ for the query q_i, the discriminator loss function is defined as follows:

$$\begin{aligned} &L_d(\theta_d;q_i,D_i,R_i) \\ &= \frac{1}{Z} \sum_{\substack{1\le j,k\le m_i \\ r_{ij}\neq r_{ik}}} \max\{0, \epsilon - \text{sign}(r_{ij}-r_{ik}) \\ &\qquad\qquad (M_d(q_i,d_{ij};\theta_d) - M_d(q_i,d_{ik};\theta_d))\} \end{aligned} \tag{3}$$

where Z is a normalization factor and is equal to the number of training pairs for the query q_i. ϵ is a hyper-parameter for the Hinge loss and is set to 1 for binary relevance labels.

The generative loss $L_g : g \in G$ is defined using a cross entropy loss function as in the seq2seq model [41], and is equivalent to maximizing the likelihood of generating the target sequences observed in the training data. The generative loss for a query generation task is:

$$L_g(\theta_g;q_i,D_i,R_i) = -\frac{1}{Z'} \sum_{\substack{1\le j\le m_i \\ r_{ij}=1}} \Pr(q_i|d_{ij}, r_{ij}=1;\theta_g) \tag{4}$$

where Z' is a normalization factor and is equal to the number of relevant documents for the query q_i. In the above loss function, $\Pr(q_i|d_{ij}, r_{ij}=1;\theta_g)$ is computed as follows:

$$\Pr(q_i|d_{ij}, r_{ij}=1;\theta_g) = -\sum_{t=1}^{|q_i|} \log \Pr(q_i^t|q_i^1, q_i^2, \cdots, q_i^{t-1}, d_{ij}) \tag{5}$$

where q_i^t denotes the t^{th} token in the query. This loss function estimates the likelihood of each target token being generated given the input and all the previous tokens in the ground truth. These probabilities are produces using the underlying model M_g.

Balancing Loss As shown in Equation (2), the contributions of each task on updating the shared parameters θ_{sh} is controlled by the weights w_d and $w_g : g \in G$. A straightforward approach to assign loss weights is to treat them as hyper-parameters and tune them on a held-out validation set. However, it is expensive and sometimes impractical to exhaustively explore the parameter space when the training cost is high and there is more than one task, which reinforces the importance of automatically learning these parameters.

In this work, the Uncertainty [21] weighting scheme is used to automatically learn the weights. This method models the *homoscedastic* uncertainty of tasks, which represents the confidence in the contributions from different tasks and is independent of the input data. Intuitively, if a task has high uncertainty, our confidence is lower, and therefore the contribution to the joint loss is reduced, and so on. We adopt a variation of the Uncertainty method by Liebel and Körner [23] which has a modification to the regularization to avoid negative loss values. This makes Equation 2:

$$\begin{aligned} L_{\text{uncertainty}} = &\frac{1}{2\sigma_d^2} L_d(\cdot) + \log(1+\sigma_d^2) + \\ &\sum_{g\in G} \frac{1}{2\sigma_g^2} L_g(\cdot) + \log(1+\sigma_g^2) \end{aligned} \tag{6}$$

where σs are learnable parameter modeling the uncertainty of the models. When the loss is high, the uncertainty is high (but penalized by $\log(1+\sigma^2)$ for becoming too high), leading to low contribution of the loss, avoiding the gradients of this model dominating the training, and vice versa. Please refer to Kendall et al. [21] for further rationale in these formulations.

4.2 Architecture I: Encoder-Only GDMTL

The proposed GDMTL framework can be implemented using two different general architectures. The architectural choice of multi-task learning models affects how the shared parameters are optimized. In this subsection, we show how to implement M_d and M_g in a unified encoder architecture. M_d can be implemented using a neural network encoder that learns a representation for a query q_i and a document d_{ij}. This representation is then used to produce a relevance score for the given query-document pair. For example, a common approach is to concatenate query and document tokens using a beginning and separation token and feed it to a pretrained BERT variant [13] (i.e., the encoder). The representation produced by BERT for the beginning token is then fed to a fully-connected layer to output the relevance score [29, 34].

Other Challenges. Although an encoder-only architecture fits well with a discriminative ranking model, modeling generative tasks with no decoder is not straightforward since there is no longer an autoregressive component for text generation. To address this issue, we adapt the idea of predicting masked input, similar to that of masked language model (MLM) training as in BERT [13]. More specifically, M_g can be modeled by masking all input query tokens and using the associated output representations to predict them. However, there is an important limitation when using MLM – it does not scale when using long spans. As a point of reference, Joshi et al. [20] proposed spanBERT which extends BERT by masking contiguous random spans. In their work, the mean span length is 3.8 tokens. However, the average number of words (before WordPiece tokenization) of MS MARCO training queries is 7.4. We have conducted experiments using MLM and found that training converges prematurely, and has reduced overall performance. The limitations of this approach for our task is intuitive in retrospect. Casting query generation to the MLM task is akin to asking the model to generate a few tokens without having access to the previous tokens that were generated. This can be further improved by combining ideas from seq2seq and MLM training. However, we cannot fully adapt a seq2seq training strategy because of the bidirectional attentions in BERT layers; otherwise, computing the loss function for generating each token would depend on the future tokens to be generated. A solution to this problem is to convert each text generation training instance to t training instance, where t denotes the number of tokens in the target text. In this case, the first training instance has all of the masked tokens for all target inputs and the loss function is only computed when generating the first token; the second training instance, on the other hand, would have the first target token as input with masked tokens for the rest and the loss function would be only computed for generating the second token; and so on.

We now present a new attention mechanism we call *mixed attention*, that theoretically produces in the same loss and gradient values, but is easier to train in practice.

Mixed Attention. To overcome the aforementioned difficulties, we propose a mix of bidirectional attention, unidirectional attention, and cross attention to support seq2seq tasks in our encoder architecture. Figure 2a shows how mixed attention combines the three attention mechanisms into one: the document tokens (light blue) can fully attend to themselves (bidirectional attention), the query tokens (light green) can attend to the past query tokens (unidirectional attention), and the document tokens (cross attention).

Formally, let D denote a document, the contextualized embeddings of token t of an document-query pair after mixed attention can be expressed as follows:

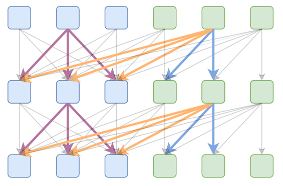
$$z_t = \begin{cases} \sum_{i=1}^{|D|} \frac{\exp(q_t^* k_i^*)}{\sum_{j=1}^{|D|} \exp(q_t^* k_j^*)} v_i^*, & \text{if } t \leq |D| \\ \sum_{i=|D|+1}^{t-1} \frac{\exp(q_t^* k_i^*)}{\sum_{j=1}^{t-1} \exp(q_t^* k_j^*)} v_i^*, & \text{otherwise} \end{cases} \quad (7)$$

where q^*, k^*, and v^* represent query, key and value vectors of the attention mechanism, respectively.

