A Modern Perspective on Query Likelihood with Deep Generative Retrieval Models

Oleg Lesota
oleg.lesota@jku.at
Johannes Kepler University Linz
Linz Institute of Technology, AI Lab
Linz, Austria

Navid Rekabsaz
navid.rekabsaz@jku.at
Johannes Kepler University Linz
Linz Institute of Technology, AI Lab
Linz, Austria

Daniel Cohen
daniel_cohen@brown.edu
Brown University
Providence, R.I., USA

Klaus Antonius Grasserbauer
klaus.grasserbauer@jku.at
Johannes Kepler University Linz
Linz, Austria

Carsten Eickhoff
carsten@brown.edu
Brown University
Providence, R.I., USA

Markus Schedl
markus.schedl@jku.at
Johannes Kepler University Linz
Linz Institute of Technology, AI Lab
Linz, Austria

ABSTRACT

Existing neural ranking models follow the text matching paradigm, where document-to-query relevance is estimated through predicting the matching score. Drawing from the rich literature of classical generative retrieval models, we introduce and formalize the paradigm of *deep generative retrieval models* defined via the cumulative probabilities of generating query terms. This paradigm offers a grounded probabilistic view on relevance estimation while still enabling the use of modern neural architectures. In contrast to the matching paradigm, the probabilistic nature of generative rankers readily offers a fine-grained *measure of uncertainty*. We adopt several current neural generative models in our framework and introduce a novel generative ranker (*T-PGN*), which combines the encoding capacity of Transformers with the Pointer Generator Network model. We conduct an extensive set of evaluation experiments on passage retrieval, leveraging the MS MARCO Passage Re-ranking and TREC Deep Learning 2019 Passage Re-ranking collections. Our results show the significantly higher performance of the T-PGN model when compared with other generative models. Lastly, we demonstrate that exploiting the uncertainty information of deep generative rankers opens new perspectives to query/collection understanding, and significantly improves the *cut-off prediction* task.

CCS CONCEPTS

• **Information systems → Probabilistic retrieval models**.

KEYWORDS

Neural IR; generative ranking model; uncertainty; cut-off prediction

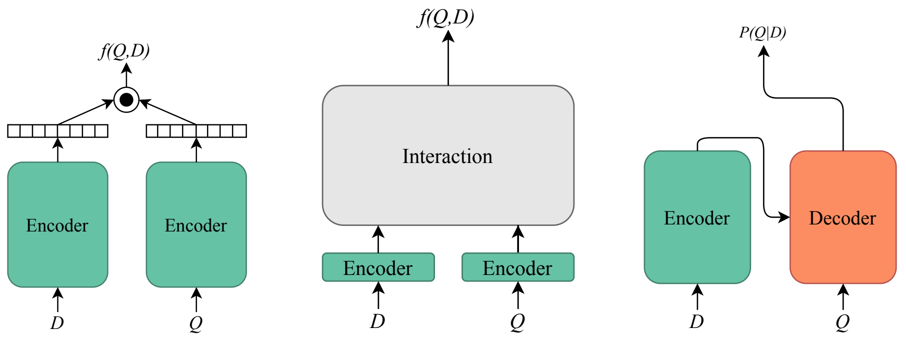
ACM Reference Format:
Oleg Lesota, Navid Rekabsaz, Daniel Cohen, Klaus Antonius Grasserbauer, Carsten Eickhoff, and Markus Schedl. 2021. A Modern Perspective on Query Likelihood with Deep Generative Retrieval Models. In *Proceedings of the 2021 ACM SIGIR International Conference on the Theory of Information Retrieval (ICTIR '21), July 11, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3471158.3472229

1 INTRODUCTION

Neural ranking models have yielded remarkable improvements to information retrieval (IR) by estimating a highly non-linear function of relevance of a query to a document. Arguably, all existing neural ranking models [7, 11, 12, 14, 25, 28, 33, 50] follow the text matching paradigm, where relevance is calculated as the predicted matching score of each document to a given query. In this sense, these neural ranking models appear to be the descendants of matching-based (or similarity-based) models such as the vector space model [42], where the model estimates the relevance score of a document D to a query Q by the matching function $f(Q, D)$. We refer to these models (whether neural or classical ones) as *matching models*.

A generative view on IR was first introduced by Ponte and Croft [36], where – unlike in matching models – relevance is expressed in terms of a conditional probability in a well-formed probabilistic framework. In particular, the query likelihood language model estimates relevance as the probability of the query being generated by a language model of the document, namely $P(Q|\Phi_D)$. This regime provides a powerful probabilistic framework to IR, and has been the base for numerous approaches (see Zhai [56] for further details). Our paper provides a modern perspective on the fundamental principle of the generative paradigm for IR through the recent advancements in deep generative models. We introduce and provide the theoretical foundations of *deep generative ranking models*, comprehensively study the characteristics and performance of the models' various architectural choices for passage retrieval, and show the immediate benefits of the probabilistic nature of this paradigm in providing more-than-relevance information.

Let us first discuss the overall architectural differences/similarities across various IR models, as highlighted in Figure 1. *Representation-based models* first encode query and document into separate embeddings, and then use these embeddings to calculate query-to-document relevance scores [18, 19, 44]. In *query-document interaction models*, the query-term-to-document-term interactions (in the form of similarities or attention networks) are used to create a final feature vector, and hence estimate the relevance score [7, 11, 12,



(a) Representation-based (e.g., DSSM)

(b) Query-Document Interaction (e.g., KNRM, BERT)

(c) Deep Generative Models

Figure 1: Schematic views of representation-based and query-document interaction models with representative examples of models (the two categories of the matching paradigm), and deep generative ranking models (the generative paradigm).

14, 18, 20, 25, 28, 34, 50]. In this category of models, large-scale pre-trained language models such as BERT [8] have shown significant improvements in retrieval performance [27, 31].

Deep generative ranking models (Figure 1c) view relevance estimation as the probability of generating the query, given the document. The models follow the sequence-to-sequence (seq2seq) encoder-decoder architecture [45]. They first encode the document, and then use the encoded embeddings to provide probability distributions over the space of possible queries at the output of the decoder. This framework in addition to effective estimation of relevance, provides a distinctive benefit in comparison with other retrieval models: the probabilistic nature of generative models enables the extraction of actionable information in addition to mere relevance scores. This probabilistic information for example can take the form of uncertainty estimates of the model. Such uncertainty estimation is directly achieved from the output of the model and does not need any model modification, nor does it impose any extra computation. As we show in the present work, this uncertainty information can effectively be exploited for better understanding of the underlying collection and training data, and also in downstream tasks such as rank cut-off prediction.

In addition, the decoupling of query/document encoding in the architecture of deep generative model enables the ranking model to store document embeddings at index time and later exploit them at inference time (similar to representation-based models). At the same time, the decoder embeddings still effectively interact with encoded embeddings (typically through attention mechanisms), which is analogous to interaction-focused models. The use of attention mechanisms over the document during decoding facilitates effective interaction with encoded document embeddings (as in query-document interaction models), but also enables the potential incorporation of orthogonal notions, such as personalization, diversity or fairness into the model.

