Re$^2$G: Retrieve, Rerank, Generate

Michael Glass$^1$, Gaetano Rossiello$^1$, Md Faisal Mahbub Chowdhury$^1$,
Ankita Rajaram Naik$^{1,2}$, Pengshan Cai$^{1,2}$, Alfio Gliozzo$^1$

$^1$ IBM Research AI, Yorktown Heights, NY, USA
$^2$ University of Massachusetts Amherst, MA, USA

Abstract

As demonstrated by GPT-3 and T5, transformers grow in capability as parameter spaces become larger and larger. However, for tasks that require a large amount of knowledge, non-parametric memory allows models to grow dramatically with a sub-linear increase in computational cost and GPU memory requirements. Recent models such as RAG and REALM have introduced retrieval into conditional generation. These models incorporate neural initial retrieval from a corpus of passages. We build on this line of research, proposing Re$^2$G, which combines both neural initial retrieval and reranking into a BART-based sequence-to-sequence generation. Our reranking approach also permits merging retrieval results from sources with incomparable scores, enabling an ensemble of BM25 and neural initial retrieval. To train our system end-to-end, we introduce a novel variation of knowledge distillation to train the initial retrieval, reranker and generation using only ground truth on the target sequence output. We make our code available as open source.

1 Introduction

GPT-3 [Brown et al., 2020] and T5 [Raffel et al., 2020] are arguably the most powerful members in a family of deep learning NLP models called transformers. Such models store surprising amount of world knowledge. They have been shown to produce good performance on a range of demanding tasks, especially in generating human like texts. However, such large transformers’ capability is tied to the increasingly larger parameter spaces on which they are trained. Recently, there has been work towards transformers that make use of non-parametric knowledge. REALM (Retrieval Augmented Language Model) [Guu et al., 2020] and RAG (Retrieval Augmented Generation) [Lewis et al., 2020b] both use an indexed corpus of passages to support conditional generation. By using the corpus as a source of knowledge these models can extend the information available to the model by tens or even hundreds of gigabytes with a sub-linear scaling in computation cost.

These recent advancements, in turn, have been inspired by BART (Bidirectional and AutoRegressive Transformer) [Lewis et al., 2020a] that combines a Bidirectional Encoder (e.g. BERT [Devlin et al., 2019]) with an Autoregressive decoder (e.g. GPT [Brown et al., 2020]) into one sequence-to-sequence model.

We build on this line of research, pioneered by REALM and RAG, and propose a new approach that we call Re$^2$G (Retrieve, Rerank, Generate), which combines both neural initial retrieval and reranking into a BART-based sequence-to-sequence generation.

There are two particular aspects on which our approach is different from the previous works. Firstly, our reranking approach permits merging retrieval results from sources with incomparable scores, e.g. enabling an ensemble of BM25 and neural initial retrieval. Secondly, to train our system end-to-end, we introduce a novel variation of knowledge distillation to train the initial retrieval, reranker and generation using only ground truth on the target sequence output.

The KILT benchmark [Petroni et al., 2021] has been recently introduced to evaluate the capabilities of pre-trained language models to address NLP tasks that require access to external knowledge. We evaluate on four diverse tasks from KILT: slot filling, question answering, fact checking and dialog. Figure 1 shows examples of these tasks. Re$^2$G
makes significant gains on all four tasks, reaching the top of the KILT leaderboards and establishing a new state-of-the-art.

The contributions of this work are as follows:

- We introduce Re$^2$G, demonstrating the effectiveness of reranking for generative language models that incorporate retrieval.
- We further extend Re$^2$G by ensembling initial retrieval methods, combining neural and traditional keyword-based approaches.
- Re$^2$G improves the current state-of-the-art of 9%, 31%, 34%, 22% and 10% relative gains on the headline KILT metrics for T-REx (slot filling), Natural Questions (question answering), TriviaQA (question answering), FEVER (fact checking), and Wizard of Wikipedia (dialog), respectively.
- We publicly release our code as open source to support continued development.

