UNIVERSITATEA POLITEHNICA BUCUREŞTI
FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

PROIECT DE DISERTAŢIE

Sistem de inferență pentru întrebări cu variante multiple de răspuns

George-Sebastian Pîrtoacă

Coordonator științific:
Conf.dr.ing. Traian Rebedea

BUCUREŞTI

2020

POLITECHNICA UNIVERSITY OF BUCHAREST
FACULTY OF AUTOMATIC CONTROL AND COMPUTERS
COMPUTER SCIENCE DEPARTMENT

MASTER'S THESIS

Multiple-Choice Question Answering

George-Sebastian Pîrtoacă

Thesis advisor
Conf.dr.ing. Traian Rebedea

BUCHAREST

2020

Table of Contents

1. Introduction

Question answering (QA) is one of the most challenging tasks in natural language processing as it requires complex reasoning [1, 5] on the top of commonsense knowledge [2, 3, 4] and general text comprehension. This dissertation paper focuses on text-only QA which means that questions are expressed solely in natural language and they do not refer to images, tables, graphs, or any kind of structured data. Different QA settings are testing various abilities:

a) *Closed-book question answering* – no supporting context or external knowledge is provided to the input. In such a case, the QA systems may employ an information retrieval component to extract knowledge relevant to the question at hand;
b) *Open-book question answering* – a gold paragraph or knowledge source is provided with all the necessary information to answer the question (commonsense knowledge may still be required). This setting gives more importance to the reasoning component [6, 7].

The closed-book QA scenario is more challenging than open-book QA as it also assessing the system's ability to extract relevant pieces of information. In this work, we focus on both closed-book QA and open-book QA by experimenting with various datasets. Traditionally, there has been a considerable amount of research invested into designing robust QA models starting with rule-based systems [8], or probabilistic models [9, 10] and ending with deep-learning solutions [11, 12, 13].

The rest of this dissertation thesis is structured as follows. Firstly, we provide an in-detail overview of the current QA methods, including a part targeted at general natural language understanding and machine reading comprehension. Secondly, we propose two methods that are slightly divergent in their approaches towards QA: one focuses on both reasoning and passage re-ranking (e.g. extracting better supporting contexts) and the other one deals with transfer learning and domain adaptation. Thirdly, we present the results along with their interpretation, different ablation studies as well as some manual analysis revealing insights on how the models behave in different scenarios.

2. Related work

This chapter is split into two parts: the first one highlights the most recent advancements in natural language processing (NLP) and machine reading comprehension (MRC) while the second part focuses on some specific QA systems that have been imperative when introduced.

2.1 Recent NLP advancements

Recurrent neural networks (RNNs), with or without an attached attention mechanism, used to be the de facto standard for many natural language deep learning architectures across different problems: question answering [13], neural machine translation [14, 15], coreference resolution [16], textual entailment [17] or sentiment analysis [18]. Despite this fact, recurrent neural networks are governed by a set of imperative problems. One of the main issues with RNNs is the phenomena of vanishing and exploding gradients [19] that make training impossible, tricky, or extremely difficult. The problem can be partially solved by using gated units as in the case of LSTMs [20] or GRUs [21]. However, even with the forgetting mechanism described by the LSTM equations, RNNs fail to capture long dependencies (in practice, we can expect LSTMs to learn dependencies of about 35 - 40 words [14]). Another problem with RNNs is their intrinsic sequential architecture that makes training very time-consuming and difficult to parallelize. Due to these core limitations, researchers have tried to move away from using LSTMs (and RNNs in general). During 2017, a revolution had begun having massive impacts on the natural language understanding field. The paper "Attention Is All You Need" [22] proposed a new paradigm that entirely replaced the recurrence mechanism with an innovative transformer architecture that only relies on self-attention layers to learn dependencies and compute representations of the input data. Self-attention (occasionally referred to as intra-attention) was, however, proposed before the transformer itself [23, 24]. The novelty is that the Transformer was the first architecture to use self-attention in an end-to-end manner, without any RNNs or convolutions computing (intermediate) sequence representations. Until the transformer was proposed [22], most of the attention mechanisms (including self-attention) were placed on the top of the hidden states of some RNN layer(s). Regarding our work, the most important part of the transformer architecture is the encoder, therefore, we are mostly referring to self-attention in the context of the encoder layers. Before any further digging into the technical details of self-attention, a good intuition of what it does can be given by referring to the process of a human reading a single paragraph. Local spans from the paragraph are related to one another and a human could potentially segment the paragraph into tokens (or groups) and compute some internal representations that they consider useful. Those representations are likely to feature the dependencies between the tokens as well as some semantic aspects. Notice that this process depends only on the paragraph itself and not on any other inputs – that is why the mechanism is called self-attention (there is only one input). Notice that after reading the paragraph, the resulted brain-internal representation can be used to solve some tasks (e.g. a question is asked based on the information in the paragraph). In contrast with self-attention, other types of attention mechanisms have been frequently proposed, involving two or more inputs between which attention scores are computed [14, 25]. For example, in the case of visual question answering (VQA) some proposed architectures [25] deploy an attention mechanism between the image and the question with the objective of highlighting relevant parts in the image given the question at hand. Another good intuition behind self-attention can be offered by arguing about the coreference resolution task. We used the Transformer architecture trained to translate sentences from English to German (link to the Colab notebook) and fed into it the following sentence "The ball didn't fit into the bag as it was too big.". We are not interested in the translated text but in the encoder's self-attention scores (e.g. input to input) - see Figure 1.

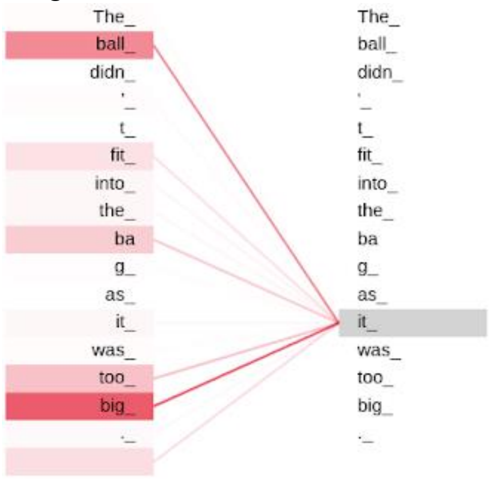


Figure 1. Self-attention scores in the 4th encoder layer (some attention head). Notice that the pronoun "it" is associated with the tokens "ball", "too" and "big".

The sentence above is quite tricky to parse as it can lead to an ambiguity if the model is not "smart" enough: the pronoun "it" can, from a syntactic point of view, reference the noun "bag" but this situation does not make sense from a semantic point of view. Choosing the right alternative requires some commonsense knowledge to associate the pronoun "it" with the noun "ball" (e.g. the ball is too big to fit into the bag, not vice-versa).

Surprisingly enough, the mathematical formalism behind self-attention is almost trivial. The input of a self-attention layer is a set of vectors (initial token representations/encodings). For each encoding, three additional vectors are created: a query vector (q), a key vector (k), and a value vector (v). Those vectors are obtained by multiplying the encoding with three matrices (learned during the training process). The next step is to obtain the similarity between the token at timestamp i with every other token at timestamp j (including $i = j$). This is done by taking the dot product between the query of token i and key of token j (that is, $q_i \cdot k_j$) and dividing by a constant (this detail is not important for understanding the mechanism but has been reported to help the training process - more details are given in the following paragraphs). A softmax is applied to map the scores to a probability distribution (the attention scores for the token i at the current layer). To obtain the representation for the token i at the next layer, a weighted sum of the computed value vectors (v) is considered, adjusted by the attention scores. The authors performed an optimization trick that they specify help training: each dot product value is divided by a well-chosen constant such that to normalize the values into a zone with more stable gradients (the softmax derivative quickly approach zero for large values). Furthermore, in the original paper, the authors reported having used multiple self-attention heads with the purpose of capturing different relations between the input tokens. What this means is that multiple sets of the three matrices (Q, V, K) are used at each layer in the encoder. The matrices are randomly initialized thus converging towards different (q, k, v) representations. The authors of the paper mentioned using 8 different attention heads in their experiments. Some skip connections are used at each self-attention layer to facilitate training (inspired by the residual connections in computer vision [26]) followed by layer normalization [27]. What we have just described is a layer of the encoder part in the Transformer architecture [22] (Figure 2). Multiple encoder layers are stacked to produce deeper contextualized representations of the input sentence (6 layers were used in the original implementation [22]).

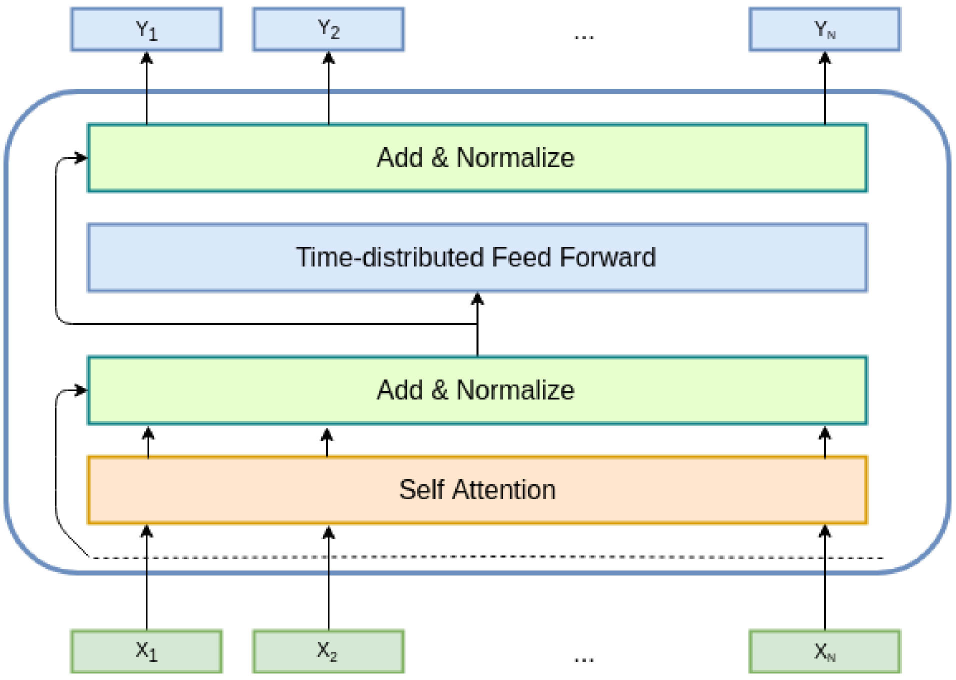


Figure 2. A transformer encoding layer with self-attention. Adapted from [22].

For the question answering task, as we are going to see in the next chapters, the interesting part of the transformer is the encoder. The decoding component is useful for sequence-to-sequence (seq2seq) tasks, such as neural machine translation. When the Transformer architecture has been proposed, it outperformed (by a large margin) any attention-based RNN models on a couple of translation tasks (English-to-German and English-to-French). Furthermore, training the transformer is computationally cheaper compared to other state-of-the-art (SOTA) models at that time [29, 30]. Still, one of the main disadvantages of the transformer architecture is the tricky and difficult training process. A large amount of data is needed and a special learning rate scheduler has been used to train the model for machine translation [22]: in the first 4000 steps (called warmup steps) the learning rate is increased linearly and, after that point, the learning rate decreases inversely proportional to the square root of the update step number. Vaswani et al. [22] provided no satisfactory explanation behind this strategy. We guess that it was inspired by the temperature-based gradient descent optimizers and simulated annealing [28]. Another main drawback of the vanilla transformer, in the context of language modeling [31], is the fixed upper bound on the attention span. This implies that the transformer is not able to learn dependencies longer than the maximum sequence length it is trained on (usually 512 tokens). This problem is usually referred to as "context fragmentation" [32]. Dai et al. [32] have proposed a novel modification to the vanilla transformer, which addresses the problem of fixed length contexts using two ideas: first, relative positional embeddings are used and then a recurrence mechanism is introduced in order to pass on information from the current segment to the following ones. The resulted architecture has been named Transformer-XL [32]. The recurrence mechanism is analogous to RNNs but there are some differences. When processing the segment $i + 1$, the self-attention "sees" the hidden states from the segment i (this happens during the forward pass). On the other hand, when backpropagating, the errors do not flow from the segment $i + 1$ to the segment i. The Transformer-XL architecture has improved the vanilla transformer on multiple language modeling benchmarks [32] showing that learning long dependencies is, indeed, important.

Since the enormous success of the transformer architecture, some research work has been directed towards pre-training transformers for language modeling. **BERT** (a shorthand for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers) [33], was introduced in late 2018 and employs a transformer architecture to compute bidirectional text representations (in contrast with previous methods, such as GPT [34], which are just unidirectional language models). Devlin et al. [33] have argued that bidirectional representations are essential for fine-tuning language models on challenging tasks such as natural language inference or question answering (e.g. downstream tasks). Although the idea of language modeling pre-training was not introduced by BERT (see for example [34, 35, 36]), their contribution is that of generating bidirectional representations and showing that they can improve the SOTA performance on a wide range of NLP tasks. The BERT framework is composed of two steps:

a) A transformer is pre-trained on unlabeled data with a language modeling objective;
b) The resulting model is fine-tuned on supervised data from a downstream task.

BERT is pre-trained on two tasks using the entire English Wikipedia and other raw text resources. The first task is called "masked language modeling" (MLM) in which 15% of the input tokens are removed (e.g. masked) and the model is trained to predict the missing tokens. The second task, called "next sentence prediction" (NSP), asks the model to say if a sentence may follow another sentence in a text or the latter comes from a different topic/article (this is not natural language entailment). Notice that the MLM task is what allows BERT to learn bidirectional representations since it can consider both left and right words in order to predict a masked token. BERT comes in two flavors: a smaller version with 110 million parameters (BERT base) and a huge network with 340 million parameters - BERT large.

Following the introduction of BERT in 2018 [33], multiple improvements have been proposed, but the general idea remained the same: unsupervised or semi-supervised pre-training with a language modeling objective, followed by fine-tuning on a downstream task. In the next sections, we are going to describe the most important BERT improvements.

Yang et al. [37] have identified two important problems with BERT that are caused by the way the MLM task is implemented. The first one highlights the fact that the "[MASK]" token used in the pre-training stage does not appear during fine-tuning, leading to a phenomenon Yang et al. [37] entitled "pretrain-finetune discrepancy", which may limit the fine-tuning performance. The second problem is that BERT predicts multiple tokens in parallel and it assumes that all the masked tokens can be approximated based only on the visible words. However, the missing tokens themselves may be dependent on each other. To solve those issues, Yang et al. [37] proposed a new model called **XLNet**. XLNet is not pre-trained with the MLM task in mind but on a novel permutation language modeling (PLM) objective. In the left-to-right conditional generation, the next word in a sentence is predicted based on the last words (or backwards, but the same sequential order is maintained). However, this leads to unidirectional representations which is exactly what BERT improved on. Instead of predicting tokens from left to right, XLNet generates tokens based on a random permutation of the original tokens in the sentence. For example, consider the sentence "The fridge is full of food". In a traditional sequential language modeling task, the tokens would be predicted in this exact order: "the", "fridge", "is", "full", "of", "food" (left to right). However, XLNet may generate words as such: "fridge", "full", "of", "the", "food", "is" (e.g. in random order). In this specific case, "full" would be conditioned on "fridge", "of" would be conditioned on "fridge" and "full", etc. This strategy allows XLNet to learn bidirectional representations while getting rid of the artifacts caused by the MLM task. Furthermore, XLNet replaces the vanilla transformer with the Transformer-XL [32] allowing longer dependencies to be captured.

RoBERTa [38] improves BERT by changing some key hyperparameters leading to significant progress in different natural language downstream tasks. Liu et al. [38] proposed the following modifications to the BERT approach:

a) Removing the NSP objective - experiments showed that NSP actually hurts the fine-tuning performance, although Devlin et al. [33] hypothesized that NSP is valuable for downstream tasks such as natural language inference. As a result, the NSP task has been dropped from the RoBERTa's pre-training stage;
b) RoBERTa still uses the MSM task but with a slight change: instead of choosing the masked tokens only once, before training, RoBERTa adopts a dynamic approach in which tokens are randomly removed every time a sequence is fed into the model;
c) Increasing the pre-training batch size from 256 to 8K. This change not only boosts the training efficiency because of the intrinsic data parallelization but also improves the perplexity of the resulting language model. Following the increase in the batch size, the learning rate has been enlarged appropriately [39];
d) Pre-training on a lot more data - RoBERTa uses a corpus of about 160GB of raw text collected from the English Wikipedia and other collections such as the CommonCrawl News dataset. In contrast, BERT was pre-trained on approximately 16GB of data. The 10x increase is expected to improve the end-to-end performance since the model is exposed to more pieces of knowledge during its pre-training.

RoBERTa is essentially a better configuration of BERT (hence, its name: **R**obustly **o**ptimized **BERT** **a**pproach) and archives SOTA results on a wide range of downstream NLP tasks (Table 1).