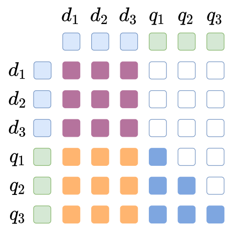
Mixed attention can be implemented by modifying the attention masks without changing the vectorized attention computations. This allows us to apply mixed attention to existing pretrained Transformer models. Properties of the special mask is shown in Figure 2b. The bidirectional attention mask is a 3×3 all-one matrix (purple), the unidirectional attention mask is a 3×3 lower triangular matrix (blue) and the cross attention mask is also a 3×3 all-one matrix (yellow). Combining them results in the mixed attention mask.

Now we consider how mixed attention can be stacked a transformers often contain multiple attention blocks. One important property of unidirectional attention is that it can be stacked arbitrarily in a decoder transformer without leaking labels. This property is retained in mixed attention, as shown in Figure 2a. The query tokens in the top layer represented with light green attend past query tokens in the middle layer which regressively attend solely to past query tokens in the bottom layer. Thus, mixed attention can be used safely in an encoder to imitate complete encoder-decoder behaviors.

Model Implementation using Mixed Attention. Using mixed attention, we can extend an encoder Transformer to implement discriminative and generative retrieval models using multi-task learning, as shown in Figure 3. For ranking, the model takes the concatenation of the document and query as input. An embedding produced from the "[CLS]" token is fed into a feed-forward layer to



(a) Mixed Attention of bidirectional (purple), cross (yellow) and unidirectional (blue) attention.



(b) Mixed Attention Mask. Each row represents a token attending to other tokens with filled boxes. For example, d_1 can attend to d_1 to d_3; q_2 can attend to d_1 to q_2.

Figure 2: Mixed attention imitates bidirectional, unidirectional and cross attention behaviors.

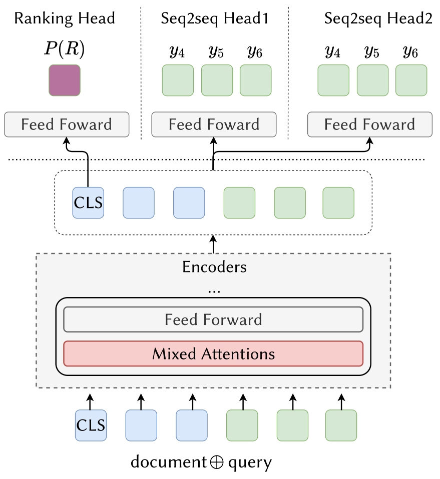


Figure 3: The MTL encoder combines ranking and sequence-to-sequence tasks. The key idea is to incorporate sequence-to-sequence tasks using mixed attention. The model fully attends a document but only attends backwards for a query, imitating the behavior of a decoder.

produce a score for the input pair (Ranking Head in Fig 3). For query generation, we also concatenate the document and the query as input into the model, but apply mixed attention instead of bidirectional attention to imitate encoder-decoder behavior. This model produces the next token probabilities for a target input using Seq2seq Head 1. The model can easily be extended by adding additional heads, as is now common practice in multi-task learning regimes. We refer to this implementation as GDMTL in later sections.

4.3 Architecture II: Encoder-Decoder GDMTL

In this subsection, we present an alternative solution to implement the GDMTL framework using an encoder-decoder architecture. It is less common to use such an architecture for ranking tasks, as encoder-only architectures are reasonable decoders and are most valuable for autoregressive generation tasks where each prediction depends

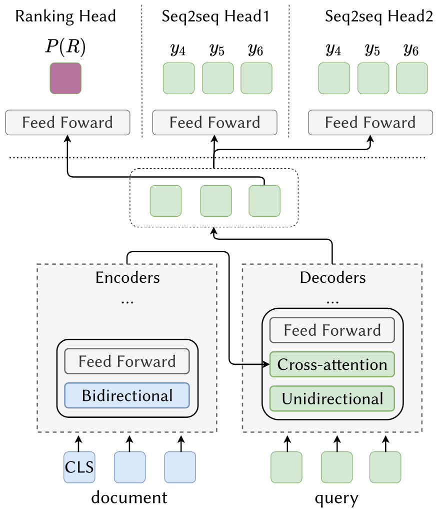


Figure 4: An MTL encoder-decoder architecture that combines ranking tasks and sequence-to-sequence tasks.

on the previous one. However, unlike an encoder-only architectures, encoder-decoders make it simpler to model generative models M_g.

That said, an encoder-decoder Transformer can also be used for ranking. As discussed in Section 2, Raffel et al. [36] feed the same concatenation of the query and the document to both the encoder and the decoder and fine-tune the model to predict "True" and "False" literals.

We also adapt an encoder-decoder Transformer with multiple attention heads, as shown in Figure 4. In our proposal, the model produces a score with the Ranking Head and sequences of generated text with the Seq2seq Heads. For ranking, the encoder takes the document as input and the decoder takes the query as input, and the prediction of the next token for the entire sequence is fed into a feed-forward layer. Note that, in contrast to an encoder Transformer, the first token cannot be regarded as a complete sentence representation given the decoder limitation of having only unidirectional attention. So, only the last input token has access to the entire sequence, and can be used to fine-tune the model. For query generation, the model produces a distribution over the entire vocabulary for each token in a target instance. Thankfully, we can use pretrained encoder-decoders, such as BART [22], and take advantage of self-supervision pretraining. We refer to this model as GDMTL_S in later sections.

We also propose a variation of BART for ranking. Instead of feeding separate documents and queries into the encoder and decoder, they are concatenated and feed to both the encoder and decoder. Then the model is fine-tuned as have described. This variation is referred to as GDMTL_C in later sections.

5 EXPERIMENTAL SETUP

5.1 Dataset

We use two datasets to evaluate our methods: MS MARCO and CAsT 2019.

MS MARCO Passage. For passage retrieval, we use MS MARCO, which consists of 1 million queries sampled from Bing search logs and 8.8 million passages extracted from web documents [4]. The queries are split into train, dev, and eval sets by the organizers. The training set contains around 532k relevant query-document pairs, the majority of which have only one relevance assessment per query (95% of training queries have a single passage assessment). The dev set contains 6,980 queries, 6,950 of which have one relevant passage. Model evaluation is performed using the dev set and 5-fold cross validation. It is not possible to use the eval set since the judgments are not publicly available.