2 Related Work

The KILT benchmark and public leaderboard combines eleven datasets across five tasks. The main advantage of the KILT distribution of these datasets is that the provenance information from each dataset is realigned to reference the same snapshot of Wikipedia. A unified evaluation script and set of metrics is also provided. In this work, we focus on four tasks, such as Slot Filling [Levy et al., 2017, Elsahar et al., 2018], Question Answering [Kwiatkowski et al., 2019, Joshi et al., 2017], Fact Checking [Thorne et al., 2018a,c], and Dialog [Dinan et al., 2019] (see Figure 1).

A set of baseline methods have been proposed for KILT. GENRE [Cao et al., 2021] is trained on BLINK [Wu et al., 2020] and all KILT tasks jointly using a sequence-to-sequence language model to generate the title of the Wikipedia page where the answer can be found. This method is a strong baseline to evaluate the retrieval performance, but it does not address the downstream tasks. On the other hand, generative models, such as BART [Lewis et al., 2020a] and T5 [Raffel et al., 2020], show interesting performance when fine-tuned on the downstream tasks relying only on the implicit knowledge stored in the weights of the neural networks, without the use of any explicit retrieval component.

RAG [Lewis et al., 2020b], an end-to-end retrieval-based generative, is the best performing baseline in KILT and it incorporates DPR [Karpukhin et al., 2020] to first retrieve relevant passages for the query, then it uses a model initialized from BART [Lewis et al., 2020a] to perform a sequence-to-sequence generation from each evidence passage concatenated with the query in order to generate the answer. Figure 2 shows the architecture of RAG.

Multi-task DPR [Maillard et al., 2021] exploits multi-task learning by training both DPR passage and query encoder on all KILT tasks. DensePhrases [Lee et al., 2021] addresses the knowledge intensive tasks with a short answer, such as slot filling. It indexes the phrases in the corpus that can be potential answers. The extracted phrases are represented by their start and end token vectors from the final layer of a transformer initialized from SpanBERT [Joshi et al., 2020].

Knowledge Graph Induction (KGI) [Glass et al., 2021] combines DPR and RAG models, both trained with task and dataset specific training. KGI employs a two phase training procedure: first training the DPR model, i.e. both the query and context encoder, using the KILT provenance ground truth. Then, KGI trains the sequence-to-sequence generation and further trains the query encoder using only the target output as the objective. This results in large improvements in retrieval performance and, as a consequence, in the downstream tasks.

KILT-WEB 2 [Piktus et al., 2021] addresses the KILT tasks by broadening the knowledge source used. Rather than rely only on KILT’s Wikipedia snapshot, KILT-WEB 2 creates SPHERE as a knowledge source. SPHERE is built from CCNet [Wenzek et al., 2020] and over twenty times the size of the Wikipedia corpus. It can use either BM25 or DPR retrieval (though not both combined) followed by a ‘reader’ component, but not trained end-to-end. The reader component is the Fusion-in-Decoder [Izacard and Grave, 2021] model, where retrieved documents are encoded independently, then their encoded representations are concatenated for the decoder.

SEAL [Bevilacqua et al., 2022] introduces a novel generative approach to retrieval. Rather than generating the unique document identifier like GENRE, SEAL can generate any ngrams present
Dracula is an 1897 Gothic horror novel by Irish author Bram Stoker. It introduced the character of Count Dracula, and established many conventions of subsequent vampire fantasy. The novel tells the story of Dracula’s attempt to move from Transylvania to England so that he may find new blood and spread the undead curse, and of the battle between Dracula and a small group of men and a woman led by Professor Abraham Van Helsing.

Multi-stage or cascade approaches to retrieval have received ample attention in Information Retrieval (IR) research. The multi-stage approach begins with the initial retrieval phase, where an initial set of documents or passages form the pool of candidates to be considered for ranking. Then one or more phases of increasingly computationally demanding rerankers are applied. Early approaches in learning to rank [Liu, 2009] used features and linear classifiers. Pre-trained language models, especially BERT [Devlin et al., 2019], have shown state-of-the-art performance when applied to the task of relevance ranking. Transformers may be applied as classifiers to each query and passage pair independently [Nogueira and Cho, 2019] or as generators to produce labels for passages in a sequence-to-sequence model [Nogueira et al., 2020].

