MS MARCO Chameleons: Challenging the MS MARCO Leaderboard with Extremely Obstinate Queries

Negar Arabzadeh
University of Waterloo
narabzad@uwaterloo.ca

Bhaskar Mitra
Microsoft
bmitra@microsoft.com

Ebrahim Bagheri
Ryerson University
bagheri@ryerson.ca

ABSTRACT

During the recent years and with the growing influence of neural architectures, tasks such as ad hoc retrieval have witnessed an impressive improvement in performance. For instance, the performance of rankers on the passage retrieval task on the MS MARCO dataset has improved by an order of magnitude in less than two years. In this paper, we go beyond the overall performance of the state of the art rankers and empirically study their performance from a finer-grained perspective. We find that while neural rankers have been able to consistently improve performance, this has been in part thanks to a specific set of queries from within the larger query set. We systematically show that there are subsets of queries that are *difficult* for each and every one of the neural rankers, which we refer to as *obstinate* queries. We show the obstinate queries are similar to easier queries in terms of their number of available relevant judgement documents and the length of the query itself but they are extremely more difficult to satisfy by existing rankers. Furthermore, we observe that query reformulation methods cannot help these queries. On this basis, we present three datasets derived from the MS MARCO Dev set, called the MS MARCO Chameleon datasets. We believe that the next breakthrough in performance would need to necessarily consider the queries in the MS MARCO Chameleons, as such, propose that a well-rounded evaluation strategy for any new ranker would need to include performance measures on both the overall MS MARCO dataset as well as the proposed MS MARCO Chameleon datasets.

1 INTRODUCTION

The recent advances in neural information processing has made a noticeable impact on many information retrieval tasks including question answering, [14, 20, 25], ad hoc retrieval [10, 15, 16, 26, 31], and knowledge graph search [11], just to name a few. Particularly, the ad hoc retrieval task has witnessed a number of recent neural (re)rankers that have shown impressive performance improvements over traditional retrieval methods [13]. These developments have been made possible, in part, thanks to the large-scale datasets such as MS MARCO [32] that provide a large number of queries and their associated relevance judgements, which can be used for training neural rankers. When reviewing the leaderboard associated with the MS MARCO passage retrieval dataset, the performance improvements gained over the past two years is impressive. For instance, the best run submitted to the MS MARCO leaderboard in 2018 produced an MRR@10 of 0.271 on the development set, while the best run submitted in 2020 reported 0.426 on the same metric and dataset. This means that the effectiveness of the ranking methods has improved by an order of magnitude over a two-year period.

Table 1: Comparing the median and mean of average precision and reciprocal rank over 6,980 queries in the MS MARCO Dev set.

		Average Precision		Reciprocal Rank	
Method	Citation	Mean	Median	Mean	Median
BM25	[18]	0.1956	0.0455	0.1874	0
DeepCT	[5]	0.2500	0.0833	0.2421	0
DocT5Query	[33]	0.2850	0.125	0.2768	0.1250
RepBERT	[48]	0.3041	0.125	0.2967	0.1250
ANCE	[42]	0.3365	0.1667	0.3304	0.1667
TCT-ColBERT	[27]	0.3415	0.1667	0.3349	0.1667

While the MS MARCO leaderboard and the authors of many papers resort to reporting their effectiveness based on the whole collection of queries, the focus of this paper is to dig deeper into the performance of recent state of the art neural rankers at the *query level* and explore whether the improvements obtained by the neural rankers are consistent across the whole dataset or not. There are many ranking stacks that provide the state-of-the-art performance by multi-stage ranking [8, 34], however, in this work, we only focus on single stage retrievals. Improving first stage of the ranking stack would consequently lead to performance boost. Based on an empirical study over the runs of five leading neural-based first stage retrieval methods, we find there are a consistent set of poor-performing queries that cannot be addressed by any of the existing neural rankers. We additionally observe that the performance improvements observed by neural rankers are due to gradual improvements obtained over a certain subset of the dataset and as such, performance improvements reported in the literature are not necessarily due to the consistent performance improvement over all of the queries.

In order to substantiate our discussion, let us consider several state-of-the-art methods that have shown strong performance on the 6,980 queries in the MS MARCO Dev set. These methods along with mean and median of their average precision and reciprocal rank are reported in Table 1. The contrast between mean and median is quite meaningful and shows that a significant number of the queries report an average precision less than the overall reported average. We will show later in the paper that even for the queries that show

CIKM '21, November 1–5, 2021, Virtual Event, QLD, Australia

ACM ISBN 978-1-4503-8446-9/21/11. . . $15.00
https://doi.org/10.1145/3459637.3482011

performance improvements, the overall distribution is long-tail where a limited number of queries have been improved to a great extent while the others do not enjoy much or any improvements.

Based on this initial observation, our work focuses on studying the performance of those queries that have shown a performance lower than the median. This set of queries constitutes 50% of the queries in the dataset and are those that are the hardest for each baseline to handle. While there are always a set of queries that are more difficult to handle by a ranker, we are interested in studying: (1) whether there are a set of queries that are *difficult* for all or the majority of the state of the art neural rankers to address, and (2) if so, would there be a ranker that can specifically address these queries. Summarily, we find that there are a significant number of queries that cannot be addressed by any of the state of the art neural rankers. We refer to these queries as *obstinate* queries[1] because of their difficulty. This means that regardless of the neural ranker, these queries will not see any performance improvements and the increase in overall performance reported by the ranker are due to improvements on another selected subset of queries. To the best of our knowledge, this issue has not been reported in the literature or by the community and deserves careful treatment, if any further headways are to be made on the stable and consistent performance of neural rankers.