CAsT 2019. We also use CAsT 2019, which is a conversational search task consisting of two document collections: MS MARCO and TREC CAR (Washington Post Collection was originally included but not used in the final evaluation by the organizers). Here we use the MS MARCO subset to evaluate the effectiveness improvement and use the TREC CAR subset to evaluate our model generalizability. The CAsT 2019 test collection contains 20 multi-turn sessions. Within a session, multiple queries are issued for an information need. We use the resolved queries provided by organizers in our experiments since our focus is on the ad-hoc retrieval task, and not co-reference resolution as in the conversational case. The resulting experimental dataset contains 173 queries and 2,983 relevant passages. We perform cross validation at the session level instead of query level as the queries are not i.i.d. due to the intentional session-level dependency. We use the `GroupKFold` algorithm from sklearn [1] to guarantee that queries in the same session are in the same fold. In each evaluation iteration, we use three folds for training, one for validation and one for testing.

MS MARCO QA. We also use the MS MARCO QA dataset in the question-answering task as an auxiliary generative dataset. This dataset has overlapping queries and relevant passages as found in the passage ranking task. In addition, it contains human crafted answers derived manually from relevant passages and we use them as the target of the generative task.

5.2 Task Setup

Retrieval Task We use the MS MARCO and CAsT 2019 dataset for the retrieval task. We use Anserini toolkit for indexing and first-stage retrieval, BM25 tuned for recall based on 1,000 randomly sampled queries from the training set. The MS MARCO corpus is enriched with DeepCT [9] for first-stage retrieval only. For each query we retrieve 1,000 documents for second-stage re-ranking. All of the ranking models we use in this work are summarized in Table 1.

Generative Tasks We now explore the use of two auxiliary tasks, which are query generation and question-answering. For the auxiliary query generation task, we take advantage of known query-document pairs using QREL data. That is, we use the training data in two different forms and combine them using MTL techniques. The query-document pairs are used for seq2seq tasks. The document is the source sequence and the query is the target sequence. Nogueira et al. [33] also used query-document pairs in a similar fashion, but their goal was to predict queries for passage/summary enrichment. We also use the MS MARCO QA dataset by converting the data into a seq2seq format for MTL training by concatenating the query and the relevant passage as the source sequence and use the human written answer as the target sequence, which is similar to conditioned summarization [14].

[1]https://scikit-learn.org

Table 1: Model notations.

Notation	Description
mono	monoBERT ranking model from Nogueira and Cho [29] and Nogueira et al. [32]
BERT	BERT ranking model (our implementation)
GDMTL	Our multi-task encoder model: ranking and query generation
GDMTL+	Our multi-task encoder model with 3 tasks: ranking, query generation, question answering
BART_S	BART ranking model using *separated* passage and query input
GDMTL_S	Our multi-task encoder-decoder model using *separated* passage and query input
BART_C	BART ranking model using *concatenated* passage and query input
GDMTL_C	Our multi-task encoder-decoder model using *concatenated* passage and query input

Table 2: Model inputs. P represents passage; Q represents query; A represents answer; ⊕ represents concatenation. Encoders require the sequences to be concatenated. Encoder-decoders have two ways to feed queries and passages for ranking, separately or concatenated.

	Ranking Input		Generative Input	
	Encoder	Decoder	Encoder	Decoder
mono	P⊕Q	-	-	-
BERT	P⊕Q	-	-	-
GDMTL	P⊕Q	-	P⊕Q	-
GDMTL+	P⊕Q	-	P⊕Q or P⊕Q⊕A	-
BART_S	P	Q	-	-
GDMTL_S	P	Q	P	Q
BART_C	P⊕Q	P⊕Q	-	-
GDMTL_C	P⊕Q	P⊕Q	P	Q

Task Conditioning. Task conditioning is a crucial component in achieving effective MTL. The approach we take is similar to T5 [36], where special task identifiers are added at the beginning of input sequences. For example, `rank:` is added for ranking tasks so the input for ranking becomes: `rank: <passage> <query>`. In this work, we use the identifier "`rank:`" for ranking, "`sum:`" for query generation, and "`answer:`" for question-answering. The specific text for task conditioning is arbitrary as long as it is unique for each task. During training, two or three heads are used depending on the number of generative tasks used as input. For inference, we ignore the seq2seq heads and use only the the ranking output layer.

5.3 Training Method

One training instance contains one query, one positive document and one negative document (and one answer for the query if we use the QA task). We summarize in Table 2 how we feed the three components into the models. The principles applied are intuitive. For encoders sequences need to be concatenated while for encoder-decoders there are two ways to feed queries and passages for ranking, separately or concatenated, as the names GDMTL_S and GDMTL_C suggest.

Using GDMTL as an example, we first use the concatenation of the document and the query as input and calculate ranking loss using Eq 3. Then we use the concatenation of the positive document-query pair only and instruct the model to use mixed attention and calculate the generative task loss using Eq 4. The two losses are weighted according to their uncertainty and summed up for back-propagation.

5.4 Model Implementation

In this experiment, we use BERT as the encoder-only architecture and BART as the encoder-decoder architecture, but our approach is easily amenable to any similar transformer architecture such as the ones made available by Hugging Face.[2] We implement a single task learning (STL) baseline and all MTL approaches using Transformers[3] and PyTorch library. All models are trained using mixed-precision floating point arithmetic on two NVIDIA V100-32GB GPUs. We use the AdamW [26] optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and weight_decay = $5e^{-5}$ for training. We have observed that training works best when using small learning rates, and have set our learning rate accordingly to $2e^{-5}$. We use a linear scheduler to adjust the learning rate stepwise with a one epoch warm-up and decays linearly to a minimum value of $1e^{-6}$. All models are trained for 10 epochs, and final epoch selection is a hyper-parameter decided during cross-validation.

For ranking tasks, the length of the input sequence is limited to 256 tokens. For seq2seq tasks, the length of the encoder input is limited to 256 tokens and the decoder input to 56 tokens. We do not use a maximum length of 512 tokens as the default, since the average length of the MS MARCO training passages is 91 terms and the average length of the training queries are around 6 terms. Setting the maximum length greater than 256 results in no measurable differences in effectiveness and increases training costs, so we have limited it. Due to data alignment constraints in GPUs, token padding tends to lower computational throughout but is ultimately discarded at aggregation time when using short input sequences. For other details, see the code repository for this work which is publicly available. [4]

Baseline Performance. Before diving into the results, we present a baseline implementation BERT using monoBERT [29, 32] as a reference since they are conceptually similar. We apply two-stage retrieval in our experiments: BM25 as the first stage ranker to retrieve top 1,000 passages, and then neural ranking models as the second stage ranker for re-ranking. Note that we use the base version of both BERT and BART, which have $110M$ and $139M$ parameters respectively, while monoBERT is derived using a"large" BERT model which contains $340M$ parameters.