3 Methodology

The approach of RAG, Multi-DPR, and KGI is to train a neural IR (Information Retrieval) component and further train it end-to-end through its impact in generating the correct output. Figure 2 illustrates the end-to-end RAG system. It has been previously established that results from initial retrieval can be greatly improved through the use of a reranker [Liu, 2009, Wang et al., 2011]. Therefore we hypothesized that natural language generation systems incorporating retrieval can benefit from reranking.
In addition to improving the ranking of passages returned from DPR, a reranker can be used after merging the results of multiple retrieval methods with incomparable scores. For example, the scores returned by BM25 [Robertson and Zaragoza, 2009] are not comparable to the inner products from DPR. Using the scores from a reranker, we can find the top-k documents from the union of DPR and BM25 results. Figure 3 illustrates our extension of RAG with a reranker. We call our system Re²G (Retrieve, Re-rank, Generate).

3.1 Reranker

The reranker we use is based on the sequence-pair classification of Nogueira and Cho [2019]. This model is shown in Figure 4. The query and passage are input together to a BERT [Devlin et al., 2019] transformer. Cross attention is applied over the tokens of both sequences jointly. This is called an interaction model.

This model contrasts with the representation model used for initial retrieval. Figure 5 shows the bi-encoder representation model for DPR. The representation vectors for the query and passage are produced independently. This allows for efficient retrieval by pre-computing vectors for all passages in the corpus and indexing them with an ANN (Approximate Nearest Neighbors) index. By using an interaction model to rerank the top-N passages from the representation model, we can get the advantages of both model types: accuracy and scalability.

We initialize the reranker from the BERT model trained on MS MARCO [Nguyen et al., 2016] by NBoost [Thienes and Pertschuk, 2019] and available through Hugging Face3.

3.2 Training

As Figure 1 illustrates, KILT tasks are provided with two types of ground truth: the target output sequence and the provenance information indicating the passage or passages in the corpus that support the output.

Our training is carried out in four phases: DPR training, generation training, reranking training, and full end-to-end training. The initial DPR and reranking phases make use of the provenance ground truth. The generation and full end-to-end training make use of only the target output.

Formally:

- The original KILT instances are a tuple: \(\langle q, t, \text{Prov} \rangle\) where \(q\) is the input or prompt, \(t\) is the target output, and \(\text{Prov}\) is the set of provenance passages that support the target output.
- DPR training is a tuple: \(\langle q, p^+, p^- \rangle\) where \(p^+ \in \text{Prov}\) and \(p^- \notin \text{Prov}\)
- Reranking training begins with the application of DPR and BM25, producing tuples: \(\langle q, P, \text{Prov} \rangle\) where \(P = \text{BM25}(q) \cup \text{DPR}(q)\)
- Generation and end-to-end training instances are pairs of query and target: \(\langle q, t \rangle\)

The first two phases, DPR and generation, are identical to KGI, specifically KGI₀. We use the codes from Glass et al. [2021]4.

DPR Stage 1 training is the same training used by Karpukhin et al. [2020]. The triplets of query, positive passage and “hard negative” passages from BM25 are put into batches of 128 instances. The positives and hard negatives from other instances form the “batch negatives” for each instance. The DPR bi-encoder model gives each query a probability distribution over the positive, hard negative, and batch negatives. The loss is the negative log-likelihood for the positive. After DPR Stage 1 training the passages from the corpus are indexed with a Hierarchical Navigable Small World (HNSW) [Malkov and Yashunin, 2018] using FAISS [Johnson et al., 2017].

---

3 https://huggingface.co/nboost/pt-bert-base-uncased-msmarco
4 https://github.com/IBM/kgi-slot-filling
Generation training extends the training of the query encoder and trains the BART\textsubscript{LARGE} sequence-to-sequence model on the target sequence output. This training is the same as that described by Lewis et al. [2020b].