The present work explores various aspects of the generative IR paradigm from the perspective of deep generative models. Concretely, we first formalize the theoretical connection between the introduced deep generative ranking models and classical generative IR models, in particular the query likelihood language model.

We then investigate the effectiveness of various generative architectures in passage retrieval. To this end, we conduct a comprehensive study on the use of the state-of-the-art neural language generation techniques for retrieval. We study various models, among them Pointer Generator Networks (PGN) [43], and a recently proposed combination of BERT and Transformers [24]. In addition to these models, we combine the benefits of PGN in query decoding with those of BERT in document encoding, and propose a new generative ranking model referred to as T-PGN. We evaluate these generative models on the MS MARCO Passage Re-ranking [30] and the TREC Deep Learning 2019 Passage Re-ranking task [6]. The results demonstrate that among the generative models, our introduced T-PGN model shows the overall best performance.

Finally, drawing from the probabilistic framework of deep generative models, we calculate a measure of uncertainty reflecting the model's confidence in the prediction of a given query. The uncertainty estimate is achieved by calculating the entropy values of the probability distributions of query term generation. We use the resulting uncertainty estimates to first analyze the existence of bias with respect to term positions in the queries of MS MARCO and then exploit this extra information for cut-off prediction, observing a significant improvement in the task.
To summarize, our main contribution is four-fold:

- Introducing the novel deep generative ranking models and formalizing them in the perspective of classical generative IR models.
- Adopting several recent deep generative language models for ranking, and introducing a new generative ranker (T-PGN).
- Conducting a large set of evaluation experiments on various generative models for passage retrieval.
- Showcasing the potential of deep generative ranking models for uncertainty estimation of relevance scores and its use in a cut-off prediction task.

The paper is organized as follows: Section 2 reviews related literature. In Section 3, we introduce the deep generative ranking models and explain various architectural choices as well as their potential for uncertainty estimation. Section 4 describes our design of experiments, whose results are reported and discussed in Section 5. The accompanying source code is available at https://github.com/CPJKU/DeepGenIR.

2 RELATED WORK

2.1 Neural Retrieval Models

In the category of query-document interaction models, we can distinguish between three groups of models. The first group captures patterns of similarity values across terms that appear close together within the query and within the document [11, 20, 21, 34]. The second group captures patterns of frequencies across ranges of similarity values [7, 12–14, 50]. The last ones are based on large-scale pre-trained language models, as the use of these models has shown significant performance gains in various IR tasks.

For instance, the BERT model is used for document/passage retrieval through fine-tuning [31], combining them with other ranking models [27], expanding to other more efficient variations [26] or dense retrieval approaches [22, 51]. In this paper, we also investigate the benefits of exploiting such large-scale pre-trained language models in the context of deep generative ranking models.

Finally, in addition to the mentioned neural models, other studies exploit the inherent efficiency of classic IR models while aiming to improve their effectiveness using pre-trained embedding models. This is done for instance by generalizing term salience with translation models [38, 40], and re-weighting terms [57], or through adapting word embeddings for document retrieval by post-filtering [39], retrofitting [16], or re-training on local documents [9].

2.2 Neural Generative Models in IR

Neural generative models have been utilized in various IR tasks. As examples, Zamani et al. [54] study the use of a seq2seq model to generate queries, whose results are used as a source of weak supervision for asking clarifying questions. Ren et al. [41] and later Yang et al. [53] approach the task of reformulating conversational queries into search-engine-friendly queries using seq2seq with attention models. Ahmad and Chang [1], Ahmad et al. [2] use a similar generative model to train a query recommender, which facilitates the reformulation of users' queries and hence the effective ranking of documents.

In the context of neural ranking models, Nogueira et al. [33] use a seq2seq Transformer model [47, 55] to expand documents with generated queries, and adopt a BERT-based matching model to conduct retrieval on the expanded documents. In a more recent work, Nogueira et al. [32] exploit the T5 model [37] (a pre-trained Transformer-based seq2seq model) to perform binary classification of query-document pairs as relevant or irrelevant. In this approach, query and document are both given as the input to the encoder, and the generated output of the decoder is two possible tokens ("true" or "false") corresponding to the relevant and non-relevant class. The authors use the logits corresponding to the two tokens to infer the relevance score. Based on the discussion in Section 1 and on Figure 1, despite the seq2seq architecture of this model, this approach can in fact be categorized among the query-document interaction models, since the input is the concatenation of query and document, and the output is their relevance score. In contrast, the deep generative models presented in the work at hand generate text queries from input documents, where the probability of generating each query term is defined over all possible words (and not over two tokens). Parallel to our work, Zhuang et al. [58] investigate query likelihood models built upon a Transformer-based seq2seq architecture. Our work expands their study by investigating a wide range of deep generative architectures in IR ranking, and showing the fundamental benefits of generative ranking models for query understanding and in downstream tasks.

In a larger context, neural language generation models span various language processing tasks, i.e. machine translation [47], abstractive document summarization [24, 43], dialogue generation [48], and question answering [10, 49]. The present work benefits from and contributes to these studies by adopting deep generative models in the context of retrieval, and introducing a new generative model.

3 DEEP GENERATIVE RANKING MODELS

In the following, we first formulate deep generative ranking models by highlighting their connections to classical generative models [3, 36, 56]. We then describe the various loss functions used to train the models, followed by a detailed description of the proposed T-PGN and other generative ranking models used in this study. Finally, we introduce our approach to uncertainty estimation defined on the probability space resulting from deep generative rankers.

3.1 Definition

Ponte and Croft [36] introduced the language modeling approach to IR and proposed a new scoring model based on this approach, which later has been called the query likelihood model (QL). The language modeling approach defines the relevance of document D to query Q based on the conditional probability $P(Q|D)$.[1] This probability is rooted in the idea that a user who wants to find document D would utilize query Q to retrieve the document [56]. $P(Q|D)$ is defined by $P(Q|\Phi_D)$ – the probability of generating query Q using the language model Φ_D, built based on document D. Zhai [56] explains the objective of Φ_D as "modeling the queries that a user would use in order to retrieve documents", highlighting the fact that, although Φ_D is a document language model, it is effectively a model meant for queries and not for documents.