ALBERT (**A** **L**ite **BERT**) [40] focuses on reducing the size of the language model without hurting the fine-tuning performance on the downstream tasks. Lan et al. [40] proposed two parameter-reduction techniques the most notable one being the cross-layer parameter sharing method. This enables the self-attention and the feed-forward network weights to be reused in sequential encoder blocks. As a consequence, the number of parameters does not grow proportionally with the number of layers in the transformer. Furthermore, ALBERT disconnects the size of the embeddings from the size of the hidden space allowing an additional reduction in the number of parameters. Finally, ALBERT replaces the NSP task with a sentence order prediction (SOP) objective - the BERT's MSM is still used. The SOP task requires the model to predict if two sequences are in order or swapped. This task is potentially more challenging than NSP as it focuses more on text coherence rather than topic prediction (the negative samples in the NSP task are usually semantically distant as they may come from different paragraphs or even articles). ALBERT (the XLarge flavor) improves or matches the BERT's performance on multiple downstream datasets while contracting the number of parameters from 334M to 60M (please see Table 1, below).

The **T**ext-**t**o-**T**ext **T**ransfer **T**ransformer (**T5**) [41] introduces a unified framework in which all NLP tasks are cast into a seq2seq format. This is achieved by prefixing the inputs with a string describing the task at hand. For example, in the case of English to German translation the input has the following format: "translate English to German: <text to be translated>". As a result, T5 can be pre-trained on a wide range of NLP tasks by converting them to the above format. T5 uses the vanilla transformer [22] with a slight change to the positional embeddings [41] and it was pre-trained on a huge amount of data (about 750GB of English text). The largest T5 model has about 11 billion trainable parameters and has reached a state-of-the-art (SOTA) performance on a bunch of NLP datasets (Table 1).

Table 1

Results obtained by different transformer architectures on the GLUE [42] benchmark

Model	**MNLI**	**QNLI**	**RTE**	**CoLA**	**STS**
BERT Large	86.6	92.3	79.4	60.6	90.0
XLNet Large	89.8	93.9	83.8	63.6	91.8
RoBERTa Large	90.2	94.7	86.6	68.0	92.4
ALBERT XXLarge	90.8	95.3	89.2	69.1	**93.0**
T5 11B	**91.7**	**96.7**	**92.5**	**71.6**	92.5

To conclude this chapter, we have presented some interesting improvements and iterations on the ideas proposed by BERT, however, the general framework is the same: language modeling pre-training followed by fine-tuning on various downstream datasets. This approach is standard nowadays and has produced SOTA models on multiple NLP datasets (in some cases, by a wide margin compared to pre-transformer models).

2.2 Question answering specific models

This section describes some interesting models that played a pivotal role in improving the QA performance over the years. In the context of closed-book question answering, we can identify models using either structured information (e.g. knowledge graphs, ontologies) or knowledge encoded in raw text corpora (such as Wikipedia). Inference engines are diverse as well: neural networks, SVMs [43], rule-based systems [44], Markov Logic Networks [45] or simple Pointwise Mutual Information (PMI) solvers [1, 44].

Zhang et al. [46] have proposed a QA system (called KG^2) based on structured knowledge and graph embeddings [47]. KG^2 is designed to answer multiple-choice questions in a closed-book format and has been one of the most successful QA models with a structured knowledge component under-the-hood. The proposed approach has the following steps:

a) Hypotheses are generated based on the question and the candidate answers (one hypothesis for each answer choice). This is basically translating a pair *(question, possible answer)* into an affirmative sentence. For example, the question "Which is the tallest building in the world?" along with the candidate answer "Burj Khalifa" would generate the hypothesis "Burj Khalifa is the tallest building in the world". This step transforms the QA task into establishing which hypothesis is true;
b) A supporting context is retrieved that best matches each hypothesis (textual similarity). The context is composed of 20 sentences [46] and has the role of validating the hypothesis.
c) Knowledge graphs are constructed based on the hypothesis and the corresponding supporting context. Relation triples in the format (*subject, predicate, object*) are extracted using Open IE [48] and aggregated into a knowledge graph by coupling *subjects* with *predicates* from other relations (e.g. aligning nodes by identifying similar entities). The graph edges are labeled with relations such as "object", "time" and "location" – all words are lemmatized to help to remove some noise and to build a more robust knowledge graph. In Figure 3 we provide an example with two knowledge graphs generated from some hypothesis and its supporting context.

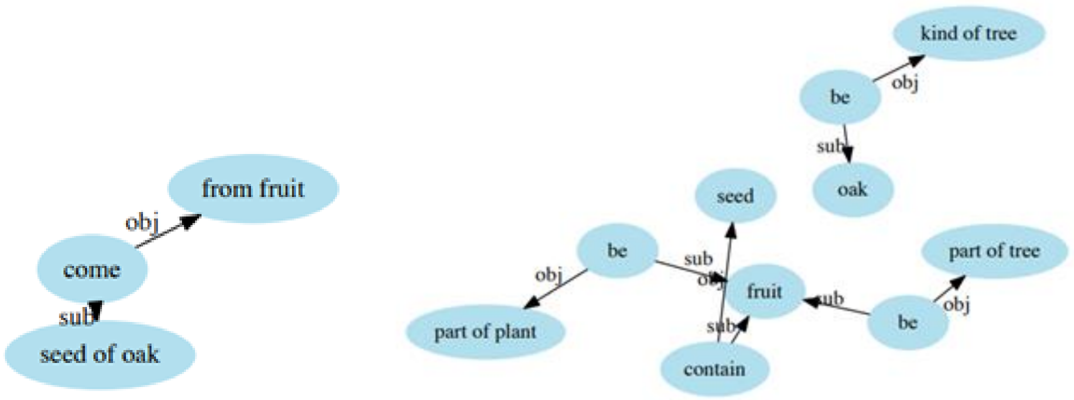


Figure 3. Knowledge graphs generated by KG^2 (source: Zhang et al. [46])

d) A graph scoring function ($f\colon G \ x \ G \rightarrow \mathbb{R}$) is leaned to assign high scores to pairs of graphs that represent correct answers and low scores to pairs of graphs that represent wrong answers (G is the set of the knowledge graphs). Graph embeddings [49, 50] are employed to encode each knowledge graph into a set of vectors, one for each node, capturing the local information to that node (those embeddings are functions of the current node and its neighbors). The inner product values between the embeddings dictate the graph scoring function and, ultimately, the correct answer.

The method has a couple of weak points that affect its end-to-end performance. Firstly, converting sentences to knowledge graphs is not perfect due to failures in Open IE. The node matching phase is also perturbed by noise in the text (such as words with multiple textual forms). Secondly, the graph scoring function may not be able to "simulate" complex reasoning (as identified by the authors [46]) KG^2 has been tested on the ARC Challenge dataset [1] achieving 31.70% accuracy. Zhang et al. [46] empirically measured the effects of the weak components by looking at a subset of 100 questions: insufficient support (51% of the total questions), failures with Open IE (12%), requires very complex reasoning (21%), still learnable with KG^2 if optimizing the framework (15%), others (1%).

One of the biggest problems for question answering systems is the lack of common-sense knowledge. Ideally, with enough training data to cover all the possible pieces of knowledge in the world and a way to guarantee that models can be trained to capture that knowledge and properly use it at inference time, the need for injecting external common-sense knowledge would not exist. However, this is not the case and researches tried to come up with ideas of including common-sense knowledge into QA systems using available knowledge bases such as ConceptNet [51, 52, 53] (see Figure 4).

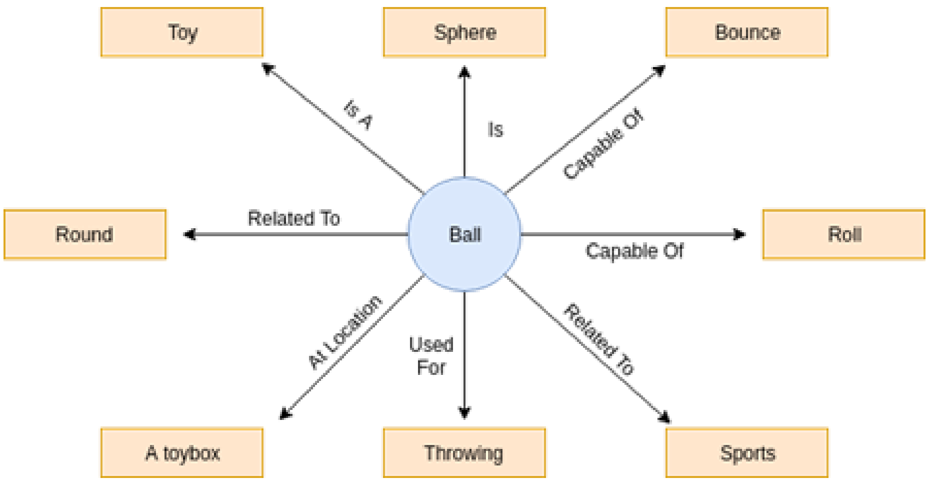


Figure 4. ConceptNet common-sense knowledge for the concept "ball"

Zhong et al. [2] have proposed a question answering model that incorporates commonsense knowledge from ConceptNet. Their approach is generic enough so that it can be applied to any other knowledge graph. First of all, the multiple-choice QA task is translated into a binary QA task in which each candidate answer is assigned a score such that the answer with the largest score should be the correct one. The scoring function, $f(q, a_i)$, is a weighted sum of two scores:

a) A document-based scoring function which verifies the given answer based on information from an external passage. TriAN [54] is used as an attention mechanism between the context, the answer, and the question. We will not dig into the details of TriAN here, as it is not particularly important for blending common sense knowledge into QA systems. The reader is recommended to refer to Wang et al. [54];
b) A commonsense-based scoring function pre-trained to understand semantic associations between concepts. ConceptNet facts *(subject, relation,* object) are extracted by matching n-grams from questions and candidate answers to nodes in the knowledge graph. Thus, a set of relevant concepts is retrieved and further handled to train a model to distinguish between correct and incorrect answers. Concepts are embedded in dense vectors using an algorithm inspired by graph neural networks [55]. It considers both direct and indirect relations between concepts as well as the concept text (it can be multiple words) encoded using GloVe [56] and a bidirectional LSTM [20]. A direct relation means that there is a link in the knowledge graph between the concepts, whereas an indirect relation means that there is a third pivot concept that links the two initial concepts (a triangle in the graph);

The inner product between question concepts and answer concepts is used to assign a score, directly proportional to the probability that the candidate answer is correct. Zhong et al. [2] have shown that both direct and indirect relations lead to an increase in the end-to-end QA performance (Table 2).

Table 2

Results reported by Zhong et al. [2] on the ARC Challenge dataset [1]

Model	**QA Accuracy**
TriAN (document-based scoring function only)	31.25%
TriAN + direct relations (between concepts)	32.28%
TriAN + indirect relations	32.96%
TriAN + both direct and indirect relations	33.39%

Another important aspect of QA systems is how to identify and manipulate essential terms in questions. This is especially significant for closed-book QA which requires external information to be retrieved. However, identifying essential words can be useful in the inference stage as well, allowing models to attend on “important” terms. Ni et al. [57] proposed a retriever-reader model (ET-RR) that learns to make use of the essential term information in both the retrieval and the inference stage. The essential term selector is a neural network trained on labeled data collected by Khashabi et al. [58] (terms in questions are labeled with an essential score). ET-RR analyzes all candidate answers at a time instead of assigning a score for each choice and then selecting the one with the highest score. This is important to discriminate between different choices for questions that require some relative ordering of the candidate answers or for “none of the above” type questions.

Sun et al. [59] have introduced a set of reading strategies aimed to improve MRC performance. A pre-trained language model is fine-tuned on downstream QA datasets using multiple approaches (called reading strategies):

a) *Back and forth reading*: consulting the original order of the inputs and the reverse order;
b) *Highlighting*: which tries to simulate the highlights used by humans when reading;
c) *Self-assessment*: generate “practice” questions on a document and use those to first fine-tune the transformer before further fine-tuning on the downstream QA task itself.

The use of the above reading strategies improves the model MRC performance compared to the same language model fine-tuned directly on the downstream task [59]. UnifiedQA [60] is another approach manipulating pre-trained language models, achieving impressive results on a multitude of datasets targeting multiple-choice or extractive question answering. Khashabi et al. [60] proposed a unified QA format which allows the same pre-trained model to be fine-tuned on a mixture of question answering datasets. The emerged model can be fine-tuned, again, on other downstream QA tasks.

Under the closed-book QA circumstances, a common approach for incorporating external knowledge is to extract passages from sources such as Wikipedia using classical information retrieval techniques that employ a token-based scoring function to denote textual similarity (examples: standard TF-IDF, Okapi BM25). However, it has been previously reported that such a retrieval engine is often not able to extract relevant documents from the reference corpora [46, 61]. Depending on what closed-book QA dataset is targeted, the fraction of questions having insufficient and irrelevant support can even reach 50% [46]. Nonetheless, the aforementioned approach has its advantages such as the ability to search huge amounts of unstructured text (TB of data) in an efficient and scalable way. To improve the quality of the retrieved passages, a typical approach is to re-rank the top N results using a more advanced method (e.g. semantic re-ranking). The latter does not apply to the entire corpora because it is very slow and does not scale well. Pîrtoacă et al. [62] have proposed a novel approach of jointly answering (multiple-choice) questions and semantically (re-)ranking relevant supporting documents. A set of discriminators have been trained to differentiate between relevant and “noisy” passages, for a fixed question:

a) A *document relevance discriminator* (DRD) which considers a pair (*question, document*) and examines if there is enough information in the document to answer the question at hand. This discriminator is pre-trained on a slightly modified version of the SQuAD 2.0 dataset [63] where "not answerable" questions are assumed to lack a correct answer because the document is not pertinent;
b) An *answer verified discriminator* (AVD) which is (pre-)trained to inspect if a candidate answer is indeed the correct answer to a question, given some external document. The RACE dataset (Reading Comprehension Dataset) [66] is used to pre-train the AVD discriminator.

The discriminators are trained on separate datasets, independent of the main QA task. They are used for inference on a third dataset (authors have used ARC [1] as the target QA dataset). BERT Large [33] has been the underlying model for both discriminators. The original paper [62] described a combiner network based on a self-attention mechanism to rank the documents and, at the same time, assign a score to each candidate answer (a high score means that the answer is correct). The end-to-end model has been named "Attentive Ranker" and reached a state-of-the-art accuracy on the ARC dataset [1] when published in April 2019.

3. A robustly optimized Attentive Ranker

This chapter is dedicated towards improving the Attentive Ranker model, described at the end of the "Related work" section. We hypothesize that the effectiveness of both discriminators (DRD and AVD) can be highly improved by employing the latest advancements in language modeling. The next sections describe different experiments and insertions brought to the original Attentive Ranker model.

3.1 Improving the Document Relevance Discriminator

The best DRD model, until this work, has been a fine-tuned BERT architecture on the modified SQuAD 2.0 dataset (as described in [62])). Around 150k pairs in the format (*question, document*) are employed to train the discriminator to predict whether the document is relevant or not in answering the question. We fine-tuned more advanced transformer architectures with the help of the Google Cloud TPUs Research Program. We have been awarded 9 months of free TPU usage that has been crucial to our research.

3.1.1 XLNet Discriminator

We have used the Large version of XLNet [37] pre-trained as mentioned in the model's paper. The checkpoints are used to fine-tune the model on our downstream task – document relevance. Deep transformer architectures are known to be sensitive to hyperparameter selection. Therefore, we have tried different hyperparameter configurations and select the best one on the validation dataset. In our experiments, we started from a base (default) hyperparameter set and then adjusted some values empirically, reporting the performance each time a modification has been made. Table 3 presents the default set of hyperparameters for XLNet and other language models (from the following chapters).

For XLNet we have conducted the experiments highlighted in Table 4. Model #4 (which increases the learning rate warmup steps) has the best test accuracy and it is going to be used for inference on the ARC dataset. We would like to point out that although the accuracy difference between experiment #1 and experiment #4 is negligible, the evaluation loss obtained by #4 is much lower (0.5295 compared to 0.5969). Therefore, we have reasons to believe that the model trained in experiment #4 has a better generalization capacity.