3.3 Reranking Training

The next phase, training the reranking in isolation, begins with gathering the initial retrieval results from DPR and BM25 on the training set. These results are merged and used as training data for the reranker.

In some datasets there are multiple positive passages. Therefore, we use the negative of the summed log-likelihood for the positive passages as the loss function. The logits given by the reranker are $z_r$ and the indices for the correct passages (from the ground truth provenance) are $\text{Prov}$.

$$loss = - \sum_{i \in \text{Prov}} \log(\text{softmax}(z_r)_i)$$

3.4 End-to-End Training

Training end-to-end poses a special challenge. In RAG, the gradient propagates to the query encoder because the inner product between the query vector and the passage vector is used to weight the influence of each sequence, a process RAG calls marginalization. The inputs to the BART model are sequences ($s_j = p_j \text{ [SEP]} q$) that comprise a query $q$ plus retrieved passage $p_j$. The probability for each sequence is determined from the softmax over the retrieval (or reranker) scores for the passage. The probability for each target token $t_i$ given the sequence $s_j$ is a softmax over BART’s token prediction logits. The loss therefore is a negative log-likelihood summed over all target tokens and sequences, weighted by each sequence’s probability.

Consider that in Re$^2$G the score from the reranker, not the initial retrieval, is used to weight the impact of each sequence in generation. This allows the reranker to be trained through the ground truth on target output, but it means the gradient for the query encoder will be zero since the marginalization no longer depends on the inner product from the query and passage representation vectors.

$$P(s_j) = \text{softmax}(z_r)_j$$

$$P(t_i | s_j) = \text{softmax}(\text{BART}(s_j)_i | t_i)$$

$$loss = - \sum_{i,j} \log (P(t_i | s_j) \cdot P(s_j))$$

We consider three possible resolutions to this issue.

- Combine the DPR and reranker scores
- Freeze the query encoder
- Online Knowledge Distillation

The first candidate solution is tempting but fatally flawed. By adding the log softmax from DPR and the reranker we can ensure that both systems are trained through impact in generation. However, if the DPR score is added to the reranker score, then the DPR score is being trained to provide a complementary signal to the reranker. Therefore, when DPR is used to gather the candidate passages, it does not give the highest scores to the passages that are most likely to be relevant, but instead gives the highest scores to the passages the reranker is most likely to underrate. We find that this theoretical concern is also a practical concern, as DPR performance (and overall system performance) declines greatly when trained in this way.

The simplest solution is to freeze the parameters of the query encoder, training only the reranker and generation components. We find this is indeed the best solution for one of our datasets, Wizard of Wikipedia. Note that DPR has already been trained in two phases, first from the provenance ground truth and then again in generation training in the RAG model.

The third solution is our novel application of knowledge distillation [Hinton et al., 2015]. We use the reranker as a teacher model to provide labels to the DPR student model. We distill the knowledge across architectures: from an interaction model to a representation model. Further, this knowledge distillation occurs online, while the reranker is being trained. The loss for the initial retrieval is therefore the KL-divergence between the probability distribution it gives over the retrieved passages and the reranker’s probability distribution over the same passages. A temperature hyperparameter $T$ smooths these distributions to prevent excessive loss and stabilize training.
Table 1: KILT leaderboard top systems

The knowledge distillation has the usual advantage of providing signal not only of positive and negative instances, but degrees of negativeness. In addition, since we retrieve \( n = 12 \) passages from DPR but only use the top-\( k \) (\( k = 5 \)) for generation, the knowledge distillation loss is providing a (soft) label for more passages.