We provide a complete empirical treatment of this situation and report to what extent *obstinate* queries can be observed across the state of the art neural rankers. We then systematically develop three datasets, which we refer to as *MS MARCO Chameleons*, consisting of obstinate queries that cannot be addressed by neural rankers. The objective is for the community to report the effectiveness of newly proposed rankers not only based on the full MS MARCO dataset, but also to report performance on the datasets proposed by this paper, which will show whether newly proposed methods are able to diversify the set of queries that they improve or that they are also limited to a small set of queries that have consistently been improved in the past. Furthermore, given the literature has reported that hard queries can often be due to issues such as vocabulary mismatch, and hence can be improved through query reformulation [9, 12, 24, 41], we report the performance of several strong query reformulation techniques on the *MS MARCO Chameleons* dataset and show that such queries remain stubborn and do not report noticeable performance improvements even after systematic reformulation.

The concrete contributions of this *resource-track* paper is as follows:

- We empirically study the performance of the state of the art neural rankers on the MS MARCO dataset and report the effectiveness of such methods on very obstinate queries also known as difficult queries. We report our findings on whether and to what extent the obstinate queries for the neural methods are common between them;
- We release a collection of three datasets, referred to as MS MARCO Chameleons, that consist of queries that are consistently obstinate for all neural rankers to address. The three datasets are categorized based on the level of the difficulty of their queries;
- We report the performance of the state of art neural rankers on the MS MARCO Chameleons datasets and further investigate whether the performance of these queries can be improved through query reformulation techniques.

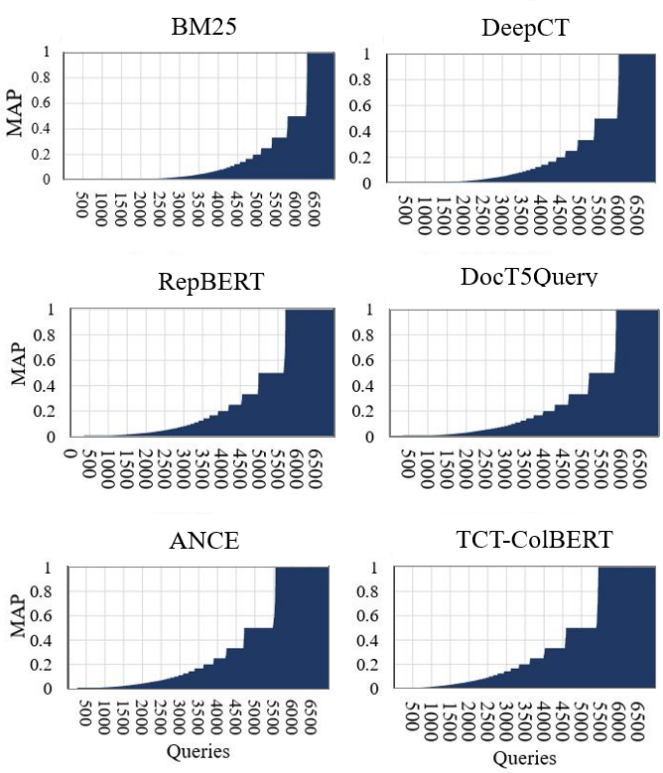

Figure 1: Performance of individual queries on the MS MARCO Dev set in terms of mean average precision. In each subfigure, the Y-axis represents MAP and the X-axis represents individual queries when sorted by MAP.

We make all three MS MARCO Chameleons datasets, the runs for all neural rankers on the three datasets, the reformulated queries, and the associated code publicly available for future research.[2]

2 EMPIRICAL STUDY ON MS MARCO

2.1 Study Setup

We perform our empirically study on the Microsoft MAchine Reading Comprehension (MS MARCO) dataset[3], which is a large-scale collection with focus on enabling the application of deep learning methods for information retrieval. Specifically, the MS MARCO dataset for passage retrieval consists of over 8.8 million passages and more than 500K pairs of queries and relevant passages for training purposes. In addition, there are 6,980 queries that are intended to be used for evaluation purposes, which is known as the MS MARCO Development set (Dev set). We focus our empirical study on the MS MARCO Dev set to measure the performance of the state of the art neural rankers and to potentially identify obstinate queries. In order to measure performance, we use the mean average precision (MAP)

[1]Some other authors have referred to these queries as hard or difficult queries.

[2]https://github.com/Narabzad/Chameleons

[3]https://microsoft.github.io/MSMARCO/

Table 2: Performance of individual baselines on their bottom X% of their worst performing queries when sorted on average precision where X =10, 20, 30, 40, 50. We also reported performance on all queries, i.e., X=100.

Evaluation	MAP						MRR@10
Method	10%	20%	30%	40%	50%	100%(All queries)	100%(All queries)
BM25	0.0000	0.0005	0.0019	0.0047	0.0098	0.1956	0.1874
DeepCT	0.0000	0.0016	0.0049	0.0108	0.0209	0.2499	0.2421
DocT5Query	0.0009	0.0042	0.0104	0.0205	0.0354	0.2850	0.2768
RepBERT	0.0010	0.0044	0.0103	0.0195	0.0344	0.3040	0.2967
ANCE	0.0017	0.0066	0.0151	0.0280	0.0476	0.3365	0.3304
TCT-ColBERT	0.0020	0.0072	0.0161	0.0299	0.0499	0.3415	0.3349
Average	**0.0011**	**0.0045**	**0.0423**	**0.0202**	**0.0351**	**0.2884**	**0.2782**

metric and compute it over the top-1000 retrieved documents. In our opinion, this is a more appropriate evaluation strategy for our purpose compared to the adopted MRR@10 in the MS MARCO leaderboard, as we would like to show the potential for improvement by considering a deeper list of retrieved documents. However, in Table 2, we report MRR@10 as well.

2.2 Ranking Baselines

While the leaderboard associated with the MS MARCO dataset consists of a multitude of neural rankers, many of them are either various interpolations of multiple methods, are multi-stage retrievers or do not come with an associated formal or informal documentation describing them. As such, we have chosen five neural rankers that are well documented and have been replicated by multiple research groups in our set of baselines. In addition, we include BM25, which has shown to be a strong traditional ranker. Based on [27], We can categorize our baselines into sparse retrievers (bag-of-word based), dense retrievers (those who utilized contextualized pre-trained embedding such as BERT) and Hybrid retrievers (those that employs both sparse and dense retrievers). These rankers are briefly introduced here:

BM25 [18][4]: We adopt the BM25 implementation provided by Anserini [44] to serve as a representative of stable traditional rankers.