Our new implementation of BERT significantly outperforms monoBERT for all effectiveness metrics we tested at the $p < 0.01$ level, which is shown in Table 3. There are several implementation differences between BERT and monoBERT, and we are unable to attribute the effectiveness differences to any one of these. Our best conjecture is that we have trained our model using a pair-wise loss while monoBERT used a point-wise. This outcome does not agree with the previous findings of Han et al. [18] who found only small

[2]https://huggingface.co
[3]https://github.com/huggingface/transformers
[4]https://github.com/binshengliu/gdmtl

Table 3: Retrieval performance of STL and MTL models on MS MARCO. △ and ▲ indicate statistical significance at $p < 0.05$ and $p < 0.01$ over BERT with Holm-Bonferroni correction, respectively.

	MRR		NDCG		RBP	
	10	100	10	20	0.5	0.8
BERT	0.384	0.393	0.448	0.471	0.177 +0.823	0.099 +0.901
GDMTL	0.392▲	0.401▲	0.454△	0.476△	0.182 +0.818▲	0.101 +0.899△
GDMTL+	0.394▲	0.403▲	0.458▲	0.480▲	0.182 +0.818▲	0.101 +0.899▲
BM25	0.244▼	0.256▼	0.299▼	0.324▼	0.108 +0.892▼	0.067 +0.933▼
mono	0.372▼	0.381▼	0.433▼	0.455▼	0.172 +0.828▼	0.096 +0.904▼

Table 4: Pair-wise win/tie/loss analysis on MS MARCO Dev set based on MRR@10, indicating the number of queries being improved, unchanged (within 10%), and hurt. The comparing base is listed in the headers.

	mono			BERT			GDMTL		
	W	T	L	W	T	L	W	T	L
BERT	1735	3775	1470		-			-	
GDMTL	1755	3805	1420	1375	4333	1272		-	
GDMTL+	1827	3764	1389	1445	4281	1254	1317	4459	1204

difference between the two approaches for the MS MARCO collection. Exploring the differences further is beyond the scope of this work as it is orthogonal to our primary aims. Our code is available if anyone wishes to explore it further.

These preliminary results show that competitive baselines are being used to benchmark our new approaches.

6 RESULTS AND ANALYSIS

In this section, we attempt resolve our original research questions. In Sec 6.1 we test if our approach improves retrieval effectiveness on two different test sets and a provide failure analysis. In Sec 6.2 we test if our approach can generalize well with different training and testing distributions. Then in Sec 6.3, we provide some other related analysis. Finally in Sec 6.4 we discuss the impact of implementing MTL when using different architectures.

6.1 Improving Ranking Effectiveness

RQ1. Do generative tasks improve discriminative retrieval models?

We first answer RQ1 by experimenting using MS MARCO and CAsT 2019 test queries, each of which have different properties. MS MARCO contains thousands of queries with shallow judgments while CAsT 2019 has fewer queries and deep judgments. MS MARCO also contains several related tasks such as Question-Answering and Query Categorization in addition to the ranking task. Results using query generation and the QA tasks as the auxiliary tasks are provided for MS MARCO, and include a further analysis using query categorization.

MSMARCO. We begin our analysis with a head-to-head comparison of single task and multi-task learning as described in the previous section using MS MARCO. Table 3 summarizes the performance comparisons when incorporating the query generation task into GDMTL. We found that the GDMTL and GDMTL+ models significantly outperform their STL counterpart BERT for every metric, showing that the addition of the generative task can improve the performance of a discriminative ranking model. In the next section, we will more exhaustively analyze *how* different generative tasks reinforce different aspects in the discriminative ranking model task, which translated to the improved performance we observe here.

When we consider the addition of a third task, such as the QA task, GDMTL+ shows even more improvement across all of the effectiveness metrics tested. The QA dataset reinforces the model by providing additional information that connect queries and the

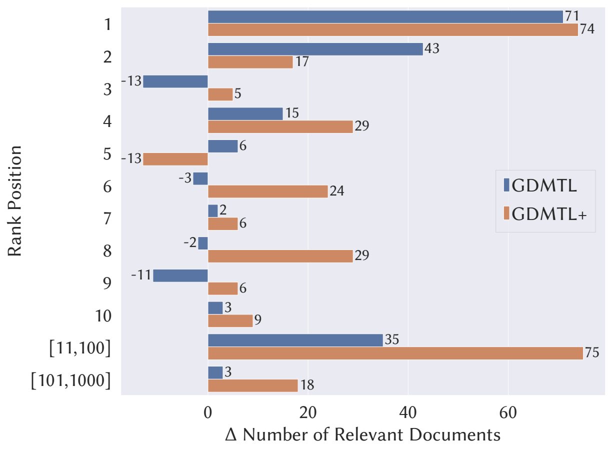


Figure 5: Rank position changes for relevant documents between BERT and MTL systems. Each bar on the right represents more documents added to a rank position. Conversely, bars on the left indicate that more documents moved from this position to another than were added. The more documents added to higher positions, the higher the overall effectiveness.

most important part of the passage. Consider the example query "what county is columbus city in", the passage "Columbus is a city in [...] Bartholomew County [...]. The population was [...]" which is very general description, and the answer "Columbus is a city in Bartholomew County." The final answer focuses the attention on the most important words in the passage and thus further strengthens contextual information in the model.

In Table 4 we show the MRR@10 comparison using monoBERT, BERT, GDMTL, and GDMTL+ as a pairwise win/tie/loss comparison. The trend is consistent across all of the comparisons. From top to bottom, more queries get improved than are degrade as more tasks are added. From left to right, it is also clear that when the base system is more competitive, it becomes harder to get any large improvements.

We also analyze all of the document ranking changes between BERT and MTL systems in Figure 5. Here, GDMTL and GDMTL+ rank 71 and 74 more documents respectively at position 1 than BERT. Note that BERT already achieves a perfect score for 26% of all queries, with nearly 50% of all queries retrieving a relevant passage in the top 3 positions, which means in practical terms that dramatic improvements for BERT are not possible as there are few opportunities to achieve big gains. Nevertheless, it is clear that that GDMTL and GDMTL+ do in fact consistently rank relevant documents at higher positions than BERT alone.