3.5 Inference

At inference time the query is encoded using the DPR query encoder and the top-12 passages from the HNSW index are returned. The query is also passed to BM25 search, specifically Anserini\(^5\), gathering the top-12 BM25 results. Both sets of passages are passed to the reranker and scored. The top-5 passages are then joined with the query and passed to BART\(^{LARGE}\) to generate the output. The five output sequences are weighted according to the softmax over the reranker scores to produce the output.

\[ loss = D_{KL} \left( \frac{softmax \left( \frac{z_h}{T} \right)}{softmax \left( \frac{z_t}{T} \right)} \right) \cdot T^2 \]

\(^5\)https://github.com/castorini/anserini
4 Experiments

We test our model on five datasets, over four distinct tasks in the KILT benchmark: slot filling, question answering, fact checking and dialog. Figure 1 shows an example of these four tasks.

The slot filling dataset, T-REx [Elsahar et al., 2018], provides as input a head entity and relation, and expects as output the entity or term that fills the slot, also called the tail entity. The T-REx dataset contains 2.3M instances. We use only 370k training instances by downsampling the relations that occur more than 5000 times. This reduces the training time required while keeping state-of-the-art performance. The development and test sets each have 5k instances.

The question answering datasets are “open” versions of Natural Questions [Kwiatkowski et al., 2019] and TriviaQA [Joshi et al., 2017]. Unlike the original versions, the relevant Wikipedia page must be found by a retrieval step. The training sets for Natural Questions and TriviaQA contain 87k and 62k questions, with another 3k and 5k for the development and 1.4k and 6.5k for test.

The fact checking dataset in KILT is FEVER (Fact Extraction and VERification). It is a combination of the two FEVER versions [Thorne et al., 2018b, 2019] omitting the NOTENOUGH-INFO class. There are approximately 10k instances in the development and test sets each and 5k for the development and 1.4k and 6.5k for test.

The dialog dataset is Wizard of Wikipedia [Dinan et al., 2018] is the dialog dataset. The input is a short dialog history ending with the information seeker’s turn. The expected output is a fact presented conversationally or just an utterance or question mentioning content from a relevant Wikipedia page. It is the smallest dataset with approximately 3k instances in development and test and 64k in train.

For all tasks, systems are expected to produce the target output as well as justify it with provenance information from the KILT knowledge source. The metrics of R-Precision and Recall@5 measure the correctness of the provenance. R-Precision measures what fraction of the $R$ documents in the ground truth provenance ($|Prov| = R$) are present in the top-$R$ documents returned by the system. Accuracy and (token-level) F1 measure the correctness of the generated output.

4.1 Retrieval

Table 2 examines how the retrieval improves through each step of training. In the first half of the table we consider the initial retrieval alone. DPR Stage 1 is the DPR training described earlier - training only from the provenance ground truth with batch negatives and hard negatives from BM25. KGI$_0$ further trains the query encoder of DPR Stage 1 through its impact in generating the target output. Finally Re$^2$G extends the training of DPR with online knowledge distillation from the reranker. This step is beneficial in two of the three datasets, while the previous steps improve performance across all datasets.

In the second half of the table we examine the improvement in reranking. The baseline of KGI$_0$ DPR+BM25 merges the results of KGI$_0$’s DPR and BM25 by scoring each passage by the sum of the inverse rank from each method. For both T-REx and FEVER, even this simple approach to ensembling DPR and BM25 improves Recall@5, although not R-Precision. Following reranker training using the provenance ground truth (Reranker Stage 1), we find improvement over DPR across all five datasets on both retrieval metrics. The reranker’s improve-
Table 2: Development Set Results for Retrieval