DeepCT [5][5] : Dai et al. employ BERT [7] to generate a context-aware bag of words term weights for documents and queries [6]. Their proposed deep contextualized term weighting framework predicts term weights for documents and queries. The predicted weights are converted into term frequency representations in the corresponding documents and queries. Based on the modified documents, the authors apply BM25 to retrieve relevant documents for the modified version of the query.

DocT5Query [33][6]: This method is based on expanding a document in the collection with the set of queries that discriminatively represent the document content [35]. Nogueira et al. trained a T5 transformer to generate queries for any given document. These queries are then appended at the end of each document. Similar to DeepCT, DocT5Query employs a BM25 retriever to find relevant documents from amongst the expanded documents for a given query.

RepBERT [48][7]: This dense retriever represents documents and queries with fixed length contextualized embeddings to address semantic mismatch. At inference time, the inner product of the representations for queries and documents are calculated as the relevance of the document for the query.

ANCE [42][8]: Approximate nearest-neighbour Negative Contrastive Estimation (ANCE), is a training method that builds negative samples from an Approximate Nearest Neighbor (ANN) index. Similar to other BERT-based dense retrievers [8, 20, 48], ANCE uses the dot product between the learned dense representation of each query and document pair.

TCT-ColBERT [27][9]: This model builds on the TCT-ColBERT method [21] by enabling an approximate nearest neighbor search by distilling knowledge from TCT-ColBERT's MaxSim operator. In this method, the tight coupling between the teacher model and the student model enables more accurate dense representations of documents and queries, which leads to higher performance gains. TCT-ColBERT is a Hybrid approach which employs both traditional bag-of-word as well as bi-encoder architecture to do the retrieval in a single stage.

2.3 Performance on MS MARCO

In this section, we thoroughly show the performance of all the baselines introduced in Section 2.2 on the 6,980 queries of the MS MARCO Dev set. However, in addition to comparing the average performance of baselines over all queries, we dig deeper into individual queries. Figure 1 demonstrates the performance of each baseline on each query sorted based on MAP.

We observe from Figure 1 that the performance of all of the six baselines follows a long-tail distribution. This indicates that no matter which baseline method is considered, whether it be a traditional BM25 ranker or a complex neural ranker, there is a noticeable number of queries for which the rankers are unable to return any reasonable ranking. For instance, when considering the best performing dense retriever, i.e., TCT-ColBERT or ANCE, there are over 3,000 queries (out of a total of 6,980) that have an average precision of lower than 0.1. This observation lays the foundation of our work in this paper. While each recent neural ranker has extensively evaluated its

[4]https://github.com/castorini/anserini
[5]https://github.com/AdeDZY/DeepCT
[6]https://github.com/castorini/DocT5Query
[7]https://github.com/jingtaozhan/RepBERT-Index
[8]https://github.com/microsoft/ANCE
[9]https://github.com/castorini/pyserini/blob/master/docs/experiments-tct_TCT-ColBERT.md

Table 3: The number of queries that performed consistently poorly among all the baselines. We consider 10,20,30,40 and 50% of the most obstinate queries as "poorly performing" queries.

	Number of Rankers	Dataset Name	**10%**	**20%**	**30%**	**40%**	**50%**
Common in # rankers	6	`Lesser Chameleon` (Chameleons that are endanger of extinction)	95	359	705	1,170	1,693
	5	`Pygmy Chameleon` (Chameleon's that are quite rare in nature)	223	615	1,154	1,763	2,473
	4	`Veiled Chameleon` (Chameleons that are common)	400	982	1,640	2,390	3,119

performance on the MS MARCO dataset, and in cases, provided qualitative analysis of the reasons why the method performs well, to the best of our knowledge, there has not been any work that identify such a large number of poor performing queries over all neural rankers.

In order to further study the performance of the rankers at the individual query level, we arrange the queries based on their average precision. To do so, queries are classified into different difficulty groups representing the bottom X% of the worst performing queries for each baseline. We consider those queries that are in the bottom 50% of performance to be obstinate queries and those in the bottom 10% to be the most obstinate. We report five difficulty categories representing the bottom 10%, 20%, 30%, 40% and 50% of queries when sorted based on average precision.

Table 2 summarizes the performance of each baseline based on their difficulty groups. The Table clearly shows the stark difference between the overall performance of the baseline on all queries compared to their performance on each of the difficulty classes. For instance, DeepCT reports a MAP of 0 on the bottom 10% of the queries, which is equivalent to an average precision of 0 on 698 queries. The best performing neural ranker, i.e., TCT-ColBERT reports 0.002 and 0.0499 on the bottom 10% and 50% of the queries, which is equivalent to only 0.5% and 14.6% of the performance shown over all of the queries. In order to understand whether the observations on the poor performance of the rankers are generalizable across all rankers, we further investigate the degree of overlap between the different query difficulty groups of the rankers. If there is a high degree of overlap between these query groups, this shows that there is a subset of queries which cannot be effectively satisfied by any of the rankers.