Caveats. We must also note that the residuals shown in Table 3 are unacceptably high, even when using RBP, $p = 0.5$ which is analogous to MRR. This suggests that many of the results returned in higher positions for each query are unjudged, and we can therefore not be sure they are not relevant. As there is generally a single positive judgment provided for each query, this should not be surprising, and of course applies to all experimental studies appearing in the literature using the MS MARCO collection. Nevertheless, residuals of this magnitude dictate that fine grained comparisons must be interpreted with caution. In this work, we are most interested in comparing relative trends between closely related models in order to understand how multi-task learning can be used to improve model quality, and not in achieving the most competitive result for any particular collection, and is therefore sufficient for our current needs. This is an important problem that does warrant further analysis in the future.

Per-Category Breakdown Analysis. MS MARCO also provides query categorization, so we can analyze where MTL exhibits the most benefit. Table 5 shows that Numeric, Location and Person queries tend to be highly effective with no MTL. These queries are relatively straightforward and can usually be resolved using the keywords. Thus they do not rely heavily on semantic understanding. Results for Entity queries are somewhat surprising. Therefore, we manually inspected a few of these queries. This qualitative analysis suggests that some of the answer passages are not as as easy to identify as one might think. For example, the entity query "what can help dogs sleep" also implies that passages on "why" dogs cannot sleep are also relevant. In this case, the answer preferred by the assessor were about "aches and pains" in dogs, with "aspirin" being the remedy proposed, with sleep problems being a symptom of the condition being discussed in the passage. Another entity query "highest dosage of aspirin" also cannot be simply answered with one keyword. The answer also depends contextual information such as age and any related health conditions. Ultimately, a single passage selected by the original assessor is known and why it was ultimately selected is open to interpretation, and requires caution, as there are clearly other passages being returned that may or may not also be relevant, as alluded to by the RBP residuals discussed above. It is clear the Entity queries seem be more difficult for this collection. The last type of query is a Description query. These queries often require longer answers and thus have the highest demand of semantic learning, for example "what does the chief administrator do".

Description queries benefit the most when using MTL, and is in line with our expectations. Enriching the model with query generation and question answering tasks strengthens semantic and contextual dependencies in the model. Small perturbations when training the model improve performance for complex queries.

CAsT 2019. Table 6 provides a summary level effectiveness comparison when applying our models on the new test collection, and more detailed the win/tie/loss analysis for CAsT 2019 is shown in Table 7. Note is not a corresponding QA dataset for CAsT 2019 queries so we could not test the GDMTL+ model. Nevertheless, the overall trend is consistent in MS MARCO albeit with larger margins. Since CAsT 2019 includes graded relevance judgments, the rank position comparison shown for MS MARCO may be less informative as measures such as NDCG tend to combine rank, grades, and gain functions such that the score is an aggregate of every document shift, and not just the highest ranking one. So, we have plotted the per query differences for NDCG@10 when using BERT and GDMTL instead, which is shown in Figure 6. This figure shows that the total number of wins with GDMTL is higher, and each of the differences tends to be larger on average.

Table 5: MRR@10 of STL and MTL models on MS MARCO broken down by query type. △ and ▲ indicate statistical significance at $p < 0.05$ and $p < 0.01$ over BERT with Holm-Bonferroni correction.

	Description (53.12%)	Numeric (26.12%)	Entity (8.81%)	Location (6.17%)	Person (5.78%)
BERT	0.369	0.391	0.344	0.473	0.443
GDMTL	0.377	0.395	0.361	0.489	0.441
GDMTL+	0.383△	0.395	0.351	0.489	0.439

Table 6: Retrieval performance of STL and MTL models on CAsT 2019. △ and ▲ indicate statistical significance with $p < 0.05$ and $p < 0.01$ over corresponding BERT with Holm-Bonferroni correction.

	MRR		NDCG		RBP	
	10	100	10	20	0.5	0.8
BERT	0.544	0.550	0.405	0.428	0.366 +0.286	0.317 +0.355
GDMTL	0.590△	0.595△	0.431△	0.451△	0.419 +0.236▲	0.341 +0.330△
BM25	0.474▽	0.479▽	0.331▼	0.346▼	0.331 +0.259	0.284 +0.361▽

Table 7: Pair-wise win/tie/loss analysis on CAsT 2019 based on NDCG@10, indicating the number of queries being improved, unchanged (within 10%), and hurt.

	BM25			BERT		
	W	T	L	W	T	L
BERT	87	39	47		-	
GDMTL	93	41	39	65	59	49

6.2 Improving Model Generalizability

RQ3. Are the resulting models easily transferable to other retrieval tasks?

In this section, we discuss preliminary results on task-based transfer learning. We use our model trained on MS MARCO and the queries from the CAsT 2019 collection, as shown in Table 8.

Again, care must be taken when interpreting these results. As shown in Table 9, queries share a common information need and have overlapping QRELs. A similar observation can be made in other tasks that use the MS MARCO collection, and is a point of discussion with the organizers. The collection was created and maintained by Microsoft, and in live production environments, duplicates and near

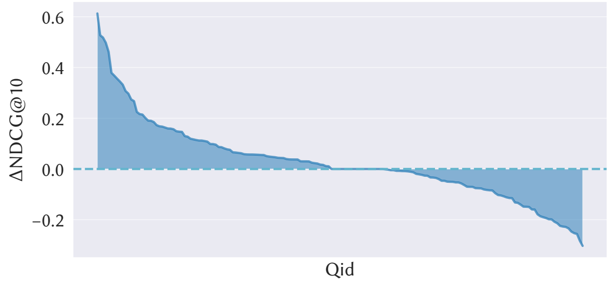


Figure 6: Per-query breakdown of NDCG@10 on CAsT 2019. Queries in descending order of score difference.

Table 8: Transferring models from MS MARCO to CAsT 2019. △ and ▲ indicate statistical significance with $p < 0.05$ and $p < 0.01$ over BERT with Holm-Bonferroni correction.

	MRR		NDCG		RBP	
	10	100	10	20	0.5	0.8
BERT	0.617	0.617	0.486	0.490	0.470 +0.230	0.392 +0.318
GDMTL	0.656	0.651	0.495	0.510	0.483 +0.231	0.402 +0.307
GDMTL+	0.686△	0.683△	0.498	0.498	0.514 +0.202△	0.409 +0.301
BM25	0.474▼	0.479▼	0.331▼	0.346▼	0.331 +0.259▼	0.284 +0.361▼

Table 9: CAsT 2019 evaluation queries and MS MARCO training queries share some information needs.