The most well-known way to use the language modeling approach is by utilizing a multinomial language model assuming query term independence [56], resulting in the following model formulation, known as QL:

$$\text{QL:} \quad P(Q|D) \approx P(Q|\Phi_D) = \prod_{q_i \in Q} P(q_i|\Phi_D) \tag{1}$$

The language model Φ_D is commonly defined as a unigram probability distribution of D over all terms in the vocabulary, smoothed using the collection as background statistics. The relevance score of query to document is defined as the logarithm of the conditional probability.

$$\text{QL:} \quad \text{score}(Q, D) = \sum_{q_i \in Q} \log P(q_i|\Phi_D) \tag{2}$$

Deep generative ranking models follow a similar perspective to relevance estimation as the QL: Document D should be scored as more relevant to query Q if the model assigns a higher value to the probability of generating Q when conditioned on D. This probability

[1]More precisely, $P(Q|D, R = r)$, i.e. the probability of Q given D and a level of relevance r.

is calculated based on an encoder-decoder architecture. The encoder receives document D as a sequence of input tokens and provides a (contextualized) representation of the document. The decoder uses the document's representation as well as previous query tokens and estimates as output the probability of generating the next query token. The decoder is in fact a *query language model* which outputs the probability of generating the next query token, conditioned on the representation of the document in auto-regressive fashion (one after another). Given such a generative model, the relevance score is defined as the probability of generation of the query, conditioned on the document. The generation probability is formulated as follows:

$$P_\theta(Q|D) = \prod_{q_i \in Q} P_\theta(q_i|D, q_{j<i}) \tag{3}$$

where $q_{j<i}$ denotes the query tokens preceding the current token, and θ indicates the model's parameters learned using training data. Similar to QL, the relevance score is defined as the logarithm of the conditional probability:

$$\text{score}(Q, D) = \sum_{q_i \in Q} \log P_\theta(q_i|D, q_{j<i}) \tag{4}$$

Having outlined the conceptual similarities of deep generative ranking models and QL, let us now discuss the differences, particularly by comparing the formulations in Eq. 1 and Eq. 3. One difference is that deep generative models are not constrained by the term independence assumption, as the generation of each token is conditioned on the previous terms. Another difference is rooted in the language models that deep generative models use to generate queries. While QL utilizes Φ_D – the language model of document D – deep generative models use the query language model that is created by observing all queries in the training data. In fact, in contrast to QL, deep generative ranking models explicitly train a language model for queries, whose generation probabilities are conditioned on the given document. In this sense, deep generative ranking models can be seen as an alternative implementation of the language modeling approach to IR, while still benefiting from the advantages of neural ranking models.

3.2 Ranking Loss Functions

Considering the provided formulation of deep generative ranking models, we discuss in this section the training loss functions used to train generative models. Given a pairwise learning-to-rank setting, the training data of retrieval models is provided in the form of a query Q with a relevant and a non-relevant document, denoted as D^+ and D^- respectively.

The first loss is the well-known *negative log likelihood* (NLL), commonly used to train generative models, and defined as follow:

$$\mathcal{L}_{\texttt{NLL}} = -\sum_{(Q,D^+) \in \mathcal{T}} \log P_\theta(Q|D^+) \tag{5}$$

where $\mathcal{T}$ denotes the collection of training data. It is evident that NLL only considers the relevant document and does not use the non-relevant one. The next loss function is *margin ranking* (Marg) formulated below:

$$\mathcal{L}_{\texttt{Marg}} = \sum_{(Q,D^+,D^-) \in \mathcal{T}} \max\{0, b - \log P_\theta(Q|D^+) + \log P_\theta(Q|D^-)\} \tag{6}$$

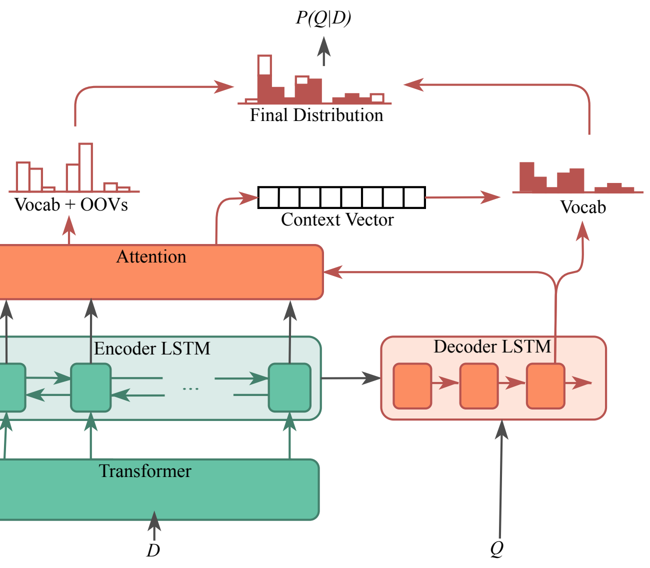


Figure 2: Transformer Pointer Generator Network (T-PGN)

The Marg loss increase the differences between the predicted relevance score of the relevant document and the predicted relevance score of the non-relevant document up to a margin threshold b. This loss can accept relevance scores in any range and is therefore commonly adopted in ranking models.

Our final loss function proposed by dos Santos et al. [10] expands NLL by adding the unlikelihood probability of negative documents, namely the logarithm of one minus probability of the negative document in training data. We refer to this loss as *negative log likelihood log unlikelihood* (NL3U), defined as follows:

$$\mathcal{L}_{\texttt{NL3U}} = -\sum_{(Q,D^+) \in \mathcal{T}} \log P_\theta(Q|D^+) + \log(1 - P_\theta(Q|D^-)) \tag{7}$$

3.3 Neural Generative Ranking Architectures

Based on our formulation of neural generative ranking models, any neural generative model can be exploited for retrieval, namely by calculating the query-to-document relevance estimated from the generation probability distributions. In the following, we first briefly describe the generative models studied in this paper, and then explain our proposed T-PGN model. These models are selected based on their strong performance in tasks such as abstractive document summarization and machine translation.

Seq2SeqAttention. The Sequence-to-Sequence with Attention model [29] is an extension to the baseline Sequence-to-Sequence model [45]. The baseline model consists of an encoder LSTM and a decoder LSTM, where the last hidden state of the encoder is given to the initial hidden state of the decoder. Seq2SeqAttention extends this models by the attention network, defined on the encoder hidden states and conditioned on the hidden state of the decoder LSTM. This attention mechanism enables the immediate access of the decoder to all document embeddings at encoder, facilitating information flow in the model.

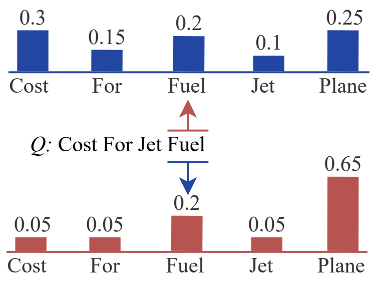


Figure 3: Illustration of different uncertainty estimates on a given query term position. While the probability of generating the term *Fuel* is the same in both cases, the upper distribution contains a much higher degree of uncertainty.

PGN. See et al. [43] introduce the Pointer Generator Network, which expands the Seq2SeqAttention model by a novel copy mechanism. The objective of this copy mechanism is to facilitate the transfer of the out-of-vocabulary (OOV) terms appearing in the document directly to the output query. This approach has shown highly competitive performance in abstractive summarization benchmarks. This is due to the fact that in summarization (similar to IR) rare words – which are commonly removed from the list of vocabularies due to their low collection frequencies – can be highly salient, and hence crucial for the success of the task.