Table 3

Default hyperparameters for fine-tuning document ranking models

Hyperparameter name	**XLNet**	**RoBERTa**	**ALBERT**
Dropout	0.1	0.1	0.1
Attention dropout	0.1	0.1	0.1
Max sequence length	512	512	425
Batch size	48	32	32
Learning rate	3e-5	2e-5	5e-5
Learning rate decay	Linear	Linear	None
Weights regularization	0.01	0.01	None
Adam epsilon	1e-6	1e-8	1e-6
Layer-wise learning rate decay	0.75	1.0 (None)	1.0 (None)
Warmup steps	800	6% of training	10% of training
Uncased text preprocessing	False	False	True

Table 4

Results for XLNet as the document relevance discriminator

#	**Hyperparameter set**	**Test accuracy (random 50%)**
1	Default (as in Table 3)	88.60%
2	Delta (from #1): Batch size decreased to 24	88.14%
3	Delta (from #1): Layer-wise learning rate decay = 1	87.91%
4	Delta (from #1): Warmup steps = 2000	88.67%

3.1.2 RoBERTa Discriminator

We have used the pre-trained large version of RoBERTa that has about 355M parameters. The default fine-tuning hyperparameters are the ones recommended by Liu et al. [38] for the SQuAD 2.0 dataset. However, with have made some small modifications: batch size 32 instead of 48, learning rate 2e-5 instead of 1.5e-5 (see Table 3). Table 5 lists the obtained results.

Table 5

Results for RoBERTa as the document relevance discriminator

#	**Hyperparameter set**	**Test accuracy (random 50%)**
1	Default (as in Table 3)	89.76%
2	Delta (from #1): Batch size = 48, LR = 1e-5, RAdam	89.75%
3	Delta (from #2): LR = 5e-5, No regularization	88.22%

Notice that increasing the learning rate by too much (1.5e-5 to 5e-5) leads to optimization overshooting during training and the resulting model has considerably poorer accuracy. Model #1 is chosen as the final RoBERTa discriminator.

3.1.3 ALBERT Discriminator

For the ALBERT XXLarge model [40], the default hyperparameter set that we used is a mixture of the recommended values from the ALBERT paper [40] and some values that we observed to work better during RoBERTa and XLNet fine-tuning (Tables 3, 4, 5). In experiment #2 we have used the exact parameters as in the ALBERT paper (see appendices from Lan et al. [40]). The experiment results are shown in Table 6. The reader should notice that the best set of hyperparameters for one transformer model (e.g. RoBERTa) may not be the best choice for another architecture (e.g. ALBERT). We observe that decreasing the learning rate from 5e-5 to 3e-5 leads to an increase in performance. This is similar to the effects of the learning rate that we noticed in the RoBERTa experiments.

Table 6

Results for ALBERT as the document relevance discriminator

#	Hyperparameter set	Test accuracy (random 50%)
1	Default (as in Table 3)	91.17%
2	Delta: Max Seq. Length = 512, Batch size = 48, Learning rate = 3e-5	91.81%

Table 7

Document relevance performance for different models

Model	SQuAD 2.0 Test Accuracy	Delta (from previous)
Random	50.00%	N/A
BERT Large (previous best)	80.43%	+30.43%
XLNet Large	88.67%	+8.24%
RoBERTa Large	89.76%	+1.09%
ALBERT XXLarge	91.81%	+2.05%

3.1.4 Final observations

Table 7, above, collects the results of various models for the document relevance task. ALBERT XXLarge has by far the best performance as a DRD (RoBERTa being the second-best). However, in the end-to-end model, we are going to use an ensemble with all the models hoping that some may cover the inaccuracies of others, boosting the final performance. In this context, it is useful to look at Cohen's Kappa coefficient between the model predictions. A value close to 1 means perfect agreement.

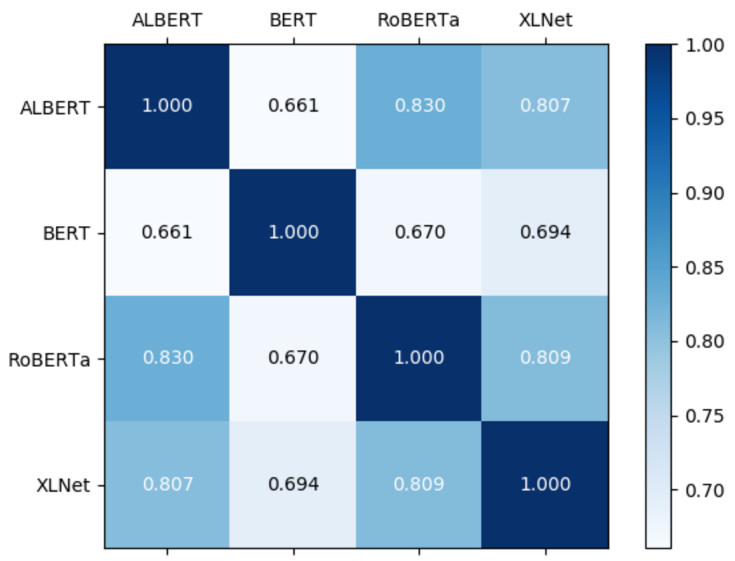


Figure 5. Cohen's kappa coefficient between predictions of DRD models

From an ensemble perspective, we don't want an agreement coefficient too close to 1 since this would mean the models make the same predictions almost all the time. On the other hand, an agreement coefficient that is too low (< 0.6) means that the models are contradictory. Again, this is not a desirable outcome. Values close to 0.8 is what we are aiming at. For example, the agreement between ALBERT and XLNet is 0.807 which means that they can work well in an ensemble, boosting each other's performance.

3.2 Improving the Answer Verifier Discriminator

The best AVD model until this work has been a fine-tuned BERT Large architecture on the RACE dataset [66]. All the models trained on the AVD task receive a tuple (*question, candidate answer, supporting document*) and should decide if the candidate answer can be inferred as correct given the information from the supporting document. The candidate answer with the highest score assigned by the model (0 means wrong, 1 means predicted as correct) is chosen as the final prediction. More advanced transformer architectures have been fine-tuned to improve the previous AVD discriminator (e.g. a fine-tuned BERT Large). We report 4-way accuracies on the RACE validation and test datasets.

3.2.1 XLNet Discriminator

In a manner similar to the DRD training procedure (explained in Chapter 3.1), we have tried different hyperparameter configurations and selected the one with the best performance on the RACE test split. To conduct our experiments, we started from a base hyperparameter set and empirically adjusted some values, reporting the performance each time a modification has been made. Table 8 below presents the default set of hyperparameters for XLNet and other models.

Table 8

Default hyperparameters for different answer verifier (AVD) models

Hyperparameter name	XLNet	RoBERTa	ALBERT
Dropout	0.1	0.1	0.1
Attention dropout	0.1	0.1	0.1
Max sequence length	635 * 4	512	512
Batch size	32	16	32
Learning rate	2e-5	1e-5	2e-5
Learning rate decay	Linear	Linear	None
Weights regularization	0.01	0.1	None
Adam epsilon	1e-6	1e-6	1e-6
Layer-wise learning rate decay	1.0	1.0 (None)	1.0 (None)
Warmup steps	1000	10% of training	10% of training
Uncased text preprocessing	False	False	True

Table 9

Results for XLNet as the answer verifier discriminator

#	Hyperparameter set	Val accuracy (random 25%)
1	Default (as in Table 8)	80.78%
2	Delta (from #1): LR = 4e-5, LR decay rate = 0.965	74.83%
3	Delta (from #1): LR = 1e-5, Warmup Steps = 2000	80.45%

The XLNet Large network fails to train successfully if fed one candidate answer at a time with the objective of predicting if the answer is correct or not, even though we have tried different sets of hyperparameters and different runs (to encourage different random initialization of the parameters in the final feed-forward layer). Each time, the network had gotten stuck in a local minimum in which the binary accuracy is 75% (always choosing to say false). If we try to balance the dataset artificially (by over-sampling the number of positive examples) then the network is again stuck at 50% accuracy and the optimizer fails to reduce the loss. To overcome this issue, we decided to feed all the answers to the model input. This way, the model can consider all the candidate answers before making a prediction. The objective is now to predict the correct answer out of the 4 possibilities. As a result, the categorical cross-entropy is used. The default set of hyperparameters for XLNet is stated in Table 8. Notice the high value for the max sequence length which is due to the concatenation of the answers in the input tokens. Also, notice that this is possible specifically due to the Transformer-XL architecture being used inside the XLNet model, allowing large contexts in the transformer (please refer to Chapter 2.1 for an in-detail explanation of how Transformer-XL has been designed). The performance of XLNet is reported in Table 9, above.

3.2.2 RoBERTa Discriminator

We have used the pre-trained large version of RoBERTa that has about 355M parameters. In contrast to XLNet, RoBERTa has been trained with one candidate answer at-a-time (results in Table 10), as described in Chapter 3.2.

Table 10

Results for RoBERTa as the answer verifier discriminator

#	Hyperparameter set	Val accuracy (random 25%)
1	Default (as in Table 8)	83.52%
2	Using hyperparameters from the ALBERT paper [40] for RACE	83.14%

3.2.3 ALBERT Discriminator

Finally, ALBERT XXLarge has been fine-tuned on the RACE dataset, experimenting with different hyperparameters. The results are listed in Table 11, below.

Table 11

Results for ALBERT as the answer verifier discriminator

#	Hyperparameter set	Val accuracy (random 25%)
1	Default (as in Table 8)	86.90%
2	Delta (from #1): Batch Size = 48, LR = 3e-5,	86.78%

3.2.4 Final observations

We have collected the results for the all AVD discriminator models in Table 12, below. As previously stated, the RACE dataset is split into 2 categories [66]: RACE Middle containing easier questions and RACE High containing more challenging questions. We also report the performance on the combined (middle + high) test split. Notice the considerable improvement that ALBERT brings over BERT Large (previous best). This indicates that the end-to-end performance on the ARC dataset should be boosted as well (assuming transfer learning occurs).

Table 12

Answer verifier performance on the RACE test dataset

Model	RACE Test Middle	RACE Test High	Race Test Combined
Random	25.00%	25.00%	25.00%
BERT Large	72.63%	66.41%	68.22%
XLNet Large	83.84%	78.58%	80.12%
RoBERTa Large	85.66%	82.62%	83.52%
ALBERT xxlarge	88.51%	84.65%	85.77%

As previously mentioned, the AVD models are going to be used in an ensemble arrangement. In this context, it is useful to look at Cohen's Kappa coefficient between the model predictions (Figure 6). A value close to 1 means perfect agreement. Values close to 0.8 are great from an ensemble point of view (as it is the case with ALBERT and RoBERTa with an agreement of 0.722).

The next sub-chapter tries to improve the network responsible for combining all the DVD and AVD scores. The ARC dataset is used as the target QA task. We briefly re-state the idea behind the Combiner [62] and extend it to include the advanced discriminators trained in Chapters 3.1 and 3.2.

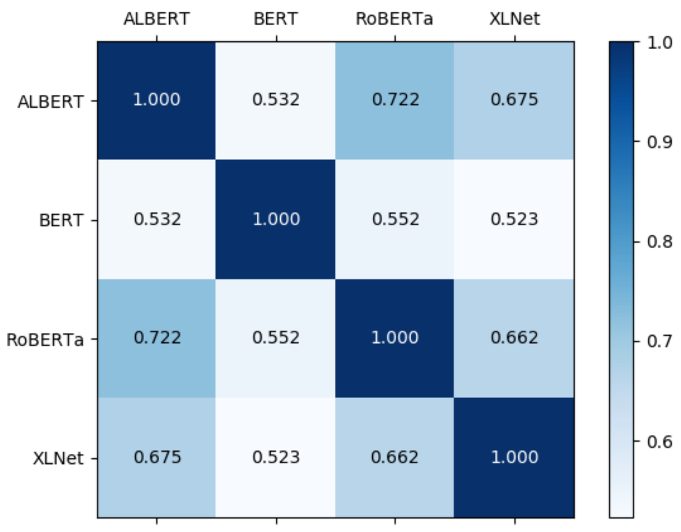


Figure 6. Cohen's kappa coefficient between predictions of AVD models

3.3 The end-to-end architecture

We turn our attention back to the ARC multi-choice QA task. External supporting documents were retrieved from Wikipedia and ARC Corpus using the procedure described by Pîrtoacă et al. [62]. From each source, top 20 documents are retrieved, thus, each candidate answer is encoded into a matrix $\mathbb{R}^{40x10}$ as follows:

- Relative position in the TF-IDF ranked list (from Lucene);
- The absolute TF-IDF score (useful for retrieval-like questions);
- 4 values from the AVD discriminators;
- 4 values from the DRD discriminators.

The combiner network is trained to predict the correct answer from the candidate list based only on the scores described above. The original Attentive Ranker model has employed a novel key-value self-attention as described by Pîrtoacă et al. [62]. In this work, we compare it with other simple approaches to demonstrate the requirement for such a component (Table 13).

Table 13

Performance obtained by different combiner architectures. Note: only BERT, XLNet, and RoBERTa have been used in this experiment (without ALBERT). We also indicate the number of trainable parameters in the combiner architecture.

Combiner Type	**ARC Easy Test Accuracy**	**ARC Challenge Test Accuracy**
The mean of the 40 scores	79.78%	55.96%
LSTM (2390 params) [20]	80.80%	57.93%
BiLSTM (2732 params) [64]	80.80%	57.25%
KV Self-Attention (4989 params) [62]	81.47%	58.88%

The KV self-attention component yields the best accuracy (Table 13). Nevertheless, it is the most complex component with almost 5k parameters (double compared to a simple LSTM). Figure 7 shows the complete architecture of the Combiner with integrated KV attention (source: Attentive Ranker poster presentation during EMNLP 2019).

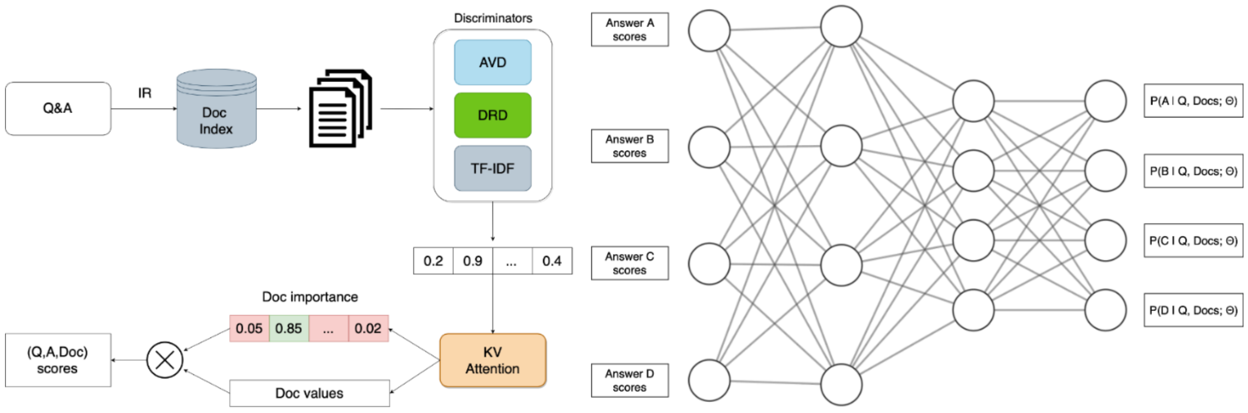


Figure 7. Combiner network architecture with key-value attention (source: Attentive Ranker poster presentation during EMNLP 2019)

3.3.1 Ablation studies

We performed some ablation studies on the AVD and DRD components to observe their contribution to the final model. In the first study (Table 14), we removed the transformer models from the ensemble (e.g. BERT, XLNet, RoBERTa, ALBERT) and in the second study (Table 15) we removed discriminator types (AVD and DRD). Notice that the best model (with all 4 fine-tuned language models in the ensemble) is better than the best single model (ALBERT) by about 2% on both ARC Easy and Challenge. Dropping any component of the system would result in a performance decrease (see Table 14 & 15). This is a solid argument in favor of keeping all the transformer-based discriminators.

Table 14

Model contribution – ablation study

Model	ARC Easy Test Accuracy	ARC Challenge Test Accuracy
Without any discriminators	64.01%	25.00%
BERT-based only	72.18%	44.71%
XLNet-based only	74.84%	54.42%
RoBERTa-based only	80.46%	56.73%
ALBERT-based only	82.87%	62.57%
All	84.69%	64.54%

Table 15

Discriminator type contribution – ablation study

Model	ARC Easy Test Accuracy	ARC Challenge Test Accuracy
Without any discriminators	64.01%	25.00%
DRD only (4 values)	70.14%	26.00%
AVD only (4 values)	82.79%	61.45%
All	84.69%	64.54%

3.3.2 Adding textual features into the combiner

Notice that the final network only combines values from the discriminators and does not take into consideration the text from the question or the candidate answers. We would like to examine if including such information would lead to a better result. A text encoding module is needed that takes a span of text (question/answer) and returns a vector (the embedding of the text in a space carrying some semantic characteristics). One such embedding can be the vector corresponding to the "[CLS]" token in the final layer from a pre-trained language model (e.g. BERT). This approach has been reported [81] to be unsuitable for semantic similarity search as well as for unsupervised tasks like clustering. The authors propose a new siamese architecture [81] that is trained to generate semantically meaningful sentence embeddings (the model is called Sentence-BERT although a variant using RoBERTa under-the-hood exists as well). We have employed the models they had made available (specifically, "RoBERTa-large-NLI-mean-tokens") to embed questions and answers into a 1024D space. However, such vectors are too high dimensional given the size of our training set. We performed dimensionality reduction using PCA [94] into a 10D space. Just for visualization, we employed t-SNE [83] to further reduce the dimensionality of the embeddings from 10D to 2D. The final question projections are displayed in Figure 8. Please observe that clusters do not form, and this is, indeed, expected. The embeddings capture the semantic meaning of the sentences (e.g. they may be clustered around categories or topics) not the difficulty in answering the question.