3 THE PROPOSED CHAMELEON DATASETS

To study if there exist a subset of queries which are extremely obstinate and there is no retrieval method that can handle it properly, we identify the set of common queries in the X% of the most obstinate queries per ranker. We report three different versions of overlap between the queries that are placed in the bottom X% of queries for the rankers. Table 3 reports three variations, namely the number of queries that are (1) common in all six rankers, (2) at least observed in five of the six rankers and (3) at least observed in four of the six rankers within the considered X%. As seen in the table, there are a considerable number of queries in each of the three variations that are common between the different rankers. When looking at the variation that identifies common queries between at least 4 of the rankers, we observe that there are a set of 400 common queries that are in the bottom 10% of the worst performing queries of at least 4 of the rankers. This number is 95 when all worst performing queries of all rankers are considered. Similarly, there are 3,119 queries that are shared among at least four rankers in their bottom 50% of the most obstinate queries. This is approximately 45% of the queries in the MS MARCO Dev set. This becomes increasingly noticeable when pointing out that the performance of the rankers on this subset of 45% of the queries is approximately 22% of the overall reported performance of the rankers. Even when considering the strictest variation, where the queries are common among all six rankers, we find 1,693 queries in the bottom 50%, equivalent to 25% of the whole query set. Similarly performance of the rankers on this subset that consists of 25% of all MS MARCO queries is around 8% of the overall performance of the rankers.

To further investigate how the six baselines deal with the common obstinate queries, we explore the number of shared obstinate queries in each possible pair of the baselines on the X% most obstinate queries for them in Figure 2. In other words, given two rankers A and B, we define the obstinate query set $H_{X\%}(A)$ as the set of all queries that are among the most obstinate X% of both rankers. We define *agreement* of ranker A and B on obstinate queries as:

$$Agreement(A, B) = \frac{|(H_{X\%}(A) \cap H_{X\%}(B))|}{|(H_{X\%}(A) \cup H_{X\%}(B))|} \quad (1)$$

In Figure 2, the y-axis represents the agreement between each pair of rankers and the x-axis illustrates the percentage of obstinate queries that were considered. We conclude from this figure that given any pairs of rankers, regardless of whether they are based on complex transformer models or low-cost bag-of-word approaches, they share a substantial amount of obstinate queries.

Based on our analysis of the MS MARCO dataset and six rankers, we find that not only are the obstinate queries not unique for each ranker, but also, there are a large number of queries that are shared among the different rankers, which we refer to as *obstinate queries*. Based on the classification provided in Table 3, we develop and publicly share three datasets consisting of obstinate queries that were simultaneously obstinate for various rankers. These datasets include:

- `Veiled Chameleon` dataset, which consists of 3,119 queries that were common between the bottom 50% worst performing queries of at least four rankers;
- `Pygmy Chameleon` dataset that includes 2,473 queries shared between at least five rankers from the bottom 50% of the queries; and,
- `Lesser Chameleon` dataset that covers 1,693 queries which are considered to be obstinate by all six rankers.

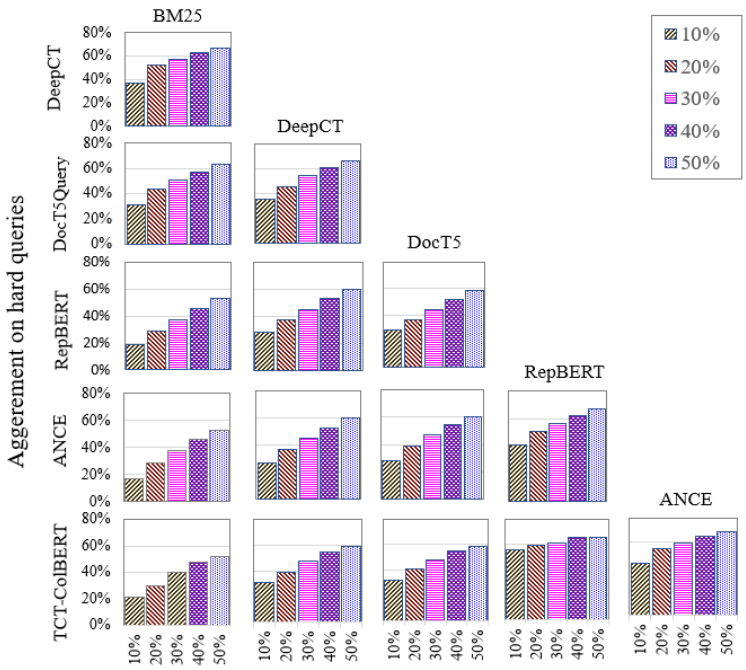

Figure 2: The degree of overlap between the most obstinate queries for each pair of baseline retrieval methods.

We provide five splits for each of these datasets as explained earlier and shown in Table 3. The performance of the six rankers for the bottom 50% of queries are reported on each of the Chameleon data sets in Table 4 in terms of MAP, nDCG and MRR@10, respectively. The reported performances show that the queries in each of the datasets are disproportionately low performing and are obstinate to address for all six rankers. Now, one might argue that the lower performance of the queries in these three datasets may be due to the lower number of relevant judgement documents for each of these queries. We report the average number of relevant judgement documents per query in each of the Chameleon datasets and compare it to the whole MS MARCO dataset in Table 5. As seen in the table, the average number of relevant documents is comparable to, and in many cases higher than, the average for the whole MS MARCO dataset. Hence the lower performance is not attributable to the relevant judgments. We also report the average length of the queries for each of the Chameleon datasets compared to the MS MARCO dataset in Table 6 and show that the average length of queries is also comparable and not a source of impact on query performance.

It is our belief that the community will significantly benefit from reporting ranker performances on these three Chameleon datasets in addition to the overall MS MARCO Dev set, as it will allow researchers to understand how each new ranker contributes to the advancement of the state of the art by showing whether a ranker improving the queries that are already treated well by other rankers, or it is in fact covering a set of queries that are obstinate to address for a host of existing strong neural rankers. The next breakthrough within this space would need to systematically address the queries that are included in the MS MARCO Chameleon datasets in order to be able to show significant improvements over existing rankers.

3.1 Correlation Between Obstinate Queries and All Queries

It is important to further investigate whether the overall performance of retrieval methods on all queries are correlated with their performance on the obstinate query sets. Figure 3 shows the mean average precision of the chameleons query sets, and all queries on y-axis and x-axis, respectively. We observe that a consistent correlation between the performance of the six baseline retrieval methods on all queries vs obstinate queries does not necessarily hold. This indicates that a better performance on all queries does not necessarily guarantee a better performance on the chameleon query sets. This is an additional indication that it is desirable to use the chameleon query sets for evaluating retrieval methods to see how a retrieval method performs over obstinate queries, which not necessarily be known when considering all queries.