<table>
<thead>
<tr>
<th></th>
<th>T-REx</th>
<th>NQ</th>
<th>TriviaQA</th>
<th>FEVER</th>
<th>WoW</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R-P@5</td>
<td>R-P@5</td>
<td>R-P@5</td>
<td>R-P@5</td>
<td>R-P@5</td>
</tr>
<tr>
<td>BM25</td>
<td>46.88</td>
<td>69.59</td>
<td>24.99</td>
<td>42.57</td>
<td>42.73</td>
</tr>
<tr>
<td>DPR Stage 1</td>
<td>49.02</td>
<td>63.34</td>
<td>56.64</td>
<td>64.38</td>
<td>60.12</td>
</tr>
<tr>
<td>KG0 DPR</td>
<td>65.02</td>
<td>75.52</td>
<td>64.65</td>
<td>69.60</td>
<td>60.55</td>
</tr>
<tr>
<td>Re2G DPR</td>
<td>67.16</td>
<td>76.42</td>
<td>65.88</td>
<td>70.90</td>
<td>62.33</td>
</tr>
<tr>
<td>KG0 DPR+BM25</td>
<td>60.48</td>
<td>80.06</td>
<td>36.91</td>
<td>66.94</td>
<td>40.81</td>
</tr>
<tr>
<td>Reranker Stage 1</td>
<td>81.22</td>
<td>87.00</td>
<td>70.78</td>
<td>73.05</td>
<td>71.80</td>
</tr>
<tr>
<td>Re2G Reranker</td>
<td>81.24</td>
<td>88.58</td>
<td>70.92</td>
<td>74.79</td>
<td>90.06</td>
</tr>
</tbody>
</table>

Table 2: Development Set Results for Retrieval

The point estimates and 95% confidence intervals are reported. Re2G-KD excludes the online knowledge distillation, instead freezing the query encoder when training the reranker and generator during end-to-end training. Re2G-BM25 excludes BM25 results, fetching 24 passages from DPR rather than 12 from DPR and 12 from BM25. The passages are still reranked. KG0 is the baseline system, without a reranker and therefore also without BM25 results or online knowledge distillation during training.

Both online knowledge distillation and ensembling with BM25 improve performance in four out of five datasets. Online knowledge distillation failed to improve for Wizard of Wikipedia and ensembling with BM25 failed to improve for Natural Questions.

5 Analysis

Since the Re2G model differs from the KGI model only in the retrieval phase, we hypothesized that its gains in output quality are driven by its better retrieval quality. To test this hypothesis we considered all cases where the Re2G model produces better output than the KG0 model and calculated the fraction of such cases where Re2G’s rank for the first correct passage is lower than KG0’s.

We find that for T-REx, NQ, and FEVER the fractions of output gains that could be attributed to improved retrieval and ranking are 67.73%, 61.08% and 66.86% respectively. While for TriviaQA and Wizard of Wikipedia only 36.86% and 27.74% of output improvements were accompanied by improved ranking for the correct passage. It is important to note that in Wizard of Wikipedia, many of these improved outputs have only a small gain in token-level F1.

While much of the gain in output quality is attributable to improved recall, at least a third is not. This reinforces an observation of Glass et al. [2021], that models trained with better retrieval can produce better output even when the retrieved passages are equivalent at test time.

5.1 Slot filling error analysis

To understand the types of errors Re2G makes we sampled 50 instances of the development set of the T-REx dataset where the Accuracy and token-level F1 score was zero.

Interestingly, the most common class of error (33/50) was due to the incompleteness of the ground truth. Often the head entity is ambiguous (19/50), or the relation has multiple fillers (16/50).

As an example, consider the following where there are two Joe O’Donnell notable for sports in the passages retrieved, and each played for at least two different teams.

Joe O’Donnell [SEP] member of sports team
Target: Buffalo Bills
Re2G: Dumbarton F.C.

- Joe O’Donnell (footballer) / Joe O’Donnell (footballer) Joseph ‘Joe’ O’Donnell (born 3 March 1961) was a Scottish footballer who played for Dumbarton and Stranraer.
- Joe O’Donnell (American football) / ... fullback, guard and tackle for the University of Michigan from 1960 to 1963. He also played professional football as a guard and tackle for eight seasons for the Buffalo Bills...

When Re2G produces genuine errors it is usually
because it has selected some entity as a filler related in a different way (6/17) or it has failed to retrieve the necessary passage (9/17).