Collection	Topic	Rel Doc
CAsT 2019	69_6: What are the side effects of melatonin?	97921
MS MARCO	564795: what are side effects of melatonin pills	97921
CAsT 2019	67_8: What are anemia's possible causes?	883439
MS MARCO	556252: what are causes of very bad anemia	883439
CAsT 2019	31_7: What is the first sign of throat cancer?	7035854
MS MARCO	574369: what are the symptoms of throat cancer	7035854

duplicates are common, especially in very large training sets such as MS MARCO. At any rate, the performance of the transfer models seem to be superior to models that are trained independently for CAsT 2019. That is, all models tested showed similar advantages, and we are most interested in a relative comparison here. When compared head-to-head in identical scenarios, GDMTL and GDMTL+ are consistently more effective than BERT.

In Table 8, consistent improvements are observed but are only significant for MRR@10, MRR@100 and RBP $p=0.5$ for GDMTL+. The lack of significance for deeper metrics may be an artifact of the shallow judgment pool, the near replicates in training data, or both. There are two important factors to account for this phenomenon. First, 90% of the MS MARCO training data has only one relevant passage for training, while the CAsT 2019 queries have 17 relevant passages on average. Second, the MS MARCO judgments are binary, but the CAsT 2019 judgments are graded. Such discrepancies may also contribute to a reduced overall performance when directly transferring a model to a new task without any modification. We will explore this further in future work.

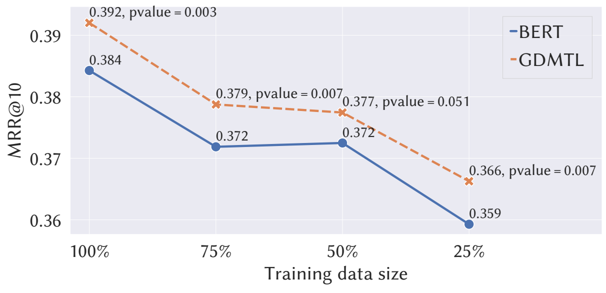


Figure 7: MTL consistently improves the model as the number of training instances is increased.

6.3 Additional Result Analysis

Impact of Training Data Size. We now turn our attention to understanding what impact the number of training instances has in our MTL model. In this analysis, we test our approaches using samples of 75%, 50%, 25%, and 12.5% of the original MS MARCO training data, while fixing all other hyperparameters. In order to minimize noise from sampling, we apply top down subset sampling. For example, we take 75% from the original set, and then take 66.7% from the 75% subset to create the 50% set and so on. We do not run the same experiment using CAsT 2019 as the total number of training instances is too small.

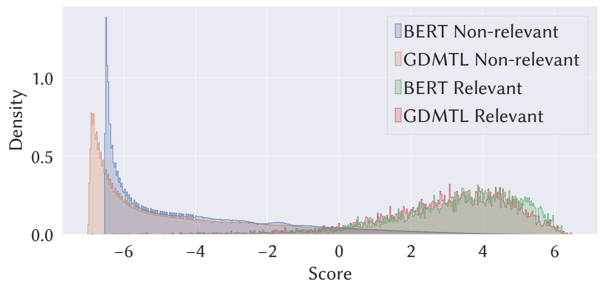
Results are shown in Figure 7. Regardless of training set size, GDMTL consistently outperforms BERT. The improvements are significant except when a 50% sample of training data was used. So, our initial experiments suggest that MTL can be benefit even in cases where the training data is limited.

Score Distributions. In term-based retrieval systems, raw retrieval scores across queries are often not comparable as they depend on query length. For example, query likelihood models assign a score to a query-document pair $P(Q|D) \propto \prod_{q \in Q} P(q|D)$ where q is a term in query Q and D is a document. Longer queries produce lower scores. However, attention-based neural ranking models apply normalization at attention aggregation time. Before the final prediction, the query and the document is projected into a sentence representation in a latent space, for example [CLS] of BERT. The contribution of each token to the sentence representation is softmax-normalized so the different query lengths do not change the distribution of the prediction scores.

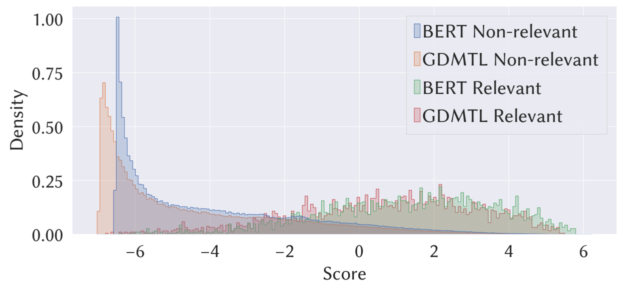
A plot of the histogram of scores produced using our models is shown in Figure 8. Relevant documents are mainly on the right and non-relevant documents are on the left. More importantly, the figure indicates how MTL has made the models more discriminative by assigning lower scores to non-relevant documents.

6.4 Impact of Architectures

RQ2. Which neural network architectures (i.e., encoder-only or encoder-decoder) produce more effective joint discriminative and generative relevance models?



(a) Score distributions for MS MARCO.



(b) Score distributions for CAsT 2019.

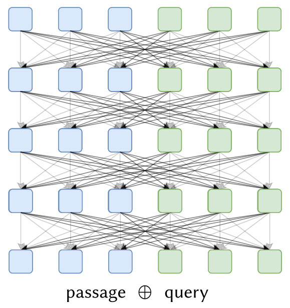
Figure 8: Score distributions for MS MARCO and CAsT 2019.

As we discussed in Section 4, not only the task balance strategy can affect the performance of an MTL framework, but also the underlying model architecture. In this experiment, we focus on two commonly used architectures for tasks: encoder-only (BERT) and encoder-decoder (BART). For the STL models, BERT has higher retrieval effectiveness than BART$_S$. There are two plausible reasons for this difference. First, BERT and BART are pretrained differently; second the encoder-decoder architecture must separate query-document interactions across the two layers as illustrated in Figure 9. In order to determine which of these contribute the most to the performance differences we have observed, we attempted to increase query-document interaction signals by feeding the concatenation of query and document into the BART encoder and decoder (shown as as BART$_C$ in Table 10). As we can see, inducing a stronger query document interaction results in improved performance when compared with BART$_S$, where interactions occur primarily in the decoder.

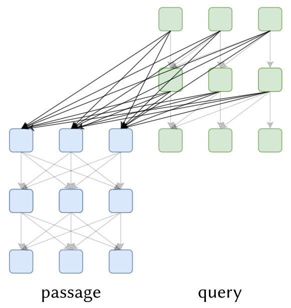
When including auxiliary query generation tasks, we can observe that GDMTL$_S$ can outperform BART$_S$ significantly, but GDMTL$_C$ also improves BART$_C$, but not as significantly. This is possibly caused by the different shape alignments required in the two tasks, as shown in Table 1. We also observed that both GDMTL (corrected $p = 0.0205$, 95% confidence interval [0.0038,0.0171]) and GDMTL$_C$ (corrected $p = 0.0260$, 95% confidence interval [0.0029, 0.0141]) also outperform GDMTL$_S$ significantly for MRR@10 when compared directly, showing architecture choices do have an important role when designing MTL models.