3.2 Validating the Properties of Obstinate Queries

In this section, we study whether the properties of obstinate queries are relatively consistent across different retrieval methods. In Figure 4 and just as an example, we visually compare the performance of two retrieval methods on three chameleons dataset in terms of their mean average precision. As shown in this Figure, the more the query set is obstinate, the lower the performance of both retrieval methods would be on the query set. We study if this observation is consistent across all the retrieval models or not. We first note that the `Lesser Chameleon` query set includes the most obstinate queries followed by `Pygmy Chameleon` query set and then followed by the `Veiled Chameleon` query set. As such, it is expected that the retrieval performance of the retrieval methods be lowest on

Table 4: Performance of the the 6 retrieval methods on the Chameleon datasets in terms of MAP, nDCG and MRR@10 on 50% most obstinate queries.

	# common in rankers	Dataset Name	BM25	DeepCT	DocT5Query	RepBERT	ANCE	ColBERT
MAP	6	Lesser Chameleon	0.0066	0.0122	0.0185	0.0212	0.0286	0.0267
	5	Pygmy Chameleon	0.0215	0.0241	0.0403	0.0398	0.0546	0.0462
	4	Veiled Chameleon	0.0392	0.0401	0.0664	0.0560	0.0847	0.0785
nDCG	6	Lesser Chameleon	0.0894	0.1143	0.1378	0.1464	0.164	0.1617
	5	Pygmy Chameleon	0.1187	0.1378	0.1716	0.1742	0.1973	0.1937
	4	Veiled Chameleon	0.1447	0.1617	0.2029	0.1988	0.2244	0.2258
MRR@10	6	Lesser Chameleon	0.0032	0.0002	0.0032	0.0039	0.0096	0.0069
	5	Pygmy Chameleon	0.0121	0.0093	0.0223	0.0208	0.0354	0.0251
	4	Veiled Chameleon	0.0285	0.0241	0.0483	0.0412	0.0612	0.0531

Table 5: Average number of relevant documents per query for each dataset.

Dataset Name	10%	20%	30%	40%	50%	100%(All queries)
Lesser Chameleon	1.06	1.09	1.08	1.09	1.09	1.06
Pygmy Chameleon	1.05	1.08	1.082	1.08	1.08	
Veiled Chameleon	1.06	1.07	1.08	1.07	1.07	

Table 6: Average query length for each dataset.

Dataset Name	10%	20%	30%	40%	50%	100%(All querie)
Lesser Chameleon	6.30	6.72	6.62	6.51	6.45	5.92
Pygmy Chameleon	6.56	6.71	6.64	6.51	6.35	
Veiled Chameleon	6.73	6.70	6.54	6.37	6.24	

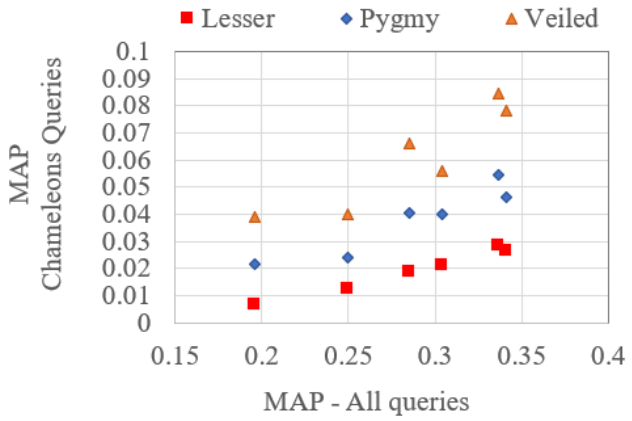

Figure 3: Mean Average Precision of queries from Chameleons datasets vs all the queries for the six different retrieval baselines.

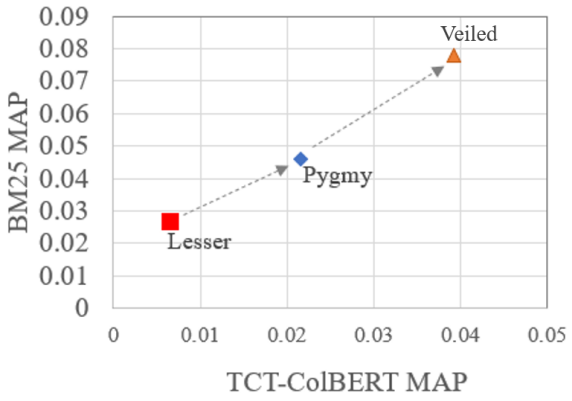

Figure 4: Comparing pair-wise performance of BM25 and TCT-ColBERT retrieval methods on the three chameleons dataset.

Lesser and highest on the Veiled Chameleon query set. This expected performance on different Chameleon query set has been visually shown on both BM25 and TCT-ColBERT. The Mean average Precision drops more notably when it comes to more obstinate query sets i.e., Pygmy and Lesser query sets, respectively. The arrows in this figure point towards easier query sets. In order to show that the relation shown in Figure 4 holds across all pairs of retrieval methods, we define a "strictly more obstinate" relation between two sets of queries. Assuming a set R consisting of n retrieval methods i.e., $R = \{R_1, R_2, ...R_n\}$, we define query set Q_1 to be *strictly more obstinate* than query set Q_2 for R as follows:

Definition 1. Query set Q_1 is strictly more obstinate than Query set Q_2 on ranker set R if and only if $P(R_i(Q_1)) < P(R_j(Q_2)) \; \forall R_i, R_j \in R$.

where $P(R_i(Q))$ shows the performance of the retrieval method R_i on query set Q. Clearly, a higher $P(R_i(Q))$ indicates that Q is easier for R_i.