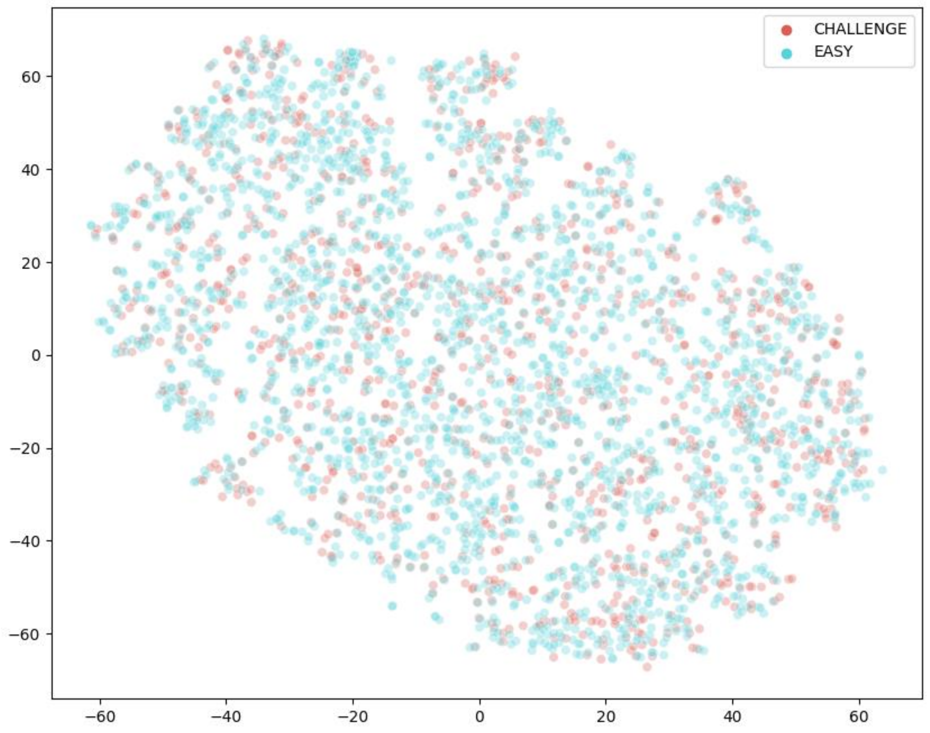


Figure 8. Question embeddings from Sentence-RoBERTa [81] projected into a 2D space using t-SNE.

The 10D question and candidate answer embeddings are fed into the Combiner network. As an additional input, we manually computed the cosine distance and the Euclidean distance between the question and each candidate answer. We manually performed this operation to "help" the neural network to learn some useful features. Experiment results are reported in Table 16. Notice that the textual information does not bring any strong improvement to the model. On the ARC Easy dataset, the performance increase is minimal and on the ARC Challenge dataset, the performance is decreasing by a little. It is important to mention that, during our experiments, we noticed a large variance in the model's performance measured on the Challenge dataset due to the small number of samples in the validation split, on which we fine-tuned hyperparameters and selected the best model.

Table 16

Adding textual features into the combiner network

Model	**ARC Easy Test Accuracy**	**ARC Challenge Test Accuracy**
TF-IDF only (no discriminators)	64.01%	25.00%
+ PCA-reduced embeddings	64.12%	27.72%
+ AVD & DRD scores	84.82%	64.12%
Without textual information	84.69%	64.54%

In this chapter, we have illustrated a robustly optimized version of the Attentive Ranker which increases its performance by 19.83% on ARC Challenge and by 12.64% on ARC Easy (Table 17). In the next section, we present some questions that are incorrectly answered by the model.

Table 17

Comparison with other state-of-the-art models

Model	**ARC Easy Test Accuracy**	**ARC Challenge Test Accuracy**
Reading Strategies [59]	68.90%	42.30%
Attentive Ranker (base)	72.18%	44.71%
QA Transfer [65][1]	76.50%	54.50%
Attentive Ranker (improved)	84.82%	64.54%
FreeLB- RoBERTa [93]	85.44%	67.75%

3.3.3 Analysis and further considerations

We extracted some examples from both ARC Easy and Challenge datasets in which the model predicts the correct answer and some in which the model is wrong. Please refer to Tables 18, 19, 20 & 21, below. The samples have been selected to be as relevant as possible.

Table 18

ARC Easy examples where the Attentive Ranker model is correct

Question: "Which technology was developed most recently?" Choice A: "cellular telephone" ✓ Choice B: "television" Choice C: "refrigerator" Choice D: "airplane" Correct Answer: "A"
Question: "An anemometer is a tool that measures" Choice A: "wind direction." Choice B: "wind speed." ✓ Choice C: "air pressure." Choice D: "air temperature." Correct Answer: "B"

[1] https://leaderboard.allenai.org/arc_easy/submission/bgkqqj923n20bdi7i2p0 (last accessed June 2020)

Table 19

ARC Easy examples where the Attentive Ranker model is wrong

Question: "After a rainfall, which process in the water cycle draws the water back up into the air?" Choice A: "condensation" ✗ Choice B: "evaporation" ✓ Choice C: "circulation" Choice D: "precipitation" Correct Answer: "B", Predicted Answer: "A"
Question: "Which type of light can cause severe eye damage if it is viewed directly?" "Choice A": "an infrared light" "Choice B": "a laser beam" ✓ "Choice C": "an incandescent light" ✗ "Choice D": "a Bunsen burner flame" Correct Answer: "B", Predicted Answer: "C"

Table 20

ARC Challenge examples where the Attentive Ranker model is correct

Question: "A human CANNOT survive the loss of which of the following?" Choice A: "The appendix" Choice B: "The liver" ✓ Choice C: "A lung" Choice D: "A kidney" Correct Answer: "B"
Question: "What part of the digestive system first causes chemical changes to food?" Choice A: "teeth in the mouth" Choice B: "saliva in the mouth" ✓ Choice C: "enzymes in the stomach" Choice D: "enzymes in the small intestine" Correct Answer: "B"

Table 21

ARC Challenge examples where the Attentive Ranker model is wrong

Question: "Which characteristic of a cheetah is more likely to be learned rather than inherited?" Choice A: "speed" ✗ Choice B: "a spotted coat" Choice C: "hunting strategies" ✓ Choice D: "claws that do not retract" Correct Answer: "C", Predicted Answer: "A"
Question: "Which of the following best describes the mass of a solid block of ice?" Choice A: "the amount of matter in the block" ✓ Choice B: "the amount of space the block takes up" ✗ Choice C: "the force of gravity acting on the block" Choice D: "the distance between the molecules in the block" Correct Answer" "A", Predicted Answer: "B"

In the next chapter, we propose a different QA model based on transfer learning that performs well in zero-shot scenarios.

4. Zero-shot QA: method description

The method proposed in this chapter is based on the Attentive Ranker, previously described, however, the objective is to investigate a pure transfer learning approach to question answering. Advanced machine reading comprehension (MRC) models are trained on a collection of datasets (hereafter referred to as base or source datasets) and the resulting models are used to predict correct answers on a target (or "downstream") dataset. The derived models are not finetuned on the downstream dataset at all, but we only rely on transfer learning to adapt the learned knowledge. This general idea is demonstrated in Figure 9, below.

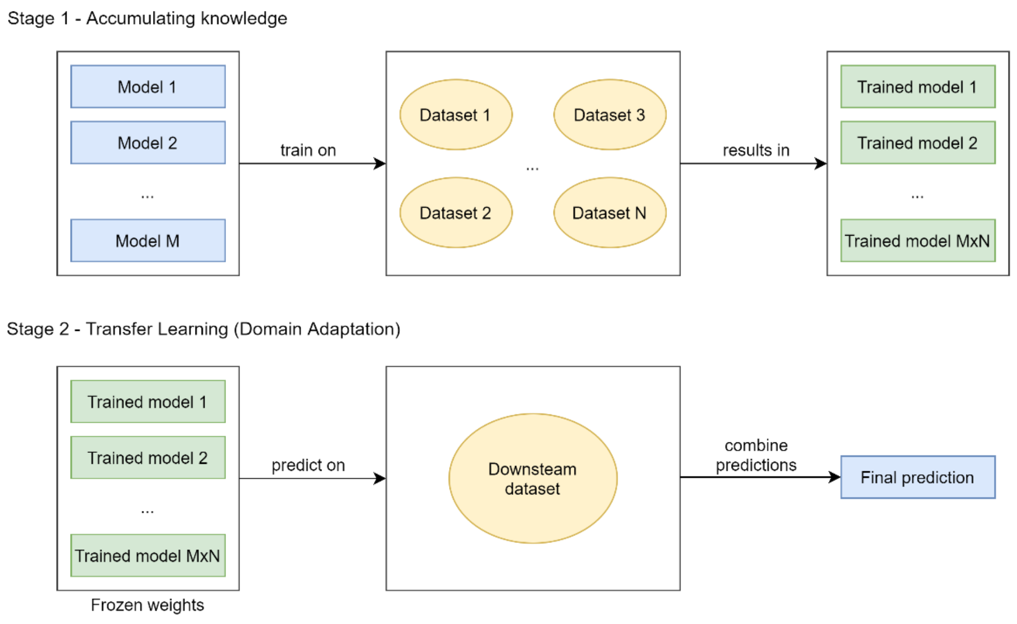


Figure 9. Transfer learning from base datasets to a downstream dataset

There are some actions in the above figure indicated with arrows:

- "train on" action: some models, possibly pre-trained (for example, on language modeling), are further fine-tuned on the base datasets;
- "results in" action: each model is trained on one dataset at a time (independently of each other) resulting in a total of (num models) x (num datasets) instances. The trained models are required to capture domain knowledge as well as commonsense knowledge to be able to answer questions from an unseen dataset (e.g. downstream) – zero-shot learning;
- "predict on" action: the set of trained models are used to predict (again independently) the correct answers to questions from the downstream dataset. The models obtained in the previous step are not trained on the downstream dataset. This step results in a set of scores;
- "combine predictions" action: a (possibly learned) function of the predicted scores is used to make the final decision. This function could consider (semantic) similarities between the downstream question and questions in the base datasets. For example, if a question is akin questions in "Dataset 1" then it should give more value to scores resulted from models fine-tuned on "Dataset 1".

The success of such a strategy resides in the ability of models to generalize on unseen questions much like in a zero-short learning scenario. To acquire this capability, models need to be trained on datasets covering a wide variety of question types and different topics. We invested a large amount of research to gather QA datasets encompassing those properties. The following sections describe both the base and the downstream datasets used in our work.

4.1 Introducing source and downstream datasets

Multiple datasets have been used in our work to train and test various models. Being familiar with the data at hand is a very important step in designing a successful system. Thus, for completeness and enhanced understanding, we shortly describe the core characteristics of each dataset employed by our work. There are five base datasets and a single downstream dataset.

4.1.1 Allen AI ARC

The AI2 Reasoning Challenge (ARC) [1] has been introduced in 2018 by the Allen Institute for Artificial Intelligence and is composed of 7787 grade-school science questions that target complex reasoning and vast commonsense knowledge. The questions are text only (no images or diagrams) and usually have 4 different candidate answers (some of them have 5, but those are ignored in our work due to some technical considerations). Moreover, no supporting context is provided in terms of a document or just some paragraphs attesting the correct answer. Therefore, most systems trying to answer ARC questions have a component that performs information retrieval on sources such as Wikipedia (for example: [62]). ARC is split into 2 sub-categories based on the question difficulty: ARC Easy and ARC Challenge. A question belongs to one category or another based on an automatic sifting process. More exactly, a question is considered to be “easy” if and only if both a retrieval and a co-occurrence model fail to predict the correct answer [1]. Table 22 shows the sizes of the train/dev/test splits in each ARC sub-dataset and as a whole.

Table 22

The total number of questions in the ARC dataset splits

Split	Easy	Challenge	Total
Train	2251	1119	3370
Development	570	299	869
Test	2376	1172	3548

The ARC authors have identified numerous types of knowledge required to answer questions in the ARC dataset [1]. We enumerate algebraic knowledge, experimental, definitions, or physics facts (but not limited to). In Table 23, we extracted two questions as relevant examples, one from the Easy partition and the other one from ARC Challenge.

Table 23

Question samples from the ARC dataset

Category	Question
Easy	What virus structure is similar in function to a cell membrane? a) protein shell b) internal protein c) tail sheath d) end plate Correct Answer: A
Challenge	An ice cube melts and then evaporates. The ice cube and the water vapor have the same a) mass b) density c) volume d) temperature Correct Answer: A

4.1.2 RACE

Large-scale **R**e**A**ding **C**omprehension Dataset from **E**xaminations (RACE) [66] is a dataset (about 100k questions) for reading comprehension that focuses on complex reasoning rather than just surface understanding of the information in some paragraphs. The RACE collection attempts to solve some critical problems in other question answering datasets: the candidate answers are often just a span from the provided context [69, 70, 71] (therefore, some word matching models could potentially predict the correct answer with decent accuracy without actually performing any reasoning), crowd-sourced questions are affected by noise or can be incorrect due to the lack of expert judgment [67, 68] and the range of covered topics is too narrow to test proper generalization in multiple scenarios. RACE has been collected from examinations of middle and high school students and contains questions generated by human experts. The sizes of RACE partitions are indicated in Table 24.

Table 24

The total number of questions in RACE dataset partitions

Split	**RACE Middle**	**RACE High**	**Total**
Train	25421	62445	87866
Development	1436	3451	4887
Test	1436	3498	4934

4.1.3 OpenBookQA

OpenBookQA (simply referred to as OBQA) is a dataset of about 6k multiple-choice science questions along with a “book” of 1329 core science facts supporting the correct answers [72]. The book is not the only information necessary to answer the questions, however. The facts are short sentences such as:

a) “Earth's tilt on its axis causes seasons to occur”;
b) “A cactus stores water”;
c) “Carbohydrates are made of sugars”;
d) “Sunlight and rain can cause a rainbow”;
e) “The moon orbits the Earth”.

OBQA questions (very often multi-hop in their style) probe the understanding of the provided facts but common-sense knowledge is also required in order to arrive at the correct answer. Solving the OpenBookQA task demands both a good information retrieval system (to extract facts from the book) and a powerful reasoning component. OpenBookQA has been collected with the help of Amazon Mechanical Turk workers with a “masters” level of qualification [72]. Table 25 indicates the sizes of different OpenBookQA partitions.

Table 25

The total number of questions in OpenBookQA splits

Split	**Number of questions**
Train	4957
Development	500
Test	500

4.1.4 CosmosQA

CosmosQA (**Co**mmon**s**ense **M**achine C**o**mprehen**s**ion) [73] is a large dataset of multiple-choice questions which targets commonsense knowledge and “reading between the lines over a diverse collection of people's everyday narratives” [73, 74]. To collect such questions, Amazon Mechanical

Turk workers were given a paragraph from a collection of personal narratives [75, 76] and asked to propose a couple of questions based on the information in the paragraph which also requires common sense knowledge to infer the correct answer. Most of the questions ask for causes/effects of events, temporal events, various facts about entities, or are just counterfactual conditionals (e.g. “What would happen if kangaroos had no tails?”) [73]. Each question has exactly one correct answer and 3 distractors. All the questions from the final CosmosQA version have been validated in a further stage, again using paid workers on Amazon Mechanical Turk. The dataset comes already partitioned into train, development, and test splits (Table 26). Table 27, below, presents some relevant CosmosQA questions.

Table 26

The total number of questions in CosmosQA partitions

Split	**Number of questions**
Train	25588
Development	3000
Test	7000

Table 27

Question examples from the CosmosQA dataset

Partition	**Question**
Train	Question: Why didn't they talk on the way to the river? Context: “We walked in silence towards the river. Our hands were joined, but the silence hissed between us. For the first time she talked about the end, and now I began to dread it.” a) There was too much tension between them. b) None of the above choices. c) They didn't feel like talking to each other. d) They are giving each other the silent treatment. Correct Answer: A
Test	Question: Which artist was their favorite of all? Context: “So yeah, what a crazy weekend. Saw the black keys, bloc party and radiohead friday. uhhh, radiohead made my weekend, completely different experience than bonnaroo, but just as good.” a) radiohead was their favorite; b) they liked all of them equally; c) Bonnaroo was their favorite; d) None of the above choices. Correct Answer: B

4.1.5 CommonsenseQA

CSQA [77] is yet another dataset built around the idea of commonsense knowledge required to answer multiple-choice questions. It has been collected using paid workers on Amazon Mechanical Turk in two consecutive stages: question generation followed by validation, resulting in about 12102 questions. The challenging nature of the CSQA questions resides in the smart usage of ConceptNet [78] from which related concepts are extracted and used to formulate distractor answer choices. With this strategy, the wrong answers to a given question are semantically adjacent to the correct one, and discriminating between the valid answer and the distractors requires various pieces of commonsense knowledge. Although similar up to some point, there is a fundamental difference between CommonsenseQA and CosmosQA. While the latter combines reading comprehension with commonsense knowledge by asking a question on a given paragraph, CSQA demands standalone sound judgment. In the case of CSQA, no supporting paragraph is provided. The sizes of the CSQA partitions are indicated in Table 28, below.