6 Conclusions

Re²G considerably advanced the state-of-the-art across five KILT datasets, and still holds the top position in four of the five. Relative to previous work, such as RAG or KGI, Re²G substantially improves both in retrieval and end-to-end performance on slot filling, question answering, fact checking, and dialog. The reranker alone improves performance and enables the inclusion of multiple sources of initial retrieval. This architecture permits us to integrate results from BM25, further improving accuracy. Our online knowledge distillation is able to improve the performance of DPR in four of the five datasets, despite the loss in end-to-end training not depending on the DPR scores. Similarly, the ensembling of DPR and BM25, which is enabled by our incorporation of a reranker, benefits four of the five datasets tested. We have directed our efforts towards improving the retrieval of relevant knowledge. This also enables improvement in end-to-end performance by supplying better passages to the generation component. Further experiments on domain adaptation of Re²G on tasks like question answering or dialog might provide useful insight on the application of this technology to real world use cases. We are releasing our source code as open source (Apache 2.0 license) to enable further research.

Table 3: Development Set Results for Re²G Variations
References


James Thorne, Andreas Vlachos, Oana Cocarascu, Christos Christodoulopoulos, and Arpit Mittal. The fact extraction and verification (FEVER) shared task. CoRR, abs/1811.10971, 2018c.


Appendix

A Hyperparameters

We have not done hyperparameter tuning for DPR Stage 1, Generation, or Reranking training. Instead we used hyperparameters similar to the original works on training DPR, BERT reranking and RAG. Table 4 shows the hyperparameters used in our experiments.

For knowledge distillation we used the same hyperparameter settings as Generation. For the additional hyperparameters in online knowledge distillation: temperature and KD learn rate scaling, we experimented with temperatures of 10 and 40 and KD learn rate scaling of 1.0 and 0.1. For our reported results we used a temperature of 10.0 and a learn rate scaling of 1.0.

When training using online knowledge distillation, there is a separate optimizer for the query encoder while training generation. This optimizer uses the same hyperparameter settings.

Table 6 shows the settings for retrieval and generation used for all datasets.

All results are from a single run. The random seed for python, numpy and pytorch was 42.

B Software Details

We used the following software versions:

- Ubuntu 18
- Pytorch 1.7
- Transformers 4.3.2
- Anserini 0.4.1 (commit 3a60106d6c83473d147218d78ac7dca7c3b6d47c)

C Model Details

Number of parameters Re²G uses three BERTBASE transformers: query encoder, passage encoder and reranker. Each has 110M parameters. The generation component is a BARTLARGE model with 400M parameters. There are 730M parameters in total.

Computing infrastructure Using a single NVIDIA V100 GPU DPR training of two epochs takes approximately 24 hours for T-REx and less than 12 hours for FEVER and WoW.

Using a two NVIDIA P100 GPUs generation training for 370k T-REx instances takes two days, while FEVER and WoW training completes in half a day.

The FAISS index on the KILT knowledge source requires a machine with large memory, we use machines with 128GB of memory.
### Table 4: Re^2G hyperparameters

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>DPR</th>
<th>Reranker</th>
<th>Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>learn rate</td>
<td>5e-5</td>
<td>3e-5</td>
<td>3e-5</td>
</tr>
<tr>
<td>batch size</td>
<td>128</td>
<td>32</td>
<td>128</td>
</tr>
<tr>
<td>epochs</td>
<td>2</td>
<td>1</td>
<td>1*</td>
</tr>
<tr>
<td>warmup instances</td>
<td>0</td>
<td>10%</td>
<td>10%</td>
</tr>
<tr>
<td>learning schedule</td>
<td>linear</td>
<td>triangular</td>
<td>triangular</td>
</tr>
<tr>
<td>max grad norm</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>weight decay</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Adam epsilon</td>
<td>1e-8</td>
<td>1e-8</td>
<td>1e-8</td>
</tr>
</tbody>
</table>

Table 4: Re^2G hyperparameters

### D Generation Analysis

We examined 20 instances coupled with 3 output texts: the baseline KGI_0, Re^2G, and the target text in the ground-truth. The three output texts were presented unlabeled and in random order to avoid bias. For each instance, we read the conversation history and then mark each text either GOOD, OK or INCONSISTENT generation. To our surprise, 5/20 ground-truth target texts are INCONSISTENT which indicates the WoW benchmark might have limitations in annotation quality. Both the systems have similar results (GOOD/OK/INCONSISTENT - Re^2G: 8/2/10; KGI_0: 9/2/9).