We confirm that based on our empirical study on all six retrieval methods, our proposed Chameleon query sets are *strictly more obstinate* for every pair of retrieval methods compared to each other according to the following order: `Veiled` $\prec$ `Pygmy` $\prec$ `Lesser` where '$\prec$' denotes a strictly more obstinate relation. The fact that the proposed Chameleon query sets have a strictly more obstinate relation with each other shows that the queries in the `Lesser` query set are more obstinate for the six retrieval methods to satisfy compared to `Pygmy`, which is again more obstinate to satisfy for the six retrieval methods compared to the queries in `Veiled`.

4 QUERY REFORMULATION ON THE MS MARCO CHAMELEONS

One of the widely-adopted strategies often used to improve the performance of obstinate queries is to reformulate the original query into a more effective one [39, 40]. This process is known as query reformulation and can broadly be categorized into unsupervised and supervised methods. Unsupervised query reformulation methods adopt heuristics such as pseudo-relevance feedback [24] to expand or revise the original query. On the other hand, supervised reformulation methods benefit from neural architectures such as sequence-to-sequence translation architectures to learn a mapping from the source query space to a more effective revised query space. These methods often require substantial pairs of queries for training. We further analyze the proposed MS MARCO Chameleon datasets from the perspective of query reformulation and discuss whether the performance of such queries can be improved through reformulation. We adopt the ReQue toolkit [38][10] to perform both unsupervised and supervised query reformulation.

4.1 Query Reformulation Baselines

Unsupervised query reformulation methods can broadly be classified as those based on pseudo-relevance feedback and those that rely on external sources. The idea behind pseudo-relevance feedback based methods is that while complete relevance for queries cannot be determined at runtime, it can be approximated based on the documents that are initially retrieved for the query [24]. On this basis, one of the more stable and widely adopted methods, known as RM3 [1], expands a source query by selectively adding weighted terms to the query to optimize retrieval effectiveness [29, 45]. Researchers have also argued that pseudo-relevant documents may represent different aspects of a query, especially if the query is ambiguous, and have hence suggested to perform clustering on documents [2, 19], and/or terms [4] within the pseudo-relevant documents. The source query is then expanded by proportionally choosing terms from each of the clusters [3].

The other group of unsupervised query reformulation methods exploit how query terms are distributed in external knowledge sources to decide on the best reformulation of the query. For instance researchers have explored how synonymy relations defined in WordNet or thesauri [30] can be used to revise the source query [36], or semantic aspects of a query can be captured and modeled through ConceptNet [17], and or automated entity linkers [22] for the purpose of query expansion. More recent approaches have explored the possibility of expanding queries based on the relation between query terms and other terms in the corpus using pre-trained [23] or relevance-based neural embeddings [46].

Supervised query reformulation methods are essentially based on variations of neural translation models that learn a transformation from the source to the destination query spaces. For instance, Attention-based Neural Machine Translation (ANMT) [28] is a recurrent neural network based machine translation method that operates based on two attention mechanisms. Similarly, the Attend, Copy, and Generate (ACG) method is also based on an attention mechanism but additionally incorporates a copy mechanism to model term retention when performing query reformulation. As another example, Hierarchical Recurrent Encoder-Decoder (HRED) [37] is a hierarchical recurrent encoder-decoder method that performs context-aware query reformulation suggestions. In our work, given supervised methods require a large set of query pairs for training purposes, we adopt the query pair dataset proposed by Zerveas et al. [47] to train each of these methods.

4.2 Findings

We expand the queries in `Veiled Chameleon` dataset using both unsupervised and supervised query reformulation methods introduced in the previous section. Table 7 summarizes the performance of the query reformulation process. Summarily, we find that the obstinate queries in our MS MARCO Chameleon do not see any notable improvement as a result of either unsupervised or supervised query reformulation methods. In fact, in many of the cases, the performance of these queries decrease when reformulated. Most notably, the performances of ANCE [42], RepBERT [48], and TCT-ColBERT [27], which are the strongest three of the baselines have consistently seen a notable decline when the source query was reformulated. There are a few instances where the performance of the queries were improved over the base ranker such as RM3 on BM25, or Term Clustering, Thesaurus and Entity Linking on BM25 and DeepCT but these improvements do not surpass the performance of the other rankers on the same query sets. For instance, while the mean average precision of DeepCT is improved based on Entity Linking to 0.0450, this performance is still much lower than that reported by four of the other rankers using the original query itself. As a result of our study on this host of query reformulation methods, we find that the obstinate queries in our MS MARCO Chameleons dataset remain extremely obstinate to address and even uncooperative to query reformulation. This is another indication that these queries form an ideal benchmark for evaluating newly proposed rankers in the future as they remain stubborn against improvement.

5 QUALITATIVE STUDY

In this section, we provide a qualitative study on the characteristics of those queries that failed to retrieve any relevant documents in their top-1000 ranked retrieved list of documents, i.e., their MAP=0. While there are too many such queries to be shown in this paper, we found two common patterns among these queries, which can partially explain why these queries remain resilient against performance

[10] https://github.com/hosseinfani/ReQue

Table 7: Query refinement on `Veiled Chameleon`. Bold values indicate reformulated query performs better than the initial query.