Table 28

The total number of questions in the CommonsenseQA splits

Split	**Number of questions**
Train	9741
Development	1221
Test	1140

4.1.6 DROP

Discrete **R**easoning **O**ver **P**aragraphs (DROP) [79] is a large-scale dataset with 96k questions requiring discrete reasoning such as simple additions, subtractions, comparisons, or ordering. This type of mathematical reasoning demands a more structured understanding of the information from the given context to enable operating with it. Amazon Mechanical Turk (AMT) has been used to crowdsource questions and answers, workers being provided with a paragraph extracted in advance from Wikipedia. The sizes of different DROP partitions are shown in Table 29.

Table 29

The total number of questions in DROP partitions

Split	**Number of questions**
Train	77409
Development	9536
Test	9622

DROP is not formulated in the multiple-choice setting but requires models to generate an exact answer without providing any candidate choices. In order to use DROP in a multiple-choice context, we automatically generated relevant distractor answers, by altering the correct one, based on some observations:

- There are 3 types of answers in all DROP questions: number, date, and spans;
- For number type (e.g. “4”, “10”, “102”) and date type (e.g. “January 2020”, “March”, “20 September”) categories, it is possible, and relatively easy, to tweak the correct answer and generate good distractors. The answer type itself guarantees that the distractors are wrong, and the correct answer is unambiguous and uniquely identified. For example, if “3” is the correct answer to a question, then it is clear than any other number denotes a wrong answer. The same argument holds for date type answers. However, things are not so straightforward for span answers, so we have chosen to drop them from the multiple-choice variant of DROP;
- About 66.5% of the DROP questions are of type number or date. Therefore, by rejecting the span type questions we still save a considerable proportion of the dataset;
- Generating a difficult multiple-choice DROP variant depends on the produced distractors not being trivially wrong. A completely random approach is likely to fail. Therefore, we propose some smart strategies that should generate decent distractor answers (semantically valid and plausible). In the next section, we describe the algorithms in some detail.

In order to alter an answer of type “number”, we first split the range of possible integers in multiple categories: numbers less than or equal to 20, between 20 and 99, less than 1000 and greater than or equal to 1000. The intervals have been determined empirically and usually have some different semantics associated with them. For example, values greater than or equal to 1000 are usually representing years in the DROP dataset. Distractor answers are randomly generated in the same range. For example, if the correct answer to a question is the year “2004” the generated distractors are possible, valid years close to “2004”. For real values, a similar procedure has been undergone with a further caveat that the distractors are generated to have the same number of decimal points as the correct answer. A date answer is altered by randomly changing one or more of its components: the day, the month, or the year. The source code to generate distractor answers is available as part of the DataMine[2] library and can be easily used by researchers to further investigate this idea. We provide some examples in Table 30.

Table 30

Examples of answer distractors generated from a gold answer (DROP)

Question	Gold answer	Generated distractors
How many touchdowns did Alex Smith throw?	1	[2, 0, **1**, 3]
How many less robberies were there in 2010 than in 1981	5400	[**5400**, 5404, 5427, 5367]
How many in percent weren't 18 to 24?	90.7	[96.8, **90.7**, 93.5, 88.6]
When was the last invasion by China?	1769	[**1769**, 1770, 1766, 1767]
When did Marion rescue the American force?	31 August 1781	[31 March 1781, 31 May 1781, **31 August 1781**, 31 June 1779]

4.1.7 Summary

We have described six different datasets whose characteristics cover a wide range of topics and aspects. We summarize their properties in Table 31, below. In the following chapters, all but the Allen AI ARC dataset are going to be used in an initial stage of model fitting (e.g. base datasets) with the final purpose of transfer learning [80] on the ARC questions (e.g. the downstream dataset).

Table 31

Some core properties of different question answering datasets

Dataset	Source	Supporting document?	Num. candidate answer choices	Num. questions
Allen AI ARC	grade-school examinations	no	Mostly 4	7787
RACE	middle and high school examinations	yes	4	97687
OpenBookQA	crowdsourcing	partial	4	5957
CosmosQA	crowdsourcing	yes	4	35588
CSQA	crowdsourcing	no	5	12102
DROP	crowdsourcing	yes	4 (generated)	96567

One insightful visualization is the distribution of the first three tokens in the questions – they give a lot of information on the question category. For each dataset, we parsed the question text and extracted only the interrogative sentence (the last sentence in the question, if there are multiple sentences). We performed this procedure because in some datasets (ARC primarily) the questions are frequently composed of a context part followed by the question itself (for example: “Maria is walking on a wood surface. What force is preventing Maria from slipping and falling?”). In the following section, we use the term “trigram” to refer to the sequence of three tokens at the beginning of the

[2] https://github.com/SebiSebi/DataMine

question text (the last sentence if the question is composed). Because of the inherent display limitations, we only show the top 150 trigrams (at most) in a sunburst chart (Figure 10). A zoomed-in version of each plot can be found in Appendix A.

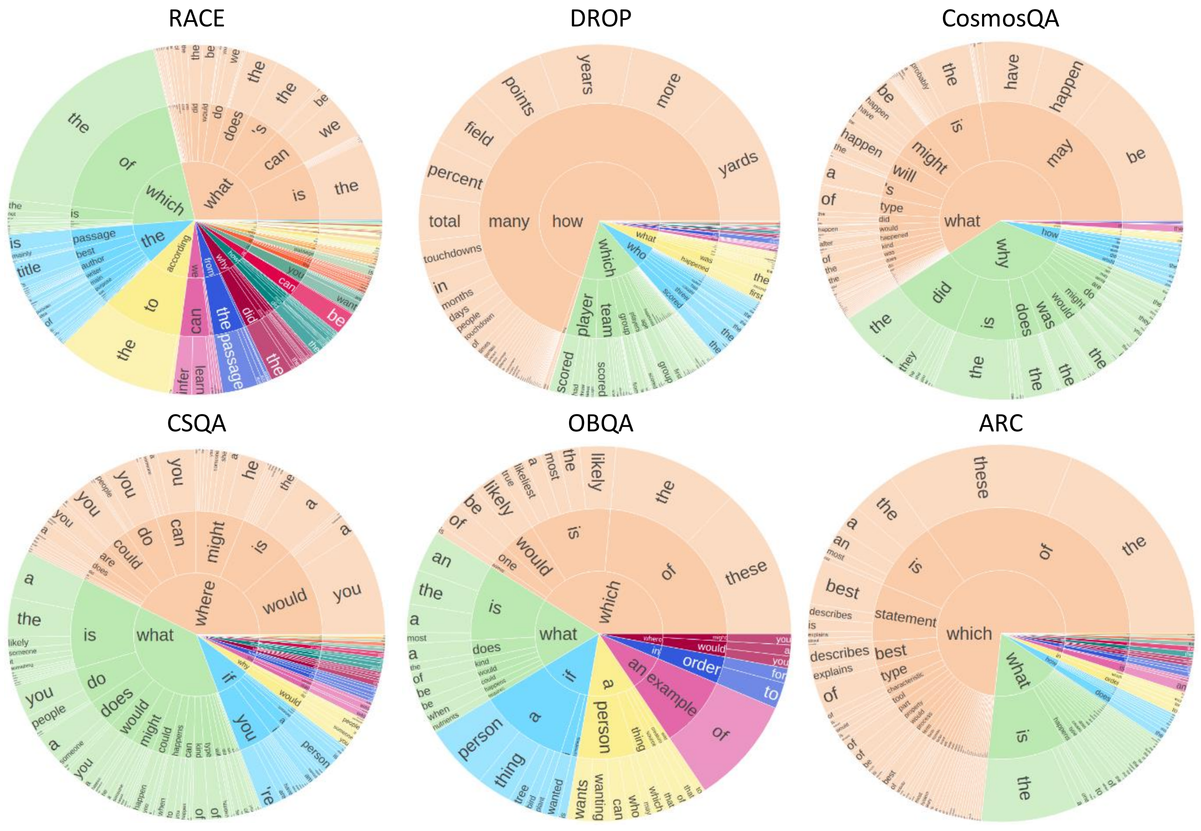


Figure 10. Distribution of the first three tokens in question (also see Appendix A)

The following observations are worth mentioning:

- Notice the high degree of similarity between OBQA and ARC questions: the top two trigrams in OBQA are “which of these” and “which of the”, whereas the top two trigrams in ARC are “which of the” and “which of these” (swapped order);
- Notice the large proportion of questions starting with “an example of” in OBQA. None of the other datasets have this trigram in their top 5 trigrams;
- A hefty proportion of RACE questions begin with “according to the” indicating that they may be descriptive questions (potentially non-factoid);
- A considerable amount of DROP questions start with “how many ...” highlighting the mathematical nature of the dataset - the answers are numerical;
- The CosmosQA dataset is dominated by questions targeting deeper reasoning and asking for explanations: “why …”, “what may be”, “what might be”. This is the only dataset with a non-negligible amount of “why” questions;

4.2 Investigating relations between QA datasets

In the current context of transfer learning, it is useful to observe how the datasets fit in a global picture, to what degree there is an overlap in their semantics, and how the datasets relate to one another. In the following experiments, pre-trained sentence transformers [81] (we used RoBERTa large with mean-tokens pooling) are applied to compute high-dimensional question embeddings. The embeddings have been shown to capture semantic aspects [81] and are usually employed in tasks such as semantic search, textual similarity, or sentence clustering. Dimensionality reduction is performed using a technique known as “Uniform Manifold Approximation and Projection (UMAP)” [82]. UMAP yields about the same visualization quality as t-SNE [83] but with considerably better runtime performance.

In the first experiment, we want to observe the relations between pairs of datasets. For this, 2000 questions are randomly sampled from each base dataset (excluding ARC, the “downstream” one), then embedded using RoBERTa and the derived vectors are reduced to two dimensions with UMAP (for visualization). A scatter plot with the question embeddings for each pair of datasets (independent plots) is shown in Figure 11.

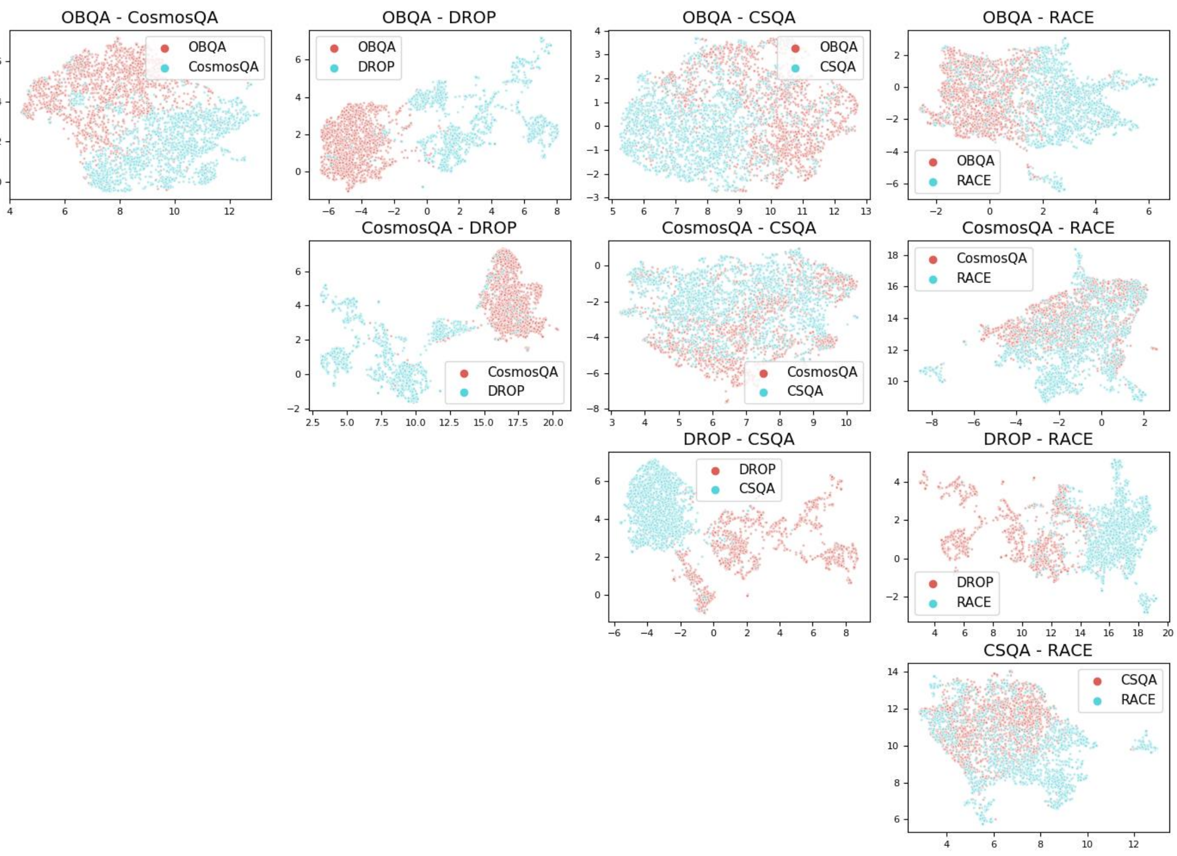
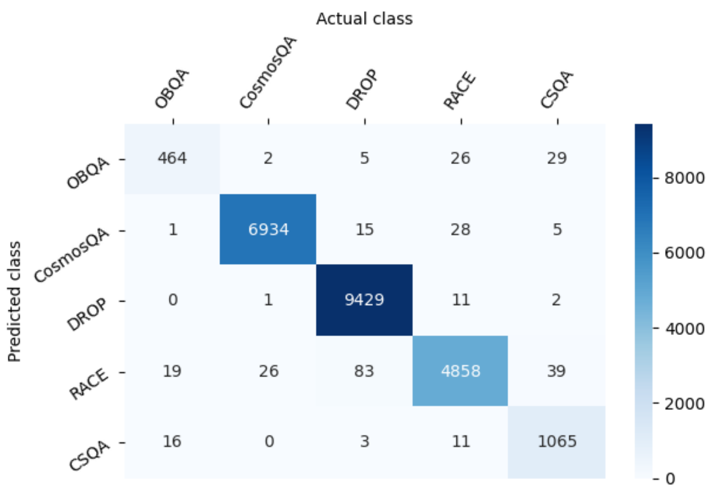


Figure 11. Question embeddings, reduced to 2D, showing relations between base datasets

Notice that some datasets are clearly separated (such as OBQA and CosmosQA or DROP and CSQA) indicating that the corresponding questions are covering different topics or are just semantically distant. On the other hand, for some datasets, there is a clear semantic overlap: RACE and OBQA or CosmosQA and CSQA. Another important observation is that we have not used specialized embeddings but some that are universal in their nature and usually applied for textual similarity, clustering, or semantic search. One can obtain a set of specialized embeddings by training a model to classify questions into their corresponding dataset. We made use of the pre-trained ALBERT XXLarge model and added a classification head consisting of two layers: first a projection on a 25D space, followed by a feed-forward layer with softmax activation (5 classes corresponding to all base datasets). The intermediate projection is going to be the source of the specialized question embeddings. The dimensionality (25) has been empirically chosen. As we have already mentioned, ARC is going to be the "downstream" dataset in the transfer learning schema and no models are going to be trained on ARC except for the final one, responsible for adapting the learned knowledge to ARC. Thus, the question classifier is only trained on data from the base datasets. The network has been fine-tuned for 3 epochs with the following hyper-parameters: batch size 32, learning rate 2e-5, maximum sequence length 192. During training, the model has been evaluated, on the validation set, once every 465 optimization steps (roughly 10 times per epoch). The best set of weights is going to be used for inference (on various questions including those from ARC). The final test accuracy is around 98.604%. Figure 12 shows the confusion matrix (predictions on the test sets). The very high classification accuracy highlights, once again, that the datasets differ in the types of questions they ask. This may be beneficial for the end-to-end purpose of transfer learning.



Predicted \ Actual	OBQA	CosmosQA	DROP	RACE	CSQA
OBQA	464	2	5	26	29
CosmosQA	1	6934	15	28	5
DROP	0	1	9429	11	2
RACE	19	26	83	4858	39
CSQA	16	0	3	11	1065

Figure 12. Confusion matrix for the question classification model (fine-tuned ALBERT)

For the second experiment, the universal sentence-transformer embeddings are replaced with the specialized embeddings extracted from the already described ALBERT XXLarge model (the 25D projection layer). We display the embeddings in Figure 13 (UMAP is used for dimensionality reduction).