Transf2Transf. The Transformer-to-Transformer is introduced by Vaswani et al. [47] in the context of machine translation. The model consists of multiple layers of encoder Transformers to contextualize document embeddings with self-attention, followed by multiple layers of decoder Transformers. The decoder Transformers generate output probability distributions by contextualizing query embeddings and attending to the final embeddings of the encoder.

BERT2Transf. The BERT-to-Transformer model, recently introduced by Liu and Lapata [24], achieves state-of-the-art results on abstractive text summarization collections. The model has a similar architecture to the one of Transf2Transf but instead of Transformers uses a BERT model encoder.

Transformer Pointer Generator Networks (T-PGN). We introduce the Transformer Pointer Generator Networks model (T-PGN) which combines the advantages of the PGN model with Transformers. The architecture of the model is shown in Figure 2. The T-PGN model provides a multi-layer encoder Transformer to create contextualized word embeddings of document terms. These embeddings are then passed to the encoder LSTM, whose final hidden state is used as the initial state of the decoder LSTM. Similar to PGN, the attention distribution over the contextualized document embeddings (containing the OOV terms) is combined with the output distribution provided by the decoder to form the final distribution. This provides a probability distribution of query generation defined over all words in the vocabulary as well as the OOV terms appearing in document.

3.4 Uncertainty Estimation in Neural Generative Rankers

Given a document, deep generative models predict a probability distribution for each term of a query. As discussed in Section 1, this probabilistic perspective enables the calculation of the uncertainty of the model with respect to this prediction. In the following, we explain our approach to calculating an uncertainty estimate given any deep generative model.

At every step i of query term generation, the deep generative models estimate a probability distribution over all terms of the vocabulary. Despite the selected probability value for the term in the position i, $P_\theta(q_i|D, q_{j<i})$, the form of the predicted probability distribution reveals parallel information about the model. In fact, the same generation probability of a term may result from different kinds of probability distributions. This point is illustrated with a toy example in Figure 3 for the term *Fuel*. As shown, if the distribution of the term generation probabilities is close to uniform (the upper graphic in Figure 3), the model is not certain about the generation probability, as many terms have comparable chance to be generated in the next position. In contrast, when the distribution is more skewed, the model is more certain about possible generation terms (the lower graphic in Figure 3). Despite these different distributions, the predicted probability values of *Fuel* in both distributions are equal. In fact, this *term-level uncertainty* provides extra information that might not be captured in the predicted probability values, and hence the predicted relevance score.

Similar to Xu et al. [52], we define term-level uncertainty as the entropy of the *nucleus* probability distribution at each step. The nucleus distribution [17] provides a well-behaved version of the original generation probability distribution, by redistributing the very low probability values. More concretely, the nucleus distribution recomputes the probability distribution only on the k most probable terms, where k is chosen such that the accumulated probabilities of these k terms is equal to or greater than a predefined threshold p. Similar to Xu et al. [52], we set $p = 0.95$.

Given the nucleus probability distribution for the generation of the term at time step i, denoted as X^i, the term-level uncertainty of the model is calculated as follows:

$$\text{term-level uncertainty}(X^i) = -\sum_{x \in X^i} P(x) \cdot \log P(x) \qquad (8)$$

Using this definition, we can estimate a model's uncertainty with respect to generating the whole query, namely *query-level uncertainty*, by aggregating term-level uncertainty values. To this end, various aggregation functions (such as mean, entropy, variance, and maximum) can be applied to the corresponding values of each query. We further investigate the characteristic of this uncertainty estimation for model/collection analysis and the cutoff prediction task in Section 5.3 and Section 5.4, respectively.

4 EXPERIMENT SETUP

Collections. We conduct our evaluation experiments on two paragraph retrieval collections. The first is the MS MARCO Passage Reranking collection [30]. In total, the development set of MS MARCO comprises 8,841,822 documents, and 55,578 queries. We follow the setting in Hofstätter et al. [13] and split the queries into a validation and a test set, containing 6,980 and 48,598 queries, respectively. Since the provided relevance judgements by this collection are highly sparse, we refer to this test set as SPARSE. The second test collection is the TREC Deep Learning Track 2019 Passage Retrieval set (TREC-19) [6], which also originates from the MS MARCO collection. The TREC-19 collection encompasses 43 annotated queries.

Generative deep ranking models. Within the proposed generative neural re-ranking framework, we investigate Seq2SeqAttention, Transf2Transf, BERT2Transf, PGN, and T-PGN.

Matching models. For the sake of a well rounded performance evaluation, we sample a number of IR models to compare generative models to: Kernel-based Neural Ranking Model (KNRM) [50] Convolutional KNRM (ConvKNRM) [7], MatchPyramid [34], and two most recent Transformer-based models: Transformer-Kernel (TK) [14] and the fine-tuned BERT model [8]. This list is indeed non-comprehensive, since our central aim is to investigate model architectures within the neural generative paradigm. We conduct experiments on BM25 as a classical matching model.

Model configuration and training. To provide a fair comparison, we aim to select similar configurations for the different models. Every model using pre-trained word embeddings (Seq2SeqAttention, PGN, T-PGN, Transf2Transf, and TK) operates with the same set of pre-trained GloVe [35] vectors of length 300. For models with BERT (BERT, BERT2Transf), we investigate a recently-released version of the pre-trained language model known as BERT-Tiny [46], which has two layers of Transformers, two attention heads on each layer, an intermediate feed-forward layer of size 512, and a final (sub)word embedding of size 128. While much smaller, this model has shown competitive performance in various language processing tasks [46] in comparison with the larger versions of BERT, making it suitable for conducting large-scale experiments with various hyper-parameter settings. We set the setting of all other model with Transformer networks (TK, T-PGN, and Transf2Transf) to the same one as BERT-Tiny. Models that contain BERT (BERT, BERT2Transf) utilize WordPiece tokenization. All state-of-the-art models are trained with their recommended loss functions, namely Cross Entropy (`CE`) for BERT and `Marg` for the other matching models. The proposed generative models are trained using three different loss functions: `NLL`, `Marg` and `NL3U`. Seq2SeqAttention and PGN have a learning rate of 0.001. Non-BERT Transformer-based models start from learning rates of 0.0001. BERT-based generative models use a learning rate of 0.0001 for training the Transformer-decoder, and 0.00003 for the pre-trained BERT encoder. The complete hyperparameter settings of all models are provided in the published repository together with the source code.[2]

Evaluation. We evaluate the performance of all models based on the re-ranking approach. To this end, we first compute the top 200 passages as retrieved by a BM25 model. The resulting candidate documents are then re-ranked by each of the investigated neural model. The final re-ranked results are evaluated using several common performance metrics, namely mean reciprocal rank (MRR), normalized discounted cumulative gain at 10 (NDCG), and recall. To investigate statistical significance of results, we conduct two-sided paired t-tests (details given below). In addition, we qualitatively analyze a selection of generated queries.