		MAP on `Veiled Chameleon` query set					
	Query	**BM25**	**DeepCT**	**DocT5Query**	**RepBERT**	**ANCE**	**TCT-ColBERT**
	Original Query	0.0392	0.0400	0.0660	0.0599	0.0790	0.0780
Psuedo-Relevance Feedback	Relevance feedback	**0.0477**	**0.0574**	0.0566	0.0513	0.0277	0.0693
	RM3	**0.0407**	0.0375	0.0603	0.0459	0.0374	0.0610
	Document Clustering	0.0392	0.0393	0.0593	0.0550	0.0609	0.0765
	Term Clustering	**0.0412**	**0.0424**	0.0567	0.0557	0.0693	0.0724
External Sources	Neural Embeddings	0.0218	0.0248	0.0285	0.0409	0.0468	0.0462
	Wikipedia	0.0277	0.0313	0.0341	0.0368	0.0466	0.0396
	Thesaurus	**0.0412**	**0.0419**	0.0564	0.0560	0.0686	0.0733
	Entity Linking	**0.0399**	**0.0450**	0.0543	0.0507	0.0533	0.0649
	Sense Disambigution	0.0359	0.0360	0.0521	0.0512	0.0653	0.0633
	ConceptNet	0.0269	0.0278	0.0342	0.0369	0.0488	0.0442
	WordNet	0.0271	0.0569	0.0346	0.0359	0.0399	0.0406
Supervised Approaches	ANMT (Seq2Seq)	0.0002	0.0007	0.0010	0.0020	0.0046	0.0066
	ACG (Seq2Seq + Attention)	0.0240	0.0307	0.0359	0.0433	0.0450	0.0470
	HRED-qs	0.0006	0.0002	0.0003	0.0060	0.0082	0.0110

Table 8: Sample queries from the Chameleon Dataset.

Uncommon or incorrect terms in queries	**Complex queries**
who is albrecht diskont), ?	which function automatically counts cells that meet multiple conditions
what is the weather in vi	how long for weed to get out of your system urine test
different types of tuberculosis 2015pff	what is used to help with urinary leakage surgeries
wnli phone number	what can cause the right side of your back to hurt
side effects of flaygl antibiotics	what formula in excel returns the number of the current business day

improvement for both neural rankers as well as query reformulation methods. A set of ten such sample queries are shown in Table 7.

The first pattern that we commonly observe is the presence of uncommon or incorrectly spelled terms in the query. For instance, the incorrect spelling of flagyl in the "side effects of flaygl antibiotics" query or the likely unintentional letters typed after diskont in the "who is albrecht diskont), ?" query have resulted in a very poor performance on such queries. Neural rankers and supervised query reformulation methods will not be able to retrieve appropriate documents for these queries because one of the main terms in the query will not be recognized and will be labeled as out of vocabulary (oov), hence, being overlooked. Similarly, unsupervised query reformulation methods also do not show reasonable performance on such queries because it will not be possible to match one or more important terms of the query with those of the document collection or the external sources due to surface form mismatch.

The second common pattern relates to complex queries that require interpretation of the query beyond the immediate meaning of the terms that form the query. For instance, as shown in Table 7, the query 'what formula in excel returns the number of the current business day' requires the understanding of several subsets of the query in order to effectively retrieve relevant documents. While there are recent rankers, mostly based on transformer architectures such as BERT [7], that are sensitive to the sequence of terms in the query and documents, to the best of our knowledge, there are no methods that provide a systematic way to reason about the intention of a complex query by building an understanding of the query, as is recently done in conversational systems [43]. The development of neural rankers or neural query translation models that are dependent on term or phrase level semantics can fall short in addressing complex queries.

6 CONCLUDING REMARKS

This paper provides an empirical examination of the performance of several state of the art neural rankers on the widely adopted MS MARCO dataset. We find that while recent neural rankers have shown impressive performance improvements on the MS MARCO dataset, there is a large collection of queries that are very obstinate to address for all or most of the rankers. We further show that neither unsupervised nor supervised query reformulation methods that are often able to improve the performance of obstinate queries have any positive impact on this set of queries. In many cases, the performance of these queries decrease as a result of reformulation. We further qualitatively show that some of the queries in this set exhibit characteristics that are not easily addressable by the current approaches in neural ranking or neural query reformulation. We systematically curate, analyze and publicly release three query sets, known as MS MARCO Chameleons, which will be an additional yet essential resource for future evaluations of neural ranking techniques.

REFERENCES

[1] Nasreen Abdul-Jaleel, James Allan, W Bruce Croft, Fernando Diaz, Leah Larkey, Xiaoyan Li, Mark D Smucker, and Courtney Wade. 2004. UMass at TREC 2004: Novelty and HARD. *Computer Science Department Faculty Publication Series* (2004), 189.

[2] Massih R Amini and Nicolas Usunier. 2007. A contextual query expansion approach by term clustering for robust text summarization. In *Proceedings of DUC*, Vol. 200. Citeseer.

[3] Hiteshwar Kumar Azad and Akshay Deepak. 2019. Query expansion techniques for information retrieval: a survey. *Information Processing & Management* 56, 5 (2019), 1698–1735.

[4] Claudio Carpineto, Renato De Mori, Giovanni Romano, and Brigitte Bigi. 2001. An information-theoretic approach to automatic query expansion. *ACM Transactions on Information Systems (TOIS)* 19, 1 (2001), 1–27.

[5] Zhuyun Dai and Jamie Callan. 2019. Context-aware sentence/passage term importance estimation for first stage retrieval. *arXiv preprint arXiv:1910.10687* (2019).

[6] Zhuyun Dai and Jamie Callan. 2020. Context-aware document term weighting for ad-hoc search. In *Proceedings of The Web Conference 2020*. 1897–1907.

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

[8] Yingqi Qu Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2020. RocketQA: An Optimized Training Approach to Dense Passage Retrieval for Open-Domain Question Answering. *arXiv preprint arXiv:2010.08191* (2020).

[9] Debasis Ganguly, Dwaipayan Roy, Mandar Mitra, and Gareth JF Jones. 2015. Word embedding based generalized language model for information retrieval. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. 795–798.

[10] Luyu Gao, Zhuyun Dai, Tongfei Chen, Zhen Fan, Benjamin Van Durme, and Jamie Callan. 2020. Complementing lexical retrieval with semantic residual embedding. *arXiv preprint arXiv:2004.13969* (2020).

[11] Emma J Gerritse, Faegheh Hasibi, and Arjen P de Vries. 2020. Graph-embedding empowered entity retrieval. In *European Conference on Information Retrieval*. Springer, 97–110.