Table 10: Retrieval performance of STL and MTL models based on BART on MS MARCO. BERT and GDMTL are listed as references. $^{\vartriangle}$ and $^{\blacktriangle}$ indicate statistical significance with $p < 0.05$ and $p < 0.01$ over BART$_S$ with Holm-Bonferroni correction.

	MRR		NDCG		RBP	
	10	100	10	20	0.5	0.8
BART$_S$	0.370	0.380	0.436	0.460	0.170 +0.830	0.097 +0.903
GDMTL$_S$	0.382$^{\blacktriangle}$	0.392$^{\blacktriangle}$	0.445$^{\blacktriangle}$	0.470$^{\blacktriangle}$	0.176 +0.824$^{\blacktriangle}$	0.099 +0.901$^{\blacktriangle}$
BART$_C$	0.385$^{\blacktriangle}$	0.394$^{\blacktriangle}$	0.451$^{\blacktriangle}$	0.474$^{\blacktriangle}$	0.178 +0.822$^{\blacktriangle}$	0.100 +0.900$^{\blacktriangle}$
GDMTL$_C$	0.390$^{\blacktriangle}$	0.399$^{\blacktriangle}$	0.454$^{\blacktriangle}$	0.478$^{\blacktriangle}$	0.180 +0.820$^{\blacktriangle}$	0.101 +0.899$^{\blacktriangle}$
BERT	0.384$^{\blacktriangle}$	0.393$^{\blacktriangle}$	0.448$^{\blacktriangle}$	0.471$^{\blacktriangle}$	0.177 +0.823$^{\blacktriangle}$	0.099 +0.901$^{\blacktriangle}$
GDMTL	0.392$^{\blacktriangle}$	0.401$^{\blacktriangle}$	0.454$^{\blacktriangle}$	0.476$^{\blacktriangle}$	0.182 +0.818$^{\blacktriangle}$	0.101 +0.899$^{\blacktriangle}$



(a) Encoder attention.



(b) Encoder-decoder attention.

Figure 9: Encoder and encoder-decoder models require different interaction mapping (lines between different colors) when including passage and query pairing.

To summarize, BERT and BART$_C$ achieve similar performance when used only for ranking. Both architectures benefit from incorporating generative tasks but GDMTL achieves the best retrieval effectiveness in our experimental setup. Using mixed attention, GDMTL is able to incorporate two heterogeneous tasks using unified inputs. In contrast, GDMTL$_S$ is also able to combine multiple tasks in one model, but can suffer from the reduced query-document interactions due to the architectural requirements. GDMTL$_C$ leverages additional query-document interactions, but input shapes are more likely to be task specific. However, encoder-decoder transformers (GDMTL$_S$, GDMTL$_C$) in general may be less suitable than encoder transformers (GDMTL) if the primary goal is only ranking effectiveness, but may be a better choice for generation tasks, or if your overall goal is to use all of the task heads, and not just one.

7 CONCLUSIONS

We have proposed the GDMTL framework, which integrates generative and discriminative tasks via multi-task learning. Our framework exploits readily available generative tasks such as query generation and QA tasks to improve discriminative retrieval models. Our experiments have answered RQ1 affirmatively – generative tasks are indeed able to improve the performance of discriminative retrieval models.

Regarding RQ2, we have provided detailed comparisons between architectures which uncover additional insights that allow us to better understand how MTL model differences are affecting performance. Finally, for RQ3 we show that the models learned using our approach are easily transferable, which is beneficial in new tasks where little training data may be available. In future work, we will continue our analysis on how generative tasks modify model behaviors via attention and gradient-based analysis. We also plan to investigate how to get the best results with our framework when the primary application is a generative task rather than a discriminative one.

ACKNOWLEDGMENTS

This work was supported in part by the Australian Research Council's Discovery Projects Scheme (DP190101113) and in part by the Center for Intelligent Information Retrieval. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsors.

REFERENCES

[1] W. U. Ahmad, K. Chang, and H. Wang. 2018. Multi-Task Learning for Document Ranking and Query Suggestion. *Proc. ICLR* (2018), 14.

[2] W. U. Ahmad, K. Chang, and H. Wang. 2019. Context Attentive Document Ranking and Query Suggestion. In *Proc. SIGIR.* 385–394.

[3] J. Bai, K. Zhou, G. Xue, H. Zha, G. Sun, B. Tseng, Z. Zheng, and Y. Chang. 2009. Multi-task learning for learning to rank in web search. In *Proc. CIKM.* 1549–1552.

[4] P. Bajaj, D. Campos, N. Craswell, L. Deng, J. Gao, X. Liu, R. Majumder, A. McNamara, B. Mitra, T. Nguyen, M. Rosenberg, X. Song, A. Stoica, S. Tiwary, and T. Wang. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *arXiv:1611.09268 [cs]* (Nov. 2016). arXiv:1611.09268 [cs]

[5] C. J. C. Burges. 2010. *From RankNet to LambdaRank to LambdaMART: An Overview.* Technical Report. Microsoft Research.

[6] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning. 2019. What Does BERT Look At? An Analysis of BERT's Attention. *arXiv:1906.04341 [cs]* (June 2019). arXiv:1906.04341 [cs]

[7] N. Craswell, B. Mitra, E. Yilmaz, D. Campos, and E. M. Voorhees. 2020. Overview of the TREC 2019 Deep Learning Track. *arXiv:2003.07820 [cs]* (mar 2020). arXiv:2003.07820 [cs]

[8] B. Croft, D. Metzler, and T. Strohman. 2009. *Search Engines: Information Retrieval in Practice* (1st ed.). Addison-Wesley Publishing Company, USA.

[9] Z. Dai and J. Callan. 2019. Context-Aware Sentence/Passage Term Importance Estimation For First Stage Retrieval. *arXiv:1910.10687 [cs]* (Nov. 2019). arXiv:1910.10687 [cs]

[10] Z. Dai and J. Callan. 2019. Deeper Text Understanding for IR with Contextual Neural Language Modeling. In *Proc. SIGIR.* 985–988.

[11] J. Dalton, C. Xiong, and J. Callan. 2019. CAsT 2019: The Conversational Assistance Track Overview. In *Proc. TREC.* 10.

[12] M. Dehghani, H. Zamani, A. Severyn, J. Kamps, and W. B. Croft. 2017. Neural Ranking Models with Weak Supervision. In *Proc. SIGIR* (Shinjuku, Tokyo, Japan). 65–74.