5 RESULTS AND ANALYSIS

In this section, we first show the performance evaluation results of the various deep generative models, followed by qualitative analysis of the query generation process. We then explore the use of uncertainty estimates to analyze the underlying characteristics of the model and data, followed by showing the benefit of including uncertainty information in the cut-off prediction task.

5.1 Performance Evaluation

Evaluation results are provided in Table 1 for all assessed models. Matching models are grouped at the top of the table, and the lower part is dedicated to generative models. For each neural generative model, the results on three loss functions (`NLL`, `Marg`, `NL3U`) are reported. The best performance among all generative models is marked in bold. To denote statistical significance, we first assign each generative model a letter a to f (see first column of Table 1). Each performance result of each model is also marked with superscript letters, indicating to which other models a statistically significant difference exists. To give an example: model T-PGN trained with loss `NLL`, obtaining a MRR of 0.278^{abcde} on the SPARSE test set, is significantly better (in terms of MRR) than generative models a, b, c, d and e which have also been trained with the same loss `NLL`.

Let us have a closer look at the results of generative models. The results indicate that the models that use the copy mechanism show the best overall performance among the generative models. In particular, T-PGN shows significantly better results than all other deep generative models on SPARSE, while PGN shows better performance on TREC-19. The better performance of PGN-based models (PGN and T-PGN) in contrast to BERT-based ones is specific to the retrieval task, and in fact stands in contrast to the common architectural preferences in other tasks such as machine translation and abstractive document summarization.

The effectiveness of the PGN-based models can be traced in their decoder architectures, particularly by comparing between PGN and Seq2SeqAttention. While the sole difference of these two models lies in the use of the copy mechanism, the PGN and T-PGN models show significantly higher results with large margins. We assume that this is due to the way that the copy mechanism in PGN-based models approach out-of-vocabulary terms (OOVs). In fact, as observed in previous studies [13], OOVs correspond to infrequent words that – due to their rarity – contain crucial information for retrieval. Models that leverage this information, therefore, reach higher performance levels. While PGN and T-PGN both benefit from effective decoding (in respect to these retrieval tasks), the improvement of T-PGN on SPARSE highlights the importance of enriching the encoding layer with Transformers which differentiates the T-PGN model from PGN.

Inspecting results for the different loss functions used for the deep generative models reveals that, overall, the differences between various loss functions are negligible, such that the models using `NLL` (as the simplest loss function) perform generally similar to the ones with `Marg` or `NL3U`. We speculate that this is due to the probabilistic nature of generative models, as the objective of such models is to estimate generation probability distributions, which (based on the results) can be achieved by solely increasing the generation probability of relevant documents. We therefore conclude that a generative model can effectively be trained with the `NLL` loss function as the simplest choice, which has the benefit of faster training time in comparison with other loss functions.[3]

[2]https://github.com/CPJKU/DeepGenIR.

[3]Since `NLL` in contrast to `Marg` and `NL3U` only processes the relevant documents.

Table 1: Results of investigated models in terms of MRR, NDCG, and Recall. Best performances among generative models are marked in bold. Superscripts show significant improvement over respective models trained with the same loss.

Model	Loss	SPARSE			TREC-19		
		MRR	NDCG	Recall	MRR	NDCG	Recall
BM25		0.199	0.231	0.383	0.825	0.506	0.129
MatchPyramid	Marg	0.242	0.280	0.450	0.884	0.577	0.135
KNRM	Marg	0.234	0.274	0.448	0.861	0.545	0.138
ConvKNRM	Marg	0.275	0.318	0.498	0.901	0.605	0.152
TK	Marg	0.308	0.355	0.545	0.943	0.661	0.159
BERT	CE	0.305	0.353	0.542	0.899	0.651	0.152
QL (*a*)		0.181	0.211	0.355	0.773	0.470	0.124
Seq2SeqAttention (*b*)	NLL	0.246^{a}	0.285^{a}	0.455^{a}	0.825	0.557^{a}	0.141
	Marg	0.210^{a}	0.243^{a}	0.399^{a}	0.860	0.530	0.123
	NL3U	0.243^{a}	0.282^{a}	0.453^{a}	0.859	0.558^{a}	0.140
Transf2Transf (*c*)	NLL	0.255^{ab}	0.297^{ab}	0.478^{ab}	0.846	0.541	0.148
	Marg	0.258^{ab}	0.299^{ab}	0.474^{abd}	0.893	0.590^{ab}	0.138
	NL3U	0.252^{ab}	0.295^{ab}	0.475^{ab}	0.883	0.544	0.142
BERT2Transf (*d*)	NLL	0.257^{abc}	0.300^{abc}	0.480^{ab}	0.831	0.554^{a}	0.149^{b}
	Marg	0.257^{ab}	0.297^{ab}	0.469^{ab}	0.863	0.573^{a}	0.136
	NL3U	0.258^{abc}	0.300^{abc}	0.478^{ab}	0.873	0.571^{a}	**0.150**^{abc}
PGN (*e*)	NLL	0.273^{abcd}	0.317^{abcd}	0.498^{abcd}	0.907^{a}	0.585^{acd}	0.150^{b}
	Marg	0.275^{abcd}	0.317^{abcd}	0.493^{abcdf}	**0.912**^{a}	**0.609**^{ab}	0.145^{b}
	NL3U	0.272^{abcd}	0.316^{abcd}	0.498^{abcd}	0.845	0.569^{a}	0.149^{b}
T-PGN (*f*)	NLL	0.278^{abcde}	0.323^{abcde}	0.506^{abcde}	0.885	0.575^{a}	0.144
	Marg	0.276^{abcd}	0.317^{abcd}	0.488^{abcd}	0.880	0.601^{ab}	0.148^{ab}
	NL3U	**0.281**^{abcde}	**0.325**^{abcde}	**0.508**^{abcde}	0.891^{a}	0.573^{a}	0.145

Finally, comparing the results of deep generative models with the state-of-the-art query-document interaction models with Transformers and BERT, we observe that overall the generative models show only marginally lower performance.[4]

These observations on the significant differences between various architectural choices are particularly important considering that, as discussed in Section 2, most current studies which exploit generative models (e.g., for tasks such as query reformulation) use similar models to Seq2SeqAttention [41, 53, 54] or the ones that utilize Transformers as decoder [33]. Based on our results, exploiting OOV-aware models such as T-PGN can provide considerable benefits for the corresponding final tasks.

5.2 Qualitative analysis of generated queries.

We now look at the query generation aspect of the models from a qualitative perspective. In the current and next section, we use T-PGN as our overall best-performing deep generative model to generate queries in a greedy generation process. In this process, for every position the token with the highest probability is selected from the generation probability distribution of words. The generated query in this way is a greedy approximation of the query with the highest generation probability for the given passage.

[4]Comparing the latency of models, it is expected that the neural generative models have overall longer inference time due to their generation process. In particular, we observe that the PGN-based models, due to the use of two LSTMs at encoder and decoder, have considerably longer inference time. However, BERT2Transf, while performing marginally lower than the PGN-based models, shows almost on-par latency to the BERT ranker.