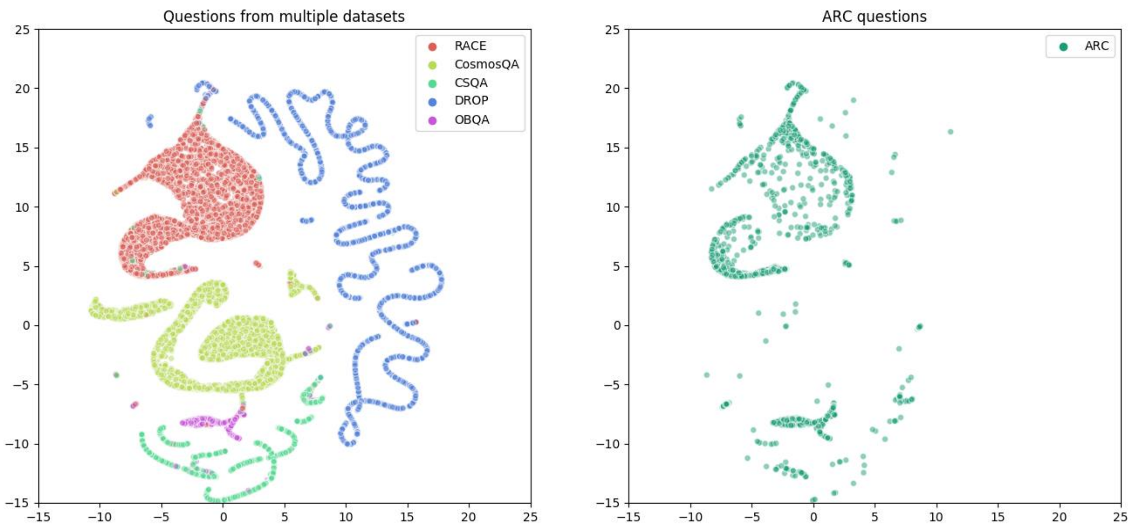


Figure 13. Specialized question embeddings projected to 2D with UMAP. Notice the semantic overlap between ARC and other datasets. Both plots share the same projection space.

Since some of the datasets are too large to visualize (e.g. RACE is over 90k questions) we just plot the dev and test splits. Moreover, using only dev and test questions guarantees that the experiment is not affected by some overfitting which may have occurred when the embedder model has been fine-tuned on the train sets. The questions in the base datasets (e.g. all but ARC) are used to fit the mapping from 25D to 2D, and the same transformation is applied to the ARC questions (Figure 13). The reader should acknowledge the self-evident separation of the base datasets and how ARC (the downstream dataset) relates to those datasets. There is a clear overlap between ARC and RACE and between ARC and OBQA. Indeed, if we use the question classification model to predict to which base dataset the ARC questions belong to, we obtain the histogram in Figure 14. The similarity between ARC and OBQA is undeniable. This is one of the main reasons OBQA is used to train or fine-tune models as an initial step in some architectures targeting ARC.

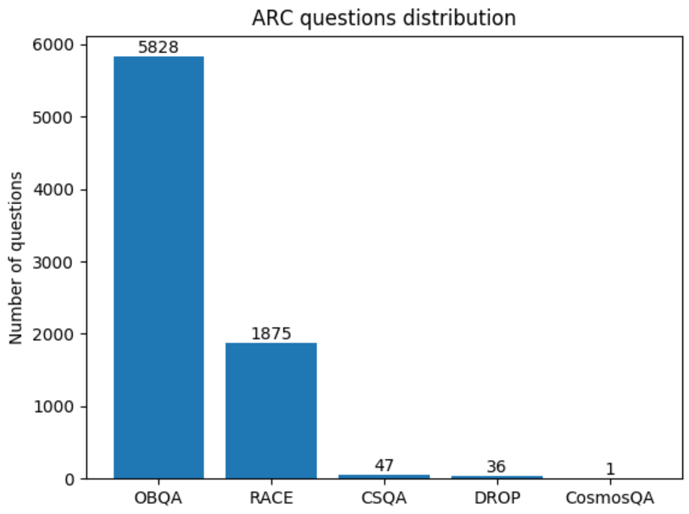


Figure 14. Each ARC question is classified according to the similarity with questions in the base datasets. OBQA has the largest semantic overlap with ARC, followed by RACE.

As a heads-up, the classification scores predicted for the ARC questions are going to be used in the final combiner model which is held responsible for adapting the learned knowledge on the downstream dataset. This would allow the combiner to take decisions accounting for the similarity between the target questions (e.g. those in ARC) and the questions in the base datasets.

4.3 Training models on the base datasets

The pre-trained ALBERT XXLarge v2 [40] has been fine-tuned, independently, on each of the aforementioned base QA datasets (RACE, MC DROP, OBQA, CosmosQA, and CSQA). The format of the input is the following one: “[CLS] <question> [SEP] Answer: <answer> [SEP] document [SEP]”. The model is trained to classify candidate answers into either wrong or correct (the models see only one candidate answer at a time). This framework allows the same architecture to be trained on datasets with **N** answer choices, irrespective of the value of **N** (4, 5, etc.). The predicted final answer is the one with the largest individual probability of being correct, from the list of **N** total choices. The hyper-parameters have been chosen based on the official recommendations [40] with very little tuning via trial and error. The default setting is specified in Table 32. The transformer model has been fine-tuned on a single Cloud TPU v3-8 for a few epochs, between 3 and 8 (depending on the dataset). Checkpoints had been saved at a rate of 10 per epoch and evaluated on the validation split. The checkpoint with the best validation accuracy is going to be used to make predictions on the ARC dataset. Table 33 shows the model performance on all base datasets.

Table 32

Default hyperparameter setting for fine-tuning ALBERT XXLarge v2

Hyperparameter name	**Value**
Train batch size	32
Maximum sequence length	512
Learning rate	2e-05
Optimizer	RAdam [84] with Lookahead [85]

Table 33

Performance of ALBERT XXLarge v2 fine-tuned on the base datasets

Dataset	**Dev Accuracy**	**Test Accuracy**
RACE (middle and high combined)	87.47%	86.01%
OpenBookQA	81.20%	81.00%
DROP	84.30%	N/A
CosmosQA	85.29%	85.38%
CSQA	80.99%	N/A

For some datasets, we experimented with a few variations of the default hyperparameters, as follows. In the case of CosmosQA, we decreased the default learning rate because the loss in the initial training suffered from an oscillating behavior and, additionally, cut down the input sequence length to 192 as the questions and contexts are short enough. The experiment results are shown in Table 34, below.

Table 34

Experiment results on the CosmosQA dataset

Hyper-parameters (delta from default)	**Dev accuracy (random = 25%)**
Maximum sequence length = 192	85.29%
Learning rate = 1e-05	84.99%

We submitted our version of fine-tuned ALBERT to the CosmosQA leaderboard and obtained a test accuracy of 85.38% (the Cosmos QA test labels are not publicly available so a submission to the leaderboard is required to evaluate the model on the test split). We also noticed that the test accuracy is very similar to the dev accuracy highlighting that the model can generalize on unseen examples. We had not performed any heavy hyper-parameter tuning on the validation set to avoid overfitting on the dev split (this can be another intuition for the similar dev and test accuracies).

The OBQA dataset requires an additional step before fine-tuning as no gold paragraphs are provided with the questions. Instead, information must be retrieved from the provided book of core science facts. In the next section, we describe the retrieval strategies and their performances.

4.3.1 Passage retrieval for OpenBookQA

We test the performance of the fine-tuned ALBERT XXLarge on OpenBookQA while experimenting with different schemes for fact extraction:

1) *No external passage* is provided during fine-tuning (the input consists of only the question and a candidate answer). Thus, we commit to the commonsense knowledge and maybe other general facts captured by the transformer model during its training [86, 87, 88];

2) *Vector-based retrieval* in which embeddings are used to retrieve semantically related facts. Sentence transformers [81] have been used to embed raw text into a high dimensional vector space where distances (e.g. cosine) capture semantic similarity. Embeddings have been extracted from the RoBERTa large model with mean-tokens pooling. The authors [81] have fine-tuned the RoBERTa transformer on natural language inference data showing that it can produce embeddings carrying semantic information. Evidence of such a fact has been also highlighted by Conneau et al., 2017 [17]. They articulate the fact that training models on natural language inference (NLI) data can produce universal sentence embeddings. A K-nearest neighbors (kNN) index with the 1326 science facts is created and saved for further querying. The cosine distance is used to measure the semantic similarity between a query vector and the vectors from the index (denoting science facts). The query vector is deduced from the question and maybe the candidate answers. Thus, the cosine distance is a proxy for the similarity between questions and facts. In this fashion, we try to retrieve information that is meaningful for answering the question. The query vector is computed using different approaches:

   i) Query vector = embedding(question);
   ii) Query vector = embedding(question + a candidate answer);
   iii) Query vector = embedding(question + all candidate answers, concatenated);

3) Token-based retrieval, using the "classical" information retrieval approach with token-based scoring. Lucene 8.4 [89, 90] has been used in our implementation, with the default TF-IDF metric parameters. An index is created with the 1326 facts and different strategies are tested at query time (same as for the vector-based approach):

   i) Query string = question;
   ii) Query string = question concatenated with a candidate choice;
   iii) Query string = question concatenated with all candidate choices;
   iv) Query string = (question) AND (a candidate choice). This would match facts that contain at least one term from the question text and one from the candidate answer. Given the fact that OpenBookQA has very short answers to a lot of questions, the number of returned facts is limited, missing a lot of important sentences from the knowledge base (sometimes no facts are returned at all). This explains the low accuracy scores (Table 35);

In all experiments, the top 10 retrieved sentences have been used as the supporting context. In some cases, Lucene returned less than 10 matching facts from the KB, in which case all of them have been used but not nothing else was added to compensate the missing ones. The resulted contexts are used to fine-tune ALBERT XXLarge v2 to predict the correct answer. The QA accuracy is a proxy for the passage quality: higher accuracy should imply that a better context is provided. Table 35 shows the accuracies for each of the retrieval strategies. It is clear that providing extra supporting information helps ALBERT to predict better answers. For the vector-based retrieval, the best querying strategy is the second one (2ii) in which both the question and the candidate answer are used in the query representation. The same applies to the token-based setting. The final context for OBQA is a combination of the token-based and vector-based retrieved facts: sentences are interleaved until the context exceeds 512 tokens (maximum sequence length allowed for ALBERT). We have made the resulting contexts publicly available as part of the DataMine[3] library. Examples of extracted facts can be found in Appendix B & C (using strategy 2i and 3i).

[3] https://github.com/SebiSebi/DataMine

Table 35

Performance on OpenBookQA for different retrieval strategies. The token-based approach extracts more meaningful facts (indicated by the better QA accuracies).

Experiment	Dev accuracy	Test accuracy
No external information	68.20%	66.80%
2i (question only)	79.20%	75.20%
2ii (question and 1 answer)	79.00%	**77.00%**
2iii (question and all answers)	80.60%	75.20%
3i (question only)	79.80%	76.00%
3ii (question and 1 answer)	81.80%	**80.60%**
3iii (question and all answers)	83.00%	79.60%
3iiii (question AND answer)	75.40%	73.60%

The code and instructions to reproduce the experiments can be found at this URL.

4.4 Transferring knowledge to the downstream dataset

The models fine-tuned on the base datasets are employed to answer questions in ARC (the downstream dataset). We are going to refer to those models simply as **base models**. There are five base models in total: ALBERT RACE, ALBERT DROP, ALBERT OBQA, ALBERT CosmosQA, and ALBERT CSQA. Each base model computes a probability distribution over the candidate answers (e.g. the scores of the base model). A classification model is then trained to learn a function of the predicted scores to maximize the QA accuracy on ARC. This is the only neural network we train on ARC and it is an extension of the combiner used in the Attentive Ranker [62]. The combiner network is responsible for adapting the learned knowledge from the base datasets to the downstream task. The combiner architecture with all the details can be found in Appendix D. We refer to the end-to-end approach as **ZeroQA**. It is very important to highlight, once again, that no transformer has been fine-tuned on the ARC data. ZeroQA relies only on transfer learning to predict the correct answer. For the first experiment, we fed the combiner only one score at a time – one for each of the base datasets. The results indicate the relative importance of each base dataset when adapting knowledge to ARC (Table 36).

Table 36

Performance of ZeroQA when a single score is provided as the input (one dataset at a time). This shows which datasets are the most appropriate for transfer learning on ARC. The shape of the encoded answer sent to ZeroQA is (40, 1) - 40 documents, 1 score.

Model	ARC Easy Dev Accuracy	ARC Challenge Dev accuracy
ZeroQA - RACE only	75.49%	59.32%
ZeroQA - OBQA only	65.43%	52.54%
ZeroQA - DROP only	57.50%	39.66%
ZeroQA - CosmosQA only	77.60%	59.32%
ZeroQA - CSQA only	67.02%	55.93%

Sorted in decreasing order of the induced ARC accuracy, the base datasets stand as follows: CosmosQA, RACE, CSQA, OBQA, and DROP. Both CosmosQA and CSQA (1st and 3rd on the list) are datasets targeting commonsense knowledge. This indicates that the questions in ARC themselves require a vast amount of everyday world knowledge. On the other hand, RACE covers a wide range of topics and question types and it is expected to be a valuable source for training base models for domain adaptation. The less “effective” base dataset is DROP. This is caused by the fact that not so many ARC questions focus on discrete mathematical reasoning, counting, or ordering. Figure 13 provides some insights behind the results in Table 36 and we recommend the reader to take a look at both Figure 13 and Table 36 in parallel. We computed two metrics which offer insightful information about the overlap between individual ZeroQA models:

- How many questions are correctly answered by X different models? In this context, we refer to a model as a ZeroQA instance trained with only the scores from a single base model. An important observation is that there are 26 questions in ARC Easy Validation and 28 in ARC Challenge Validation which are incorrectly answered by all models - those are the difficult questions for ZeroQA (Figure 15).
- How many questions are correctly answered by a pair of models? (Figure 16) As with the previous metric, a model here is a ZeroQA instance trained with only the scores from a single base model. The values indicate to what degree two models answer correctly the same set of questions. For example, ALBERT CSQA has the largest overlap with ALBERT CosmosQA, which makes sense given the fact that both CSQA and CosmosQA are datasets targeting common-sense knowledge.

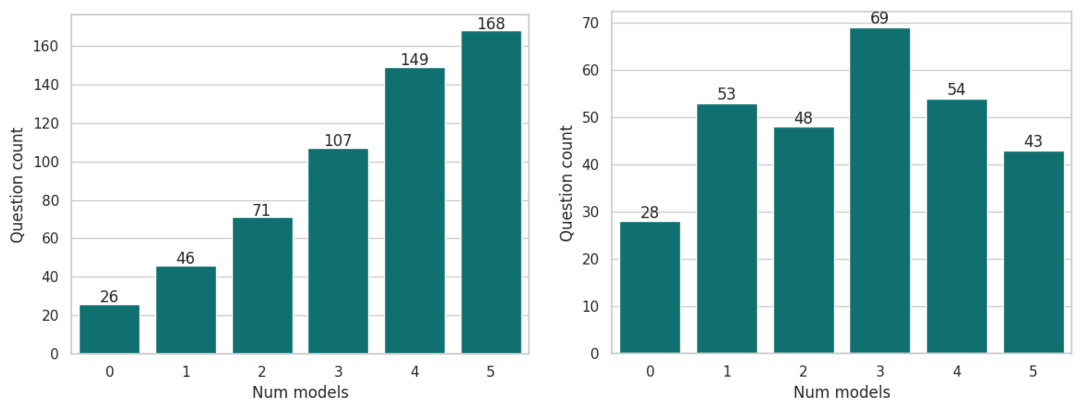


Figure 15. How many ARC questions are correctly answered by X different models? - Left is ARC easy, right is ARC challenge (validation split). A model, here, refers to a ZeroQA instance trained with only the scores from a single base model

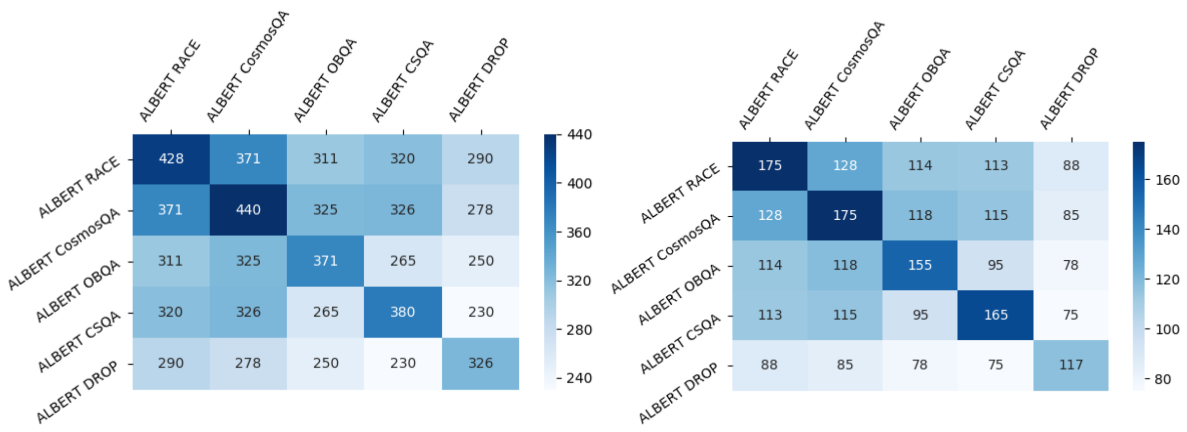


Figure 16. How many ARC questions are correctly answered by both models? - Left is ARC easy, right is ARC challenge (validation split). A model, here, refers to a ZeroQA instance trained with only the scores from a single base model

To see the full power of ZeroQA we have to feed all the scores predicted by the base models into the combiner, at the same time. Moreover, we have also provided the discriminator scores from the Attentive Ranker [62] to the combiner. We are particularly interested in the document relevance discriminator (DRD) scores which classify the supporting documents into important or "noisy". Due to the small size of the ARC dev splits, the experimental results have a non-negligible variance. To overcome the effects of this phenomenon, we have trained the combiner five times with different weight initializations and reported the mean accuracy (Table 37). As a side note, this procedure is different from an ensemble in which the predictions are averaged first and then the performance metric is computed based on the average scores (the accuracy is not continuous).