[13] J. Devlin, M. Chang, K. Lee, and K. Toutanova. 2018. BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]* (Oct. 2018). arXiv:1810.04805 [cs]

[14] A. Fan, D. Grangier, and M. Auli. 2018. Controllable Abstractive Summarization. In *Proc. WNGT.* 45–54.

[15] E. A. Fox. 1983. *Extending the Boolean and Vector Space Models of Information Retrieval with P-Norm Queries and Multiple Concept Types.* Ph.D. Dissertation. USA. AAI8328584.

[16] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. 2014. Generative Adversarial Networks. *arXiv:1406.2661 [cs, stat]* (June 2014). arXiv:1406.2661 [cs, stat]

[17] J. Guo, Y. Fan, L. Pang, L. Yang, Q. Ai, H. Zamani, C. Wu, W. B. Croft, and X. Cheng. 2019. A Deep Look into neural ranking models for information retrieval. *Inf. Process. Manage.* (2019).

[18] S. Han, X. Wang, M. Bendersky, and M. Najork. 2020. Learning-to-Rank with BERT in TF-Ranking. *arXiv:2004.08476 [cs]* (April 2020). arXiv:2004.08476 [cs]

[19] T. Joachims. 2002. Optimizing Search Engines Using Clickthrough Data. In *Proc. KDD.* 133–142.

[20] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy. 2020. SpanBERT: Improving Pre-Training by Representing and Predicting Spans. *arXiv:1907.10529 [cs]* (Jan. 2020). arXiv:1907.10529 [cs]

[21] A. Kendall, Y. Gal, and R. Cipolla. 2018. Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics. In *Proc. CVPR.* 7482–7491.

[22] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. 2019. BART: Denoising Sequence-to-Sequence Pre-Training for Natural Language Generation, Translation, and Comprehension. *arXiv:1910.13461 [cs, stat]* (Oct. 2019). arXiv:1910.13461 [cs, stat]

[23] L. Liebel and M. Körner. 2018. Auxiliary Tasks in Multi-Task Learning. *arXiv:1805.06334 [cs]* (May 2018). arXiv:1805.06334 [cs]

[24] T. Liu, J. Huang, W. Zhang, Y. Sun, and H. Wang. 2018. Improving Entity Recommendation with Search Log and Multi-Task Learning. In *Proc. IJCAI.* 4107–4114.

[25] X. Liu, J. Gao, X. He, L. Deng, K. Duh, and Y. Wang. 2015. Representation Learning Using Multi-Task Deep Neural Networks for Semantic Classification and Information Retrieval. In *Proc. NAACL.* 912–921.

[26] I. Loshchilov and F. Hutter. 2019. Decoupled Weight Decay Regularization. *arXiv:1711.05101 [cs, math]* (Jan. 2019). arXiv:1711.05101 [cs, math]

[27] B. Mitra, N. Craswell, et al. 2018. An introduction to neural information retrieval. *Found. Trends in Inf. Ret.* 13, 1 (2018), 1–126.

[28] K. Nishida, I. Saito, A. Otsuka, H. Asano, and J. Tomita. 2018. Retrieve-and-Read: Multi-Task Learning of Information Retrieval and Reading Comprehension. In *Proc. CIKM.* 647–656.

[29] R. Nogueira and K. Cho. 2019. Passage Re-Ranking with BERT. *arXiv:1901.04085 [cs]* (jan 2019). arXiv:1901.04085 [cs]

[30] R. Nogueira, Z. Jiang, and J. Lin. 2020. Document Ranking with a Pretrained Sequence-to-Sequence Model. *arXiv:2003.06713 [cs]* (March 2020). arXiv:2003.06713 [cs]

[31] R. Nogueira and J. Lin. 2019. From doc2query to docTTTTTquery. https://cs.uwaterloo.ca/~jimmylin/publications/Nogueira_Lin_2019_docTTTTTquery-v2.pdf accessed 2020-01-21.

[32] R. Nogueira, W. Yang, K. Cho, and J. Lin. 2019. Multi-Stage Document Ranking with BERT. *arXiv:1910.14424 [cs]* (oct 2019). arXiv:1910.14424 [cs]

[33] R. Nogueira, W. Yang, J. Lin, and K. Cho. 2019. Document Expansion by Query Prediction. *arXiv:1904.08375 [cs]* (April 2019). arXiv:1904.08375 [cs]

[34] H. Padigela, H. Zamani, and W. B. Croft. 2019. Investigating the Successes and Failures of BERT for Passage Re-Ranking. *arxiv:1905.01758* (2019).

[35] J. M. Ponte and W. B. Croft. 1998. A Language Modeling Approach to Information Retrieval. In *Proc. SIGIR.* 275–281.

[36] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. 2019. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *arXiv:1910.10683 [cs, stat]* (Oct. 2019). arXiv:1910.10683 [cs, stat]

[37] S. Robertson and H. Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.* 3, 4 (April 2009), 333–389.

[38] S. E. Robertson. 1977. The probabilistic character of relevance. *Inf. Process. Manage.* 13, 4 (1977), 247 – 251.

[39] B. Salehi, F. Liu, T. Baldwin, and W. Wong. 2018. Multitask Learning for Query Segmentation in Job Search. In *Proc. ICTIR.* 179–182.

[40] G. Salton and C. Buckley. 1988. Term-Weighting Approaches in Automatic Text Retrieval. *Inf. Process. Manage.* 24, 5 (Aug. 1988), 513–523.

[41] I. Sutskever, O. Vinyals, and Q. V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Proc. NIPS.* 3104–3112.

[42] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. 2017. Attention Is All You Need. *arXiv:1706.03762 [cs]* (June 2017). arXiv:1706.03762 [cs]

[43] E. Voorhees, T. Alam, S. Bedrick, D. Demner-Fushman, W. R. Hersh, K. Lo, K. Roberts, I. Soboroff, and L. L. Wang. 2020. TREC-COVID: Constructing a Pandemic Information Retrieval Test Collection. *arXiv:2005.04474 [cs]* (May 2020). arXiv:2005.04474 [cs]

[44] J. Wang, L. Yu, W. Zhang, Y. Gong, Y. Xu, B. Wang, P. Zhang, and D. Zhang. 2017. IRGAN: A Minimax Game for Unifying Generative and Discriminative Information Retrieval Models. In *Proc. SIGIR.* 515–524.

[45] S. Zou, G. Tao, J. Wang, W. Zhang, and D. Zhang. 2018. On the equilibrium of query reformulation and document retrieval. In *Proc. ICTIR.* 43–50.