Table 2: Examples of passages, actual queries for which the passage was marked relevant, and synthetic queries most likely to be generated by T-PGN.

Example 1
Passage: Fleas are holometabolous insects, going through the four lifecycle stages of egg, larva, pupa, and imago (adult). Adult fleas must feed on blood before they can become capable of reproduction. Flea populations are distributed with about 50% eggs, 35% larvae, 10% pupae, and 5% adults. Generally speaking, an adult flea only lives for 2 or 3 months. Without a host for food a flea's life might be as short as a few days. With ample food supply, the adult flea will often live up to 100 days. **Actual query**: how long is life cycle of flea **Generated query**: how long do fleas live
Example 2
Passage: I have always wanted to become a Nurse and I have been doing some research and came across the different Nursing 'titles' such as RN (Registered Nurse), BSN(Bachlor's in Science of Nursing) NA(Nurse Assistant), CRNA (Certified Registered Nurse Anesthetist), LPN and LVN.SN = Bachelor of Science in Nursing, which is just a 4 year RN degree. Both the 2 year and the BSN graduates sit for the exact same licensure exam and earn the same RN license. **Actual query**: difference between rn and bsn **Generated query**: what degree do you need to be a nurse
Example 3
Passage: The flea population is typically. made up of 50% eggs, 30% larvae, 15% pupae and only 5% biting adults. Completion of the life cycle from egg to adult varies from two weeks to eight months. Normally the female flea lays about 15 to 20 eggs per day up to 600 in a lifetime. Usual hosts for fleas are dogs, cats, rats, rabbits, mice, squirrels, chipmunks, raccoon's, opossums, foxes, chickens, and humans. **Actual query**: how long is life cycle of flea **Generated query**: how long do chickens live

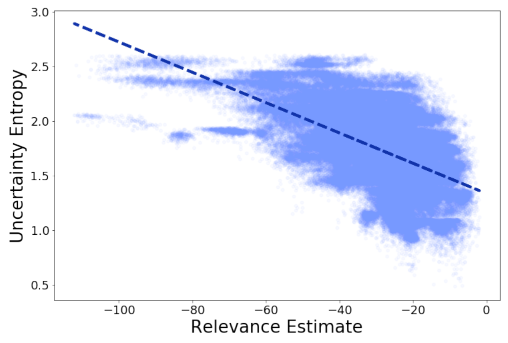


Figure 4: Relevance versus query-level uncertainty of the T-PGN model on TREC-19.

Table 2 shows examples of passages, provided queries in the dataset assessed as relevant to the corresponding passages, and the queries generated by our model. We expect that a generated query is conceptually relevant to the given passage. Looking at Example 1, we observe that the generated query is almost the same as the actual relevant one. It means that the model will predict a high relevance score of the query to the passage. Example 2 shows the opposite situation: the generated query, while being completely different from the actual one, is still a valid and relevant query to the given passage. Finally, in Example 3, the same actual query as the one in Example 1 is used but with a different (while still relevant) passage. This example highlights a failure case where the generated query (according to the discussed greedy approach) is conceptually non-relevant to the passage. These examples motivate the deeper understanding of neural generative ranking models, while the shown cases are directly relevant to the tasks that exploit query generation for downstream tasks [33].

5.3 Model Understanding Through the Lens of Uncertainty

In the following, we present model- and data-related insights we obtained from analyzing uncertainty estimates for the T-PGN model (Section 3.4) to approach the following questions: (1) Is there any connection between the model's confidence in its query generation probability and query-document relevance estimates? (2) Are there any patterns in the uncertainty distribution along query term positions and, if yes, what do they indicate?

To address the first question, we start with calculating query-level uncertainty estimates by aggregating over term-level uncertainties using mean, entropy, variance, and maximum. Then, for each query-document pair in the top-200 of a ranking list, we calculate the Spearman-*r* correlation between each query-level uncertainty and the predicted relevance scores. We calculate these correlations for TREC-19, containing 8,600 query-document pairs.

The calculated correlations for mean, variance, max, and entropy are -0.223, -0.206, -0.358, and -0.569, respectively. All different uncertainty aggregation results show a negative correlation to relevance score, indicating that a decreasing predicted relevance score (for the documents in the lower positions in the ranking list) increases uncertainty of the model. Figure 4 demonstrates the relevance and query-level uncertainty estimates using entropy for aggregation, because of its highest negative correlation. The plot

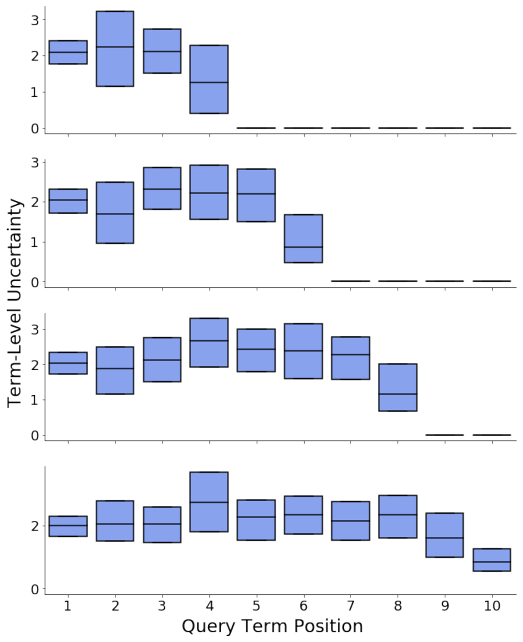


Figure 5: The interquartile ranges of term-level uncertainty scores, calculated on each term position for all queries with a given length, namely the length of 4, 6, 8, and 10. For each query length, the last term position corresponds to the <EOS> special token denoting the end of the query.

shows that uncertainty of query-document pairs with higher relevance is widely spread, but the distribution tends to get focused on a high-uncertainty area as the relevance decreases.

Considering these results, with regard to our first question, we conclude that the model tends to exhibit higher levels of uncertainty (in the likelihood of generating the query in query-document pairs) for low relevance estimates. This could indicate that uncertainty may contain additional information to relevance which can be exploited in retrieval tasks. We return to this point in Section 5.4.

To approach the second question, using the query-document pairs of the TREC-19 results, we average the term-level uncertainty values over all query terms that appear in a specific position of the queries. To make the results comparable across various query lengths, we apply this position-level aggregation over the queries with the same size. Figure 5 shows these term-level uncertainty distributions for every position in queries of length 4, 6, 8, and 10 terms, where each query ends with the <EOS> special token. Every box in the plot represents the interquartile range of term-level uncertainty distribution for each position.

Looking at Figure 5, we observe two major patterns in all four settings regarding query length: (1) All average uncertainties tend to become lower (more confident) in the last terms of the query, where the last term has consistently the lowest uncertainty with a considerable drop in comparison to the uncertainty at previous term positions. (2) The uncertainty distributions regarding the first position have similar median values across the four settings with small variances when compared to the distributions for other positions.