Table 37

Performance of ZeroQA on ARC Easy and Challenge (random accuracy 25%)

Model	**ARC Easy Dev Accuracy**	**ARC Challenge Dev Accuracy**
Attentive Ranker (ALBERT only)	82.11%	65.42%
Attentive Ranker (full)	84.16%	66.91%
ZeroQA	86.52%	70.64%
ZeroQA + question classification scores	86.24%	71.32%

It may be useful to inform the combiner about the relation between the ARC question it tries to answer and the questions in the base datasets. To do so, we ask the question classification model (previously trained on the base datasets) to analyze the questions in ARC. It indicates to what degree an ARC question is related/similar to questions in the base datasets: DROP, RACE, CosmosQA, CSQA, and OBQA. The predicted distribution is concatenated to the combiner input (Table 37, last row). We can see that the question classification scores lead to a slight improvement in the challenge split, but they are not valuable for easy questions.

We can try to improve the performance of the ZeroQA combiner network in three different ways:

a) Use more supporting documents – in all experiments up to this point, the top 40 documents extracted from Wikipedia and the ARC Corpus [1] (20 from each source) have been used by ZeroQA. The retrieval procedure is described by Pîrtoacă et al. [62]. We have trained the combiner by providing it with multiple supporting documents and the results are shown in Figure 17. There is a slight improvement after surpassing 40 documents but the variance in results prevents us from drawing a clear conclusion;

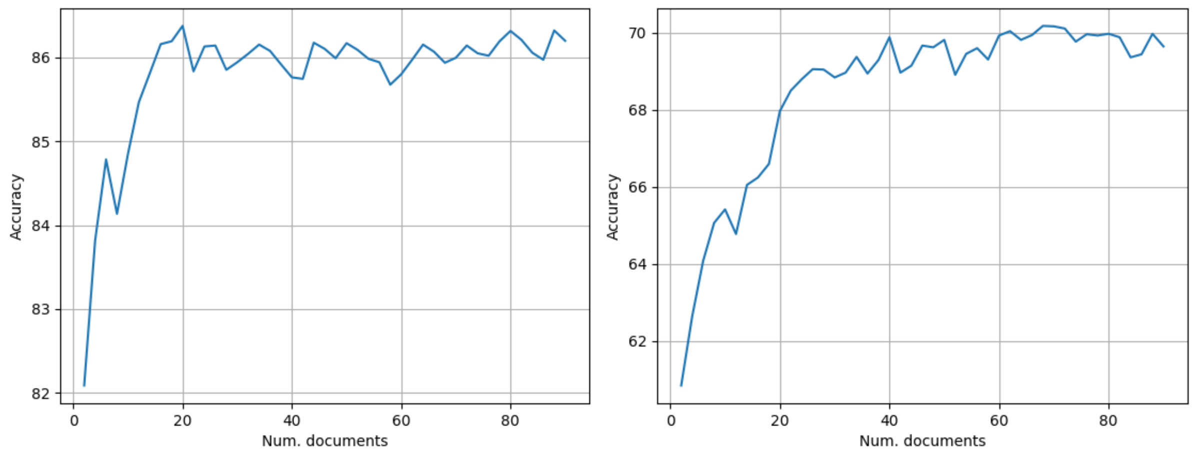


Figure 17. The effect of the number of supporting documents to the QA accuracy (left is ARC easy, right is ARC challenge, validation splits).

b) Fine-tuning various hyperparameters using Keras Tuner [91] (for TensorFlow 2.0 [92]). A Gaussian process is guiding the exploration of the hyperparameter search space. We have been experimenting with different learning rates, optimizers, activation functions, batch sizes, or the number of neurons in hidden layers. In total, about 750 configurations have been tested using 2 Tesla K40 GPUs totaling around 6 days of continuous computation. Results are shown in Table 38;

c) Ensemble several combiner models – we had trained the same architecture 10 times starting from different initializations. The final scores are averaged. Table 38 shows the results.

Table 38

Various experiments on the combiner architecture responsible for transfer learning

Model	ARC Easy Dev Accuracy	ARC Challenge Dev Accuracy
ZeroQA (previous best)	86.24%	71.32%
ZeroQA + hyperparameter fine-tuning	86.24%	71.53%
ZeroQA + ensemble combiners	86.07%	71.86%
ZeroQA + all above optimizations	**88.36%**	**73.56%**

4.4.1 Investigate a potential overfit on the base datasets

ZeroQA is built on the top of different transformer models which have been fine-tuned on various base datasets with the final purpose of transferring the knowledge on some downstream tasks (e.g. ARC). However, it is worth investigating a potential overfit caused by some questions from the downstream task being similar to questions from the base datasets. This triggers the possibility in which the base models might have just memorized the correct answers, which is feasible given the huge number of parameters in their architectures. Although this phenomenon is hard to quantify and measure, we tried to investigate to which extent ARC questions are alike those in the base datasets. To do so, we have used the TF-IDF metric as a similarity measure between questions. We have ignored questions that are too long (more than 20 words) as they may cause large TF-IDF scores although not being so (semantically) relevant. In Appendix E, we listed the "closest" pairs of questions, according to the TF-IDF scores, one from the ARC and another one from a base dataset.

We can observe that there are some questions extremely related to one another:

- "Today, almost all cars have seat belts. How does improving the design of seat belts help people the most?" (question from ARC)
- "Seat belts _" (question from RACE)

Another example would be:

- ARC: "The Sun is classified as a star. Which characteristic identifies the Sun as a star?"
- DROP: "Which stars are sun-like?"

And the last one:

- ARC: "Why is it winter in North America when it is summer in South America?"
- CSQA: "South America is in winter when North America is in summer, this is because it is located where?"

Quantifying the exact amount of question overlap between datasets is quite difficult and we would like to leave the problem open for further research (we expect it to require some manual labeling and analysis). Nonetheless, the above investigation indicates that the models are susceptible to memorizing some questions (from the base datasets) and just reproducing the correct answer when evaluated on the downstream task.

4.4.2 Using a better pre-trained language model

We have adopted the pre-trained T5 11B model [41] and fine-tuned it on the base datasets. The general framework remains the same, however, ALBERT is replaced by the T5 transformer (the flavor with 11 billion parameters). The input has the following format: "**question**: <question> **choice 0**: <answer 0> **choice 1**: <answer 1> **choice 2**: <answer 2> **choice 3**: <answer 3> **article**: <extracted passage>). In the case of CSQA, where questions have 5 candidate answers, another entry with the format "**choice 4**: <answer 4>" is added to the input. The model is fine-tuned to predict the label of the correct answer with the following hyper-parameters (Table 39):

Table 39

Hyperparameter values used for fine-tuning T5 11B on the base datasets

Hyperparameter name	Value
Train batch size	8
Maximum input sequence length	512
Learning rate	0.002 or 0.003
Fine-tuning steps	Between 10k and 100k

Table 40 reports the T5 11B performance after fine-tuning and compares the results with the scores obtained by ALBERT XXLarge V2 (see Table 33). T5 11B improves the performance on all base datasets except for DROP. We carried out the same ablation study as in the case of ALBERT, feeding the combiner one score at a time (from each base dataset) – Table 41.

Table 40

Performance of the fine-tuned T5 11B on the base datasets (delta from ALBERT)

Dataset	Dev Accuracy	Test Accuracy
RACE (middle and high combined)	89.73% (+2.26%)	87.48% (+1.47%)
OBQA	82.80% (+1.60%)	85.40% (+4.40%)
DROP	78.81% (-5.49%)	N/A
CosmosQA	90.92% (+5.63%)	90.29% (+4.91%)
CSQA	84.36% (+3.37%)	N/A

Table 41

Performance of ZeroQA when a single score is provided as the input (one dataset at a time).

Model	ARC Easy Dev Accuracy	ARC Challenge Dev Accuracy
ZeroQA T5 - RACE only	89.95%	76.61%
ZeroQA T5 - OBQA only	85.19%	70.85%
ZeroQA T5- DROP only	83.42%	63.73%
ZeroQA T5- CosmosQA only	87.13%	64.75%
ZeroQA T5 - CSQA only	79.01%	55.25%

Up to this point, we have reported results only on the development partitions to make sure the results on the test split reflect the actual performance of the model. In Table 42 we compare our best model (ZeroQA with both T5 and ALBERT scores) with other proposed approaches. ZeroQA surpasses the current SOTA model (which is UnifiedQA [60]) on both ARC datasets by a small margin. However, UnifiedQA has an underlying language model (T5 11B) fine-tuned on ARC. On the other hand, the largest model that we ever train on ARC is the combiner neural network with 20,629 parameters. ZeroQA only relies on adapting the learned knowledge from the base datasets to the downstream task. We have shown that this can happen even if the source and the target tasks have a totally different format or structure (even the question types are different). Nevertheless, we would like to highlight the end-to-end QA improvement brought by using T5 11B. We hypothesize that the pre-trained language model has captured a vast amount of knowledge (both general and commonsense) and it can use that during question answering.

Table 42

Evaluation on the ARC dataset (link to leaderboards: Easy, Challenge). Models are sorted by the ARC Challenge test accuracies. UnifiedQA was the previous SOTA on both datasets. Here, ZERO T5 includes both the T5 11B and the ALBERT XXLarge predictions.

Model	Easy Dev	Easy Test	Challenge Dev	Challenge Test
Reading Strategies [59]	N/A	68.90%	N/A	42.32%
AristoBERTv7	N/A	79.29%	N/A	57.76%
Attentive Ranker [62]	82.89%	84.82%	65.76%	64.54%
ZeroQA ALBERT (ours)	88.36%	86.77%	73.56%	65.92%
FreeLB-RoBERTa [93]	84.91%	85.44%	70.23%	67.75%
UnifiedQA [60]	N/A	92.05%	N/A	78.50%
ZeroQA T5 (ours)	**92.59%**	**92.64%**	**77.29%**	**78.71%**

5. Conclusions

This dissertation thesis has described an approach to the question answering task that relies on transfer learning to adapt the knowledge acquired on some base datasets to a downstream task. First, we have presented the most important advancements in natural language processing and how they relate to the QA field. Second, we proposed a robustly optimized version of the Attentive Ranker [62] that improves its performance by 19.83% on ARC Challenge and by 12.64% on ARC Easy. Last but not least, we introduced ZeroQA – a framework for zero-shot-like question answering. Five source datasets (RACE, OBQA, CSQA, DROP, and CosmosQA) have been used to fine-tune powerful language models (ALBERT and T5). The resulting models are used to predict correct answers in a downstream QA dataset (in our experiments, the Allen AI Reasoning Challenge dataset). Then, a combiner network is employed as the final decision component. ZeroQA obtained good results on both ARC Easy and ARC Challenge even though the largest neural network trained on ARC has only 20k parameters.

BIBLIOGRAPHY

1. Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., & Tafjord, O. (2018). Think you have solved question answering? try arc, the ai2 reasoning challenge. arXiv preprint arXiv:1803.05457.
2. Zhong, W., Tang, D., Duan, N., Zhou, M., Wang, J., & Yin, J. (2019, October). Improving question answering by commonsense-based pre-training. In CCF International Conference on Natural Language Processing and Chinese Computing (pp. 16-28). Springer, Cham.
3. Clark, P., Harrison, P., & Balasubramanian, N. (2013, October). A study of the knowledge base requirements for passing an elementary science test. In Proceedings of the 2013 workshop on Automated knowledge base construction (pp. 37-42).
4. Levesque, H., Davis, E., & Morgenstern, L. (2012, May). The winograd schema challenge. In Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning.
5. Boratko, M., Padigela, H., Mikkilineni, D., Yuvraj, P., Das, R., McCallum, A., ... & Musa, R. (2018). A systematic classification of knowledge, reasoning, and context within the arc dataset. arXiv preprint arXiv:1806.00358.
6. Liu, S., Zhang, X., Zhang, S., Wang, H., & Zhang, W. (2019). Neural machine reading comprehension: Methods and trends. Applied Sciences, 9(18), 3698.
7. Qiu, B., Chen, X., Xu, J., & Sun, Y. (2019). A survey on neural machine reading comprehension. arXiv preprint arXiv:1906.03824.
8. Riloff, E., & Thelen, M. (2000, May). A rule-based question answering system for reading comprehension tests. In Proceedings of the 2000 ANLP/NAACL Workshop on Reading comprehension tests as evaluation for computer-based language understanding sytems-Volume 6 (pp. 13-19). Association for Computational Linguistics.
9. Ko, J., Nyberg, E., & Si, L. (2007, July). A probabilistic graphical model for joint answer ranking in question answering. In Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval (pp. 343-350).
10. Echihabi, A., & Marcu, D. (2003, July). A noisy-channel approach to question answering. In Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1 (pp. 16-23). Association for Computational Linguistics.
11. Feng, M., Xiang, B., Glass, M. R., Wang, L., & Zhou, B. (2015, December). Applying deep learning to answer selection: A study and an open task. In 2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU) (pp. 813-820). IEEE.
12. Tuason, R., Grazian, D., & Kondo, G. (2017). Bidaf model for question answering. Table III EVALUATION ON MRC MODELS (TEST SET). Search Zhidao All.
13. Seo, M., Kembhavi, A., Farhadi, A., & Hajishirzi, H. (2016). Bidirectional attention flow for machine comprehension. arXiv preprint arXiv:1611.01603.
14. Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.
15. Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Advances in neural information processing systems (pp. 3104-3112).
16. Lee, K., He, L., Lewis, M., & Zettlemoyer, L. (2017). End-to-end neural coreference resolution. arXiv preprint arXiv:1707.07045.
17. Conneau, A., Kiela, D., Schwenk, H., Barrault, L., & Bordes, A. (2017). Supervised learning of universal sentence representations from natural language inference data. arXiv preprint arXiv:1705.02364.
18. McCann, B., Bradbury, J., Xiong, C., & Socher, R. (2017). Learned in Translation: Contextualized Word Vectors. ArXiv, abs/1708.00107.
19. Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks, 5(2), 157-166.
20. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

21. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.
22. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).
23. Cheng, J., Dong, L., & Lapata, M. (2016). Long short-term memory-networks for machine reading. arXiv preprint arXiv:1601.06733.
24. Lin, Z., Feng, M., Santos, C. N. D., Yu, M., Xiang, B., Zhou, B., & Bengio, Y. (2017). A structured self-attentive sentence embedding. arXiv preprint arXiv:1703.03130.
25. Yang, Z., He, X., Gao, J., Deng, L., & Smola, A. J. (2015). Stacked attention networks for image question answering. CoRR abs/1511.02274 (2015).
26. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
27. Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. arXiv preprint arXiv:1607.06450.
28. Van Laarhoven, P. J., & Aarts, E. H. (1987). Simulated annealing. In Simulated annealing: Theory and applications (pp. 7-15). Springer, Dordrecht.
29. Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. arXiv preprint arXiv:1705.03122v2, 2017.
30. Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144, 2016.
31. Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. 2018. Character-level language modeling with deeper self-attention. arXiv preprint arXiv:1808.04444.
32. Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., & Salakhutdinov, R. (2019). Transformer-xl: Attentive language models beyond a fixed-length context. arXiv preprint arXiv:1901.02860.
33. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.
34. Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning. Technical report, OpenAI.
35. Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. arXiv preprint arXiv:1802.05365.
36. Dai, A. M., & Le, Q. V. (2015). Semi-supervised sequence learning. In Advances in neural information processing systems (pp. 3079-3087).
37. Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., & Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. In Advances in neural information processing systems (pp. 5754-5764).
38. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.
39. Ott, M., Edunov, S., Grangier, D., & Auli, M. (2018). Scaling neural machine translation. arXiv preprint arXiv:1806.00187.
40. Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2019). Albert: A lite bert for self-supervised learning of language representations. arXiv preprint arXiv:1909.11942.
41. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv preprint arXiv:1910.10683.
42. Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. (2018). Glue: A multi-task benchmark and analysis platform for natural language understanding. arXiv preprint arXiv:1804.07461.
43. Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine learning, 20(3), 273-297.
44. Clark, P., Etzioni, O., Khot, T., Sabharwal, A., Tafjord, O., Turney, P., & Khashabi, D. (2016, March). Combining retrieval, statistics, and inference to answer elementary science questions. In Thirtieth AAAI Conference on Artificial Intelligence.