Second, we checked a set of 20 WoW instances where Re^2G’s F1 score was in the bottom quintile. The conversation history was presented along with Re^2G generated text and the passages retrieved. Manual examination showed 8/20 as INCONSISTENT and in 4/8 cases supporting ground-truth passages were not retrieved. Below is one of the 12/20 cases where Re^2G generated text was found CONSISTENT with respect to the conversation history, although it has low F1 and Rouge-L scores.

**Conversation History:**

- My favorite color is red.
- Red is at the end of the spectrum of light, its with orange and opposite of violet.
- I didn’t know that. What else do you know about red?

**Target:** It’s actually a primary color for the RGB and CMYK color model.

**Re^2G:** It has a dominant wavelength of approximately 625-740 nanometres.

### D.1 Generation Quality

Table 7 shows couple of examples that were part of the set of randomly selected instances from WoW dataset and used for manual inspection. We choose these two particular instances to show when we thought the ground truth (i.e. target) is not coherent with respect to the corresponding conversation history.

In the first example, the system generated outputs were judged as coherent. We found that both Re^2G and KGI_0 retrieved the following passage which might have helped generation of the above output -

Horseshoe Falls / Horseshoe Falls
Horseshoe Falls, also known as Canadian Falls, is the largest of the three waterfalls that collectively form Niagara Falls on the Niagara River along the Canada–United States border. Approximately 90% of the Niagara River, after diversions for hydropower generation,
flows over Horseshoe Falls. The remaining 10% flows over American Falls and Bridal Veil Falls. It is located between Terrapin Point on Goat Island in the US state of New York, and Table Rock in the Canadian province of Ontario. Section: International border.

As for the ground truth, we marked it (factually) inconsistent based on the following retrieved passage -

Niagara Falls / Located on the Niagara River, which drains Lake Erie into Lake Ontario, the combined falls have the highest flow rate of any waterfall in North America that has a vertical drop of more than . During peak daytime tourist hours, more than 168,000 m (six million cubic feet) of water goes over the crest of the falls every minute. Horseshoe Falls is the most powerful waterfall in North America, as measured by flow rate.

In the second example, all three texts were marked inconsistent. Interestingly, all the items in the conversation contains subjective opinion. Consequently, all the three candidate texts also contains subjective opinion. The problem is both the systems generated texts that are almost repetition of earlier conversation. In case of the ground truth, we find that the text is semantically incoherent.

We have also submitted files that contain all instances that were used to generate the different analysis reported in Section 4.2 of the paper. These files also contains our annotations/remarks where applicable.
Conversation history:

- Have you ever been to Niagara Falls? I want to go there, but I know it’s very far. Those beautiful 3 waterfalls are right between New York and Ontario, thousands of miles from me!
- I never have but I would love to. It looks beautiful.
- Yes, they do! They are different sized waterfalls, named Horseshoe, American and Bridal Veil Falls.
- I had no idea. I’ve never heard that before. Very cool.
- Yes, and Horseshoe Falls in actually the most powerful waterfall in all of the US!
- I never new any details about it. I have just seen it in pictures and on TV and it looks very neat.

Re³G: Horseshoe falls, also known as Canadian Falls, is the largest of the three waterfalls that collectively form Niagara Falls on the Niagara River along with American Falls and Bridal Veil Falls.

KGI: Horseshoe falls is also known as Canadian Falls, which straddles the international border between Canada and the U.S.

Ground truth: It’s really quite interesting and beautiful. There’s over six million cubic feet of water that crosses the fall’s crest every 60 seconds. Can you imagine?

[GOOD]

Table 7: Some of the randomly selected instances from WoW benchmark that were evaluated by a human evaluator. The comments inside [..] are the feedback provided by the evaluator.