Our first observation is similar to the findings by Xu et al. [52] in the context of abstractive text summarization. This indicates that by observing more terms during the query generation, the model becomes more and more certain about the distribution of possible next terms, and this confidence has its maximum in the last term.

Our second observation is, however, in contrast to the results reported by Xu et al. [52]. In their experiments, generative models show the greatest interquartile range of term-level uncertainty for earlier words in the generated sequence. This can potentially reveal the existence of a bias in the queries of the MS MARCO training dataset, considering that many queries in the dataset start with question words such as *what*, *how*, and *where*. In fact, the persistent uncertainty distributions for the first position can indicate the limited number of unique terms in training data, with which a query begins. This observation is inline with and reinforces the conclusions in Hofstätter et al. [15]. However, while they show the existence of bias in the MS MARCO collection through extensive fine-grained annotation, we view this from the lens of the uncertainty of the model on each query term position.

5.4 Cut-off Prediction with Uncertainty

Do the uncertainty estimates provide novel and complementary information to what is provided by relevance scores? If yes, can this information be exploited in downstream IR tasks? To answer these questions we evaluate the expressiveness of the uncertainty estimates in a similar fashion to Cohen et al. [5], via the cutoff prediction task. The objective of the cut-off prediction task is to dynamically determine a cut-off point for a ranked list of documents in order to maximize some non-monotonic metric, in our case F_1 scores. As discussed by Lien et al. [23], the task is motivated by neural models losing confidence in their estimations as documents become less relevant to the query. In a real-world scenario, cut-off prediction can be used by a retrieval system to prevent users from scraping over search results, about which the ranker is not sufficiently confident. In such scenarios, the search engine can switch to alternative strategies, such as applying different ranking model or encouraging the user to reformulate the query.

To study the effect of uncertainty on this task, we follow the same procedure as in Bahri et al. [4], namely by using the proposed Transformer-based cut-off predictor, and comparing the performance in terms of F_1 score (see Bahri et al. [4] for more details). The predictor receives a set of features in the sense of query-document interactions, and for each query provides a prediction regarding the best cut-off in its ranked list. A common feature for this task is relevance scores, assuming that the changes in relevance can be indicative of an optimal cut-off point [4]. In our experiments, we are interested in examining whether adding uncertainty information can further improve this tasks by providing new information.

We therefore conduct our experiments in two configurations: (1) using only relevance estimation from T-PGN as single feature, referred to as `Rel`; (2) adding the four query-level uncertainty estimates (through mean, entropy, variance, and maximum term-level uncertainty aggregations) as additional features, referred to as `Rel+Uncertainty`. To train the cut-off predictors we use the queries of TREC-19. While this task can benefit from the large number of the queries in SPARSE, the task intrinsically requires a sufficient amount of relevance judgements which are not available in the SPARSE collection. In addition to `Rel` and `Rel+Uncertainty`, we calculate the results of a `Greedy` approach which provides a naive baseline by selecting the same cut-off for all ranked lists, chosen by maximizing the F_1 score on the training set. Finally, the `Oracle` model indicates the score that an ideal cut-off selection would achieve. We report in Table 3, for each configuration, the resulting F_1 score as well as its percentage when compared to the F_1 score of `Oracle`. For each of the configuration, the experiment is conducted in 50 trials, where in each trial 5-fold cross validation is applied. The final results are averaged over all trials.

Table 3: Results on cut-off prediction task with features produced by T-PGN on TREC-19 test collection. The last column show the percentage of F_1 in respect to the results of `Oracle`. The † sign shows the statistical improvement of `Rel+Uncertainty` over `Rel` with $p < 0.001$.

	F_1	% to `Oracle`
`Greedy`	0.193	39.1
`Oracle`	0.493	100.0
`Rel`	0.345	70.0
`Rel+Uncertainty`	0.364†	73.8

Comparing results for `Rel` and `Greedy` in Table 3 – as reported in previous studies Bahri et al. [4], Lien et al. [23] – we observe that relevance information is an important signal for this task. Comparing the results of `Rel` with `Rel+Uncertainty` we observe additional improvements by incorporating the uncertainty information. Calculating a two-sided t-test with $p < 0.001$ between the results of `Rel+Uncertainty` and `Rel` confirms the significance of this improvement. These results substantiate the value of the uncertainty scores, inherent in the architecture of deep generative IR models, which provide additional actionable information for IR tasks.

6 CONCLUSION

We propose a modern perspective on the generative IR paradigm by introducing novel deep generative ranking models. The introduced models offer a solid granular probabilistic framework of neural retrieval, which lays the foundation for estimation of additional model-level information such as uncertainty. Proposing a novel deep generative ranking model, T-PGN, we investigate the performance of several deep generative IR models on two passage retrieval collections. Our evaluation results show the importance of the copy mechanism in the generative models in the context of retrieval, as provided by the PGN and T-PGN models. We further explore the information provided by the uncertainty estimates, and showcase the value of such uncertainty information in a cut-off prediction task.

ACKNOWLEDGEMENTS

This research is supported in part by the NSF (IIS-1956221). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of NSF or the U.S. Government. This research is also supported by Know-Center Graz, through project “Theory-inspired Recommender Systems”.

REFERENCES

[1] Wasi Uddin Ahmad and Kai-Wei Chang. 2018. Multi-Task Learning For Document Ranking And Query Suggestion. In *Sixth International Conference on Learning Representations.*

[2] Wasi Uddin Ahmad, Kai-Wei Chang, and Hongning Wang. 2019. Context Attentive Document Ranking and Query Suggestion. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval.* 385–394.

[3] Qingyao Ai. 2019. Neural Generative Models and Representation Learning for Information Retrieval.

[4] Dara Bahri, Yi Tay, Che Zheng, Donald Metzler, and Andrew Tomkins. 2020. Choppy: Cut Transformer for Ranked List Truncation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20).* 1513–1516.

[5] Daniel Cohen, Bhaskar Mitra, Oleg Lesota, Navid Rekabsaz, and Carsten Eickhoff. 2021. Not All Relevance Scores are Equal: Efficient Uncertainty and Calibration Modeling for Deep Retrieval Models. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval.*

[6] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M Voorhees. 2020. Overview of the trec 2019 deep learning track. *arXiv preprint arXiv:2003.07820* (2020).

[7] Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. 2018. Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *Proc. of the ACM Conference on Web Search and Data Mining.*

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proc. of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.*

[9] Fernando Diaz, Bhaskar Mitra, and Nick Craswell. 2016. Query Expansion with Locally-Trained Word Embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 367–377.

[10] Cicero dos Santos, Xiaofei Ma, Ramesh Nallapati, Zhiheng Huang, and Bing Xiang. 2020. Beyond [CLS] through Ranking by Generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP).* 1722–1727.