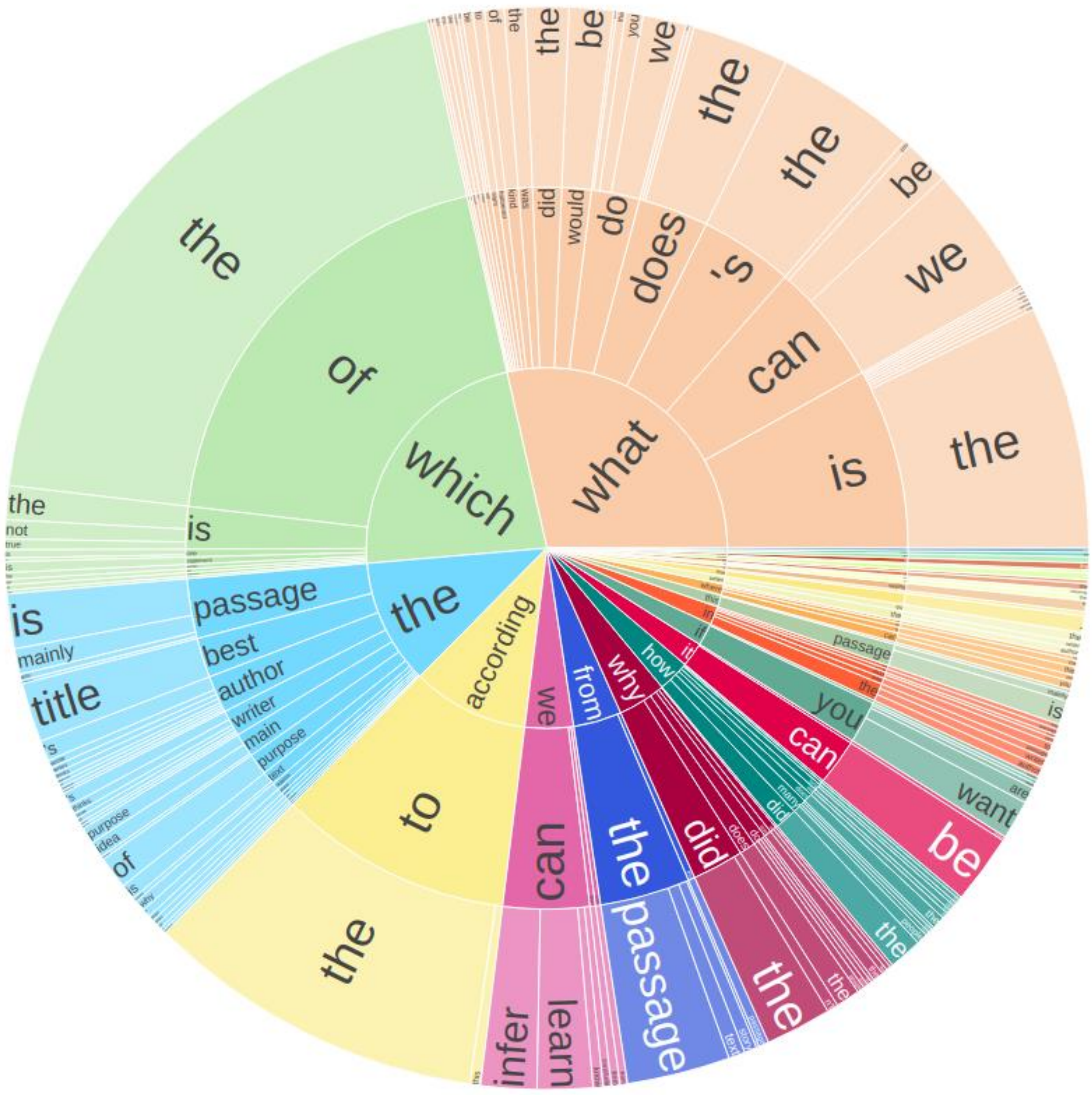
45. Khot, T., Balasubramanian, N., Gribkoff, E., Sabharwal, A., Clark, P., & Etzioni, O. (2015, September). Exploring Markov logic networks for question answering. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (pp. 685-694).
46. Zhang, Y., Dai, H., Toraman, K., & Song, L. (2018). KG^ 2: Learning to Reason Science Exam Questions with Contextual Knowledge Graph Embeddings. arXiv preprint arXiv:1805.12393.
47. Zhang, Y., Dai, H., Kozareva, Z., Smola, A. J., & Song, L. (2018, April). Variational reasoning for question answering with knowledge graph. In Thirty-Second AAAI Conference on Artificial Intelligence.
48. Angeli, G., Premkumar, M. J. J., & Manning, C. D. (2015, July). Leveraging linguistic structure for open domain information extraction. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers) (pp. 344-354).
49. Dai, H., Dai, B., & Song, L. (2016, June). Discriminative embeddings of latent variable models for structured data. In International Conference on Machine Learning (pp. 2702-2711).
50. Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017). Neural message passing for quantum chemistry. arXiv preprint arXiv:1704.01212.
51. Liu, H., & Singh, P. (2004). ConceptNet—a practical commonsense reasoning tool-kit. BT technology journal, 22(4), 211-226.
52. Havasi, C., Speer, R., & Alonso, J. (2007, September). ConceptNet 3: a flexible, multilingual semantic network for common sense knowledge. In Recent advances in natural language processing (pp. 27-29). Philadelphia, PA: John Benjamins.
53. Speer, R., & Havasi, C. (2013). ConceptNet 5: A large semantic network for relational knowledge. In The People’s Web Meets NLP (pp. 161-176). Springer, Berlin, Heidelberg.
54. Wang, L.; Sun, M.; Zhao, W.; Shen, K.; and Liu, J. 2018. Yuanfudao at semeval-2018 task 11: Three-way attention and relational knowledge for commonsense machine comprehension. arXiv preprint arXiv:1803.00191
55. Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2008). The graph neural network model. IEEE Transactions on Neural Networks, 20(1), 61-80.
56. Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).
57. Ni, J., Zhu, C., Chen, W., & McAuley, J. (2018). Learning to attend on essential terms: An enhanced retriever-reader model for scientific question answering. arXiv preprint arXiv:1808.09492.
58. Khashabi, D., Khot, T., Sabharwal, A., & Roth, D. (2017). Learning what is essential in questions. In Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017) (pp. 80-89).
59. Sun, K., Yu, D., Yu, D., & Cardie, C. (2018). Improving machine reading comprehension with general reading strategies. arXiv preprint arXiv:1810.13441.
60. Khashabi, D., Khot, T., Sabharwal, A., Tafjord, O., Clark, P., & Hajishirzi, H. (2020). Unifiedqa: Crossing format boundaries with a single qa system. arXiv preprint arXiv:2005.00700.
61. Pîrtoacă, G., Rebedea, T., & Ruseti, S. (2018). Improving Retrieval-Based Question Answering with Deep Inference Models. 2019 International Joint Conference on Neural Networks (IJCNN), 1-8.
62. Pîrtoacă, G. S., Rebedea, T., & Ruseti, S. (2019). Answering questions by learning to rank--Learning to rank by answering questions. arXiv preprint arXiv:1909.00596.
63. Rajpurkar, P., Jia, R., & Liang, P. (2018). Know what you don't know: Unanswerable questions for SQuAD. arXiv preprint arXiv:1806.03822.
64. Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. IEEE transactions on Signal Processing, 45(11), 2673-2681.
65. Pan, X., Sun, K., Yu, D., Chen, J., Ji, H., Cardie, C., & Yu, D. (2019). Improving question answering with external knowledge. arXiv preprint arXiv:1902.00993.
66. Lai, G., Xie, Q., Liu, H., Yang, Y., & Hovy, E. (2017). Race: Large-scale reading comprehension dataset from examinations. arXiv preprint arXiv:1704.04683.

67. Weston, J., Bordes, A., Chopra, S., Rush, A. M., van Merriënboer, B., Joulin, A., & Mikolov, T. (2015). Towards ai-complete question answering: A set of prerequisite toy tasks. arXiv preprint arXiv:1502.05698
68. Onishi, T., Wang, H., Bansal, M., Gimpel, K., & McAllester, D. (2016). Who did what: A large-scale person-centered cloze dataset. arXiv preprint arXiv:1608.05457.
69. Joshi, M., Choi, E., Weld, D. S., & Zettlemoyer, L. (2017). Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. arXiv preprint arXiv:1705.03551.
70. 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.
71. Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). Squad: 100,000+ questions for machine comprehension of text. arXiv preprint arXiv:1606.05250.
72. Mihaylov, T., Clark, P., Khot, T., & Sabharwal, A. (2018). Can a suit of armor conduct electricity? a new dataset for open book question answering. arXiv preprint arXiv:1809.02789.
73. Huang, L., Bras, R. L., Bhagavatula, C., & Choi, Y. (2019). Cosmos QA: Machine reading comprehension with contextual commonsense reasoning. arXiv preprint arXiv:1909.00277.
74. Peter Norvig. 1987. A Unified Theory of Inference for Text Understanding. Ph.D. thesis, EECS Department, University of California, Berkeley.
75. Andrew Gordon and Reid Swanson. 2009. Identifying personal stories in millions of weblog entries. In Third International Conference on Weblogs and Social Media, Data Challenge Workshop.
76. Kevin Burton, Akshay Java, Ian Soboroff, et al. 2009. The icwsm 2009 spinn3r dataset. In Proceedings of ICWSM 2009.
77. Talmor, A., Herzig, J., Lourie, N., & Berant, J. (2018). Commonsenseqa: A question answering challenge targeting commonsense knowledge. arXiv preprint arXiv:1811.00937.
78. Speer, R., Chin, J., & Havasi, C. (2016). ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. AAAI.
79. Dua, D., Wang, Y., Dasigi, P., Stanovsky, G., Singh, S., & Gardner, M. (2019). DROP: A Reading Comprehension Benchmark Requiring Discrete Reasoning Over Paragraphs. NAACL-HLT.
80. Pan, S. J., & Yang, Q. (2009). A survey on transfer learning. IEEE Transactions on knowledge and data engineering, 22(10), 1345-1359.
81. Reimers, N., & Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. arXiv preprint arXiv:1908.10084.
82. McInnes, L., Healy, J., & Melville, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426.
83. Maaten, L. V. D., & Hinton, G. (2008). Visualizing data using t-SNE. Journal of machine learning research, 9(Nov), 2579-2605.
84. Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., & Han, J. (2019). On the variance of the adaptive learning rate and beyond. arXiv preprint arXiv:1908.03265.
85. Zhang, M., Lucas, J., Ba, J., & Hinton, G. E. (2019). Lookahead Optimizer: k steps forward, 1 step back. In Advances in Neural Information Processing Systems (pp. 9593-9604).
86. Phu Mon Htut, Jason Phang, Shikha Bordia, and Samuel R Bowman. 2019. Do attention heads in BERT track syntactic dependencies? arXiv preprint arXiv:1911.12246.
87. Petroni, F., Rocktäschel, T., Lewis, P., Bakhtin, A., Wu, Y., Miller, A. H., & Riedel, S. (2019). Language Models as Knowledge Bases?. arXiv preprint arXiv:1909.01066.
88. Tenney, I., Xia, P., Chen, B., Wang, A., Poliak, A., McCoy, R. T., ... & Pavlick, E. (2019). What do you learn from context? probing for sentence structure in contextualized word representations. arXiv preprint arXiv:1905.06316.
89. McCandless, M., Hatcher, E., Gospodnetić, O., & Gospodnetić, O. (2010). Lucene in action (Vol. 2). Greenwich: Manning.
90. Białecki, A., Muir, R., Ingersoll, G., & Imagination, L. (2012, August). Apache lucene 4. In SIGIR 2012 workshop on open source information retrieval (p. 17).
91. O'Malley, T. (2020). Hyperparameter tuning with Keras Tuner.

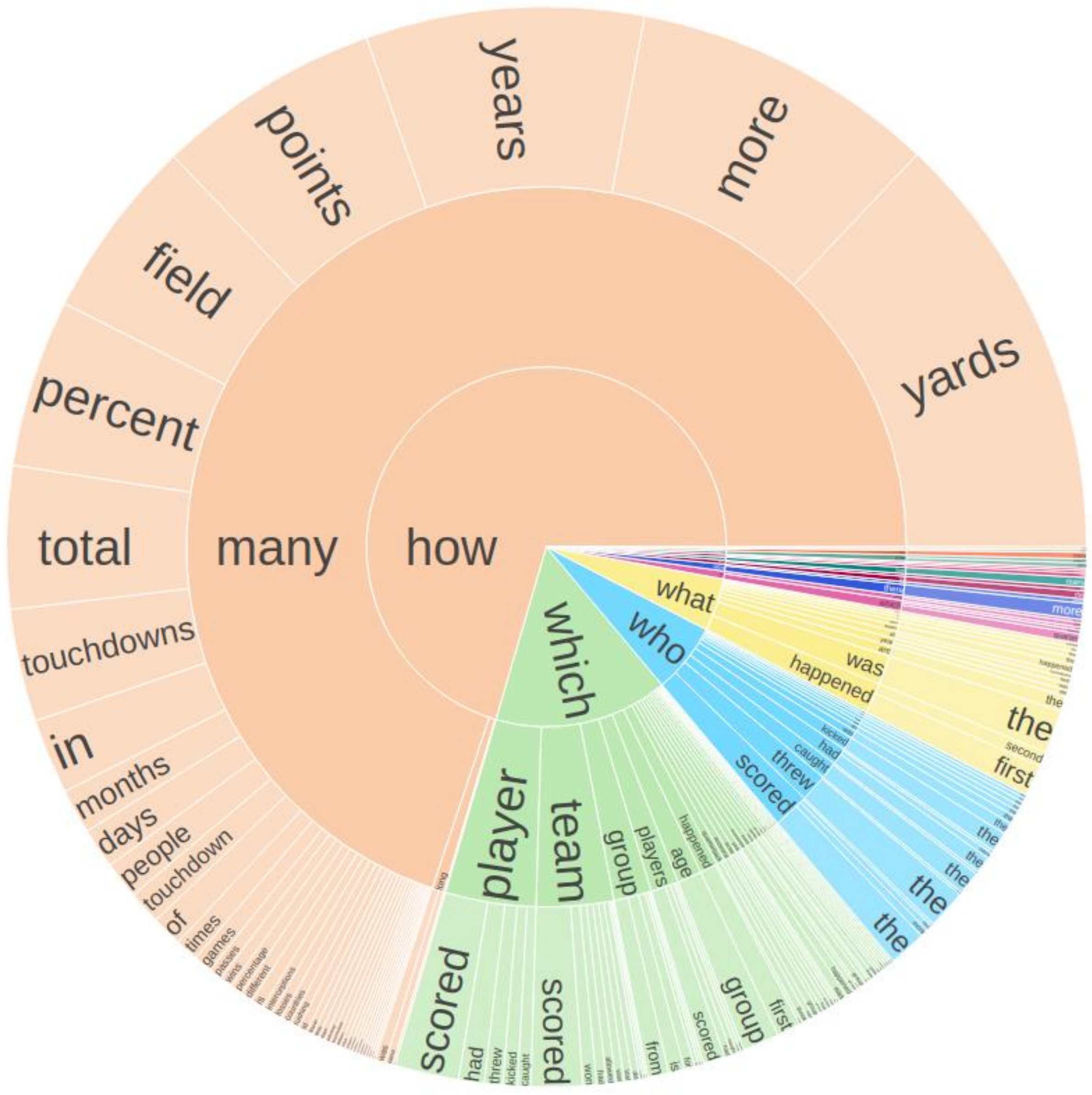
92. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Kudlur, M. (2016). Tensorflow: A system for large-scale machine learning. In 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16) (pp. 265-283).
93. Zhu, C., Cheng, Y., Gan, Z., Sun, S., Goldstein, T., & Liu, J. (2019). Freelb: Enhanced adversarial training for language understanding. arXiv preprint arXiv:1909.11764.
94. Wold, S., Esbensen, K., & Geladi, P. (1987). Principal component analysis. Chemometrics and intelligent laboratory systems, 2(1-3), 37-52.

Appendix A