[12] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. (2016). https://doi.org/10.18653/v1/p16-1154

[13] Jiafeng Guo, Yixing Fan, Liang Pang, Liu Yang, Qingyao Ai, Hamed Zamani, Chen Wu, W Bruce Croft, and Xueqi Cheng. 2020. A deep look into neural ranking models for information retrieval. *Information Processing & Management* 57, 6 (2020), 102067.

[14] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909* (2020).

[15] Sebastian Hofstätter, Hamed Zamani, Bhaskar Mitra, Nick Craswell, and Allan Hanbury. 2020. Local self-attention over long text for efficient document retrieval. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2021–2024.

[16] Sebastian Hofstätter, Markus Zlabinger, and Allan Hanbury. 2020. Interpretable & time-budget-constrained contextualization for re-ranking. *arXiv preprint arXiv:2002.01854* (2020).

[17] Ming-Hung Hsu, Ming-Feng Tsai, and Hsin-Hsi Chen. 2006. Query expansion with conceptnet and wordnet: An intrinsic comparison. In *Asia Information Retrieval Symposium*. Springer, 1–13.

[18] K Sparck Jones, Steve Walker, and Stephen E. Robertson. 2000. A probabilistic model of information retrieval: development and comparative experiments: Part 2. *Information processing & management* 36, 6 (2000), 809–840.

[19] Inna Gelfer Kalmanovich and Oren Kurland. 2009. Cluster-based query expansion.. In *SIGIR*. 646–647.

[20] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906* (2020).

[21] 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.

[22] Adit Krishnan, Deepak P, Sayan Ranu, and Sameep Mehta. 2018. Leveraging semantic resources in diversified query expansion. *World Wide Web* 21, 4 (2018), 1041–1067. https://doi.org/10.1007/s11280-017-0468-7

[23] Saar Kuzi, Anna Shtok, and Oren Kurland. 2016. Query expansion using word embeddings. In *Proceedings of the 25th ACM international on conference on information and knowledge management*. 1929–1932.

[24] Victor Lavrenko and W Bruce Croft. 2017. Relevance-based language models. In *ACM SIGIR Forum*, Vol. 51. ACM New York, NY, USA, 260–267.

[25] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. *arXiv preprint arXiv:1906.00300* (2019).

[26] Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. 2020. Pretrained transformers for text ranking: Bert and beyond. *arXiv preprint arXiv:2010.06467* (2020).

[27] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2020. Distilling Dense Representations for Ranking using Tightly-Coupled Teachers. *arXiv preprint arXiv:2010.11386* (2020).

[28] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* (2015).

[29] Yuanhua Lv and ChengXiang Zhai. 2010. Positional relevance model for pseudo-relevance feedback. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. 579–586.

[30] Rila Mandala, Takenobu Tokunaga, and Hozumi Tanaka. 1999. Combining multiple evidence from different types of thesaurus for query expansion. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. 191–197.

[31] Bhaskar Mitra, Sebastian Hofstatter, Hamed Zamani, and Nick Craswell. 2020. Conformer-Kernel with Query Term Independence at TREC 2020 Deep Learning Track. *arXiv preprint arXiv:2011.07368* (2020).

[32] 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. In *CoCo@ NIPS*.

[33] Rodrigo Nogueira, Jimmy Lin, and AI Epistemic. 2019. From doc2query to docTTTTTquery. *Online preprint* (2019).

[34] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-stage document ranking with bert. *arXiv preprint arXiv:1910.14424* (2019).

[35] Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document expansion by query prediction. *arXiv preprint arXiv:1904.08375* (2019).

[36] Dipasree Pal, Mandar Mitra, and Kalyankumar Datta. 2014. Improving query expansion using WordNet. *Journal of the Association for Information Science and Technology* 65, 12 (2014), 2469–2478.

[37] Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. 553–562.

[38] Mahtab Tamannaee, Hossein Fani, Fattane Zarrinkalam, Jamil Samouh, Samad Paydar, and Ebrahim Bagheri. 2020. ReQue: A Configurable Workflow and Dataset Collection for Query Refinement. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 3165–3172.

[39] Arthur HM ter Hofstede, Henderik Alex Proper, and Th P van der Weide. 1996. Query formulation as an information retrieval problem. *Comput. J.* 39, 4 (1996), 255–274.

[40] Olga Vechtomova. 2009. Query Expansion for Information Retrieval. In *Encyclopedia of Database Systems*, Ling Liu and M. Tamer Özsu (Eds.). Springer US, 2254–2257. https://doi.org/10.1007/978-0-387-39940-9_947

[41] Xiao Wang, Craig Macdonald, and Iadh Ounis. 2020. Deep Reinforced Query Reformulation for Information Retrieval. *arXiv preprint arXiv:2007.07987* (2020).

[42] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808* (2020).

[43] Liu Yang, Minghui Qiu, Chen Qu, Cen Chen, Jiafeng Guo, Yongfeng Zhang, W Bruce Croft, and Haiqing Chen. 2020. IART: Intent-aware Response Ranking with Transformers in Information-seeking Conversation Systems. In *Proceedings of The Web Conference 2020*. 2592–2598.

[44] Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the use of Lucene for information retrieval research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1253–1256.

[45] Ruifan Yu, Yuhao Xie, and Jimmy Lin. 2019. Simple techniques for cross-collection relevance feedback. In *European Conference on Information Retrieval*. Springer, 397–409.

[46] Hamed Zamani and W Bruce Croft. 2017. Relevance-based word embedding. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 505–514.

[47] George Zerveas, Ruochen Zhang, Leila Kim, and Carsten Eickhoff. 2020. Brown University at TREC Deep Learning 2019. *arXiv preprint arXiv:2009.04016* (2020).

[48] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2020. RepBERT: Contextualized Text Embeddings for First-Stage Retrieval. *CoRR* abs/2006.15498 (2020). arXiv:2006.15498 https://arxiv.org/abs/2006.15498