[11] Yixing Fan, Jiafeng Guo, Yanyan Lan, Jun Xu, Chengxiang Zhai, and Xueqi Cheng. 2018. Modeling diverse relevance patterns in ad-hoc retrieval. In *The 41st ACM SIGIR Conference on Research and Development in Information Retrieval.*

[12] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proc. of the ACM on Conference on Information and Knowledge Management.*

[13] Sebastian Hofstätter, Navid Rekabsaz, Carsten Eickhoff, and Allan Hanbury. 2019. On the Effect of Low-Frequency Terms on Neural-IR Models. In *Proc. of the ACM SIGIR Conference on Research and Development in Information Retrieval.*

[14] Sebastian Hofstätter, Hamed Zamani, Bhaskar Mitra, Nick Craswell, and Allan Hanbury. 2020. Local Self-Attention over Long Text for Efficient Document Retrieval. In *Proc. of the ACM SIGIR conference on Research and development in information retrieval.*

[15] Sebastian Hofstätter, Markus Zlabinger, Mete Sertkan, Michael Schröder, and Allan Hanbury. 2020. Fine-Grained Relevance Annotations for Multi-Task Document Ranking and Question Answering. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management.* 3031–3038.

[16] Sebastian Hofstätter, Navid Rekabsaz, Mihai Lupu, Carsten Eickhoff, and Allan Hanbury. 2019. Enriching Word Embeddings for Patent Retrieval with Global Context. In *Proc. of the European Conference of Information Retrieval.*

[17] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The Curious Case of Neural Text Degeneration. arXiv:1904.09751 [cs.CL]

[18] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Proc. of the Conference on Advances in Neural Information Processing Systems.*

[19] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proc. of the ACM Conference on Information and Knowledge Management.*

[20] Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2017. PACRR: A Position-Aware Neural IR Model for Relevance Matching. In *Proc. of the Conference on Empirical Methods in Natural Language Processing.*

[21] Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2018. Co-PACRR: A context-aware neural IR model for ad-hoc retrieval. In *Proc. of the ACM Conference on Web Search and Data Mining.*

[22] Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval.* 39–48.

[23] Yen-Chieh Lien, Daniel Cohen, and W. Bruce Croft. 2019. An Assumption-Free Approach to the Dynamic Truncation of Ranked Lists. In *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval (ICTIR).* ACM, 79–82.

[24] Yang Liu and Mirella Lapata. 2019. Text Summarization with Pretrained Encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP).* 3721–3731.

[25] Zhengdong Lu and Hang Li. 2013. A deep architecture for matching short texts. In *Proc. of the Conference on Advances in Neural Information Processing Systems.*

[26] Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. 2020. Efficient document re-ranking for transformers by precomputing term representations. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval.* 49–58.

[27] Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. 2019. Contextualized Word Representations for Document Re-Ranking. In *Proc. of the ACM SIGIR conference on Research and Development in Information Retrieval.*

[28] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to match using local and distributed representations of text for web search. In *Proc. of the Conference on World Wide Web.*

[29] Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, and Bing Xiang. 2016. Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning.* 280–290.

[30] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268* (2016).

[31] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *arXiv preprint arXiv:1901.04085* (2019).

[32] Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document Ranking with a Pretrained Sequence-to-Sequence Model. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings.* 708–718.

[33] Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document Expansion by Query Prediction. *arXiv preprint arXiv:1904.08375* (2019).

[34] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text Matching as Image Recognition.. In *Proc. of the AAAI Conference on Artificial Intelligence.*

[35] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proc. of the 2014 conference on empirical methods in natural language processing (EMNLP).* 1532–1543.

[36] Jay M Ponte and W Bruce Croft. 1998. A language modeling approach to information retrieval. In *Proc. of the 21st annual ACM SIGIR conference on Research and development in information retrieval.* ACM, 275–281.

[37] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research* 21 (2020), 1–67.

[38] Navid Rekabsaz, Mihai Lupu, and Allan Hanbury. 2017. Exploration of a threshold for similarity based on uncertainty in word embedding. In *Proc. of the European Conference on Information Retrieval.*

[39] Navid Rekabsaz, Mihai Lupu, Allan Hanbury, and Hamed Zamani. 2017. Word Embedding Causes Topic Shifting; Exploit Global Context!. In *Proc. of the ACM SIGIR Conference on Research and Development in Information Retrieval.*

[40] Navid Rekabsaz, Mihai Lupu, Allan Hanbury, and Guido Zuccon. 2016. Generalizing translation models in the probabilistic relevance framework. In *Proc. of the ACM on Conference on Information and Knowledge Management.*

[41] Gary Ren, Xiaochuan Ni, Manish Malik, and Qifa Ke. 2018. Conversational query understanding using sequence to sequence modeling. In *Proceedings of the 2018 World Wide Web Conference.* 1715–1724.

[42] Gerard Salton, Anita Wong, and Chung-Shu Yang. 1975. A vector space model for automatic indexing. *Commun. ACM* 18, 11 (1975), 613–620.

[43] Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get To The Point: Summarization with Pointer-Generator Networks. In *Proc. of the Annual Meeting of the Association for Computational Linguistics.*

[44] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *Proc. of the Conference on World Wide Web.*

[45] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proc. of the Conference on Advances in Neural Information Processing Systems.*

[46] Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-Read Students Learn Better: On the Importance of Pre-training Compact Models. arXiv:1908.08962 [cs.CL]

[47] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. of the Conference on Advances in Neural Information Processing Systems.*

[48] Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869* (2015).

[49] Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proc. of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

[50] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-end neural ad-hoc ranking with kernel pooling. In *Proc. of the ACM SIGIR Conference on Research and Development in Information Retrieval*.

[51] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwikj. 2021. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. In *Proceedings of the International Conference on Learning Representations*.

[52] Jiacheng Xu, Shrey Desai, and Greg Durrett. 2020. Understanding Neural Abstractive Summarization Models via Uncertainty. arXiv:2010.07882 [cs.CL]

[53] Liu Yang, Junjie Hu, Minghui Qiu, Chen Qu, Jianfeng Gao, W Bruce Croft, Xiaodong Liu, Yelong Shen, and Jingjing Liu. 2019. A hybrid retrieval-generation neural conversation model. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1341–1350.

[54] Hamed Zamani, Susan Dumais, Nick Craswell, Paul Bennett, and Gord Lueck. 2020. Generating clarifying questions for information retrieval. In *Proceedings of The Web Conference 2020*. 418–428.

[55] George Zerveas, Ruochen Zhang, Leila Kim, and Carsten Eickhoff. 2019. Brown University at TREC Deep Learning 2019. In *Proceedings of the 28th Text Retrieval Conference (TREC)*. NIST.

[56] ChengXiang Zhai. 2008. Statistical language models for information retrieval. *Synthesis lectures on human language technologies* (2008).

[57] Guoqing Zheng and Jamie Callan. 2015. Learning to reweight terms with distributed representations. In *Proc. of the ACM SIGIR Conference on Research and Development in Information Retrieval*.

[58] Shengyao Zhuang, Hang Li, and Guido Zuccon. 2021. Deep Query Likelihood Model for Information Retrieval. In *Proceedings of the 43rd European Conference on IR Research, ECIR 2021, Virtual Event*. Springer, 463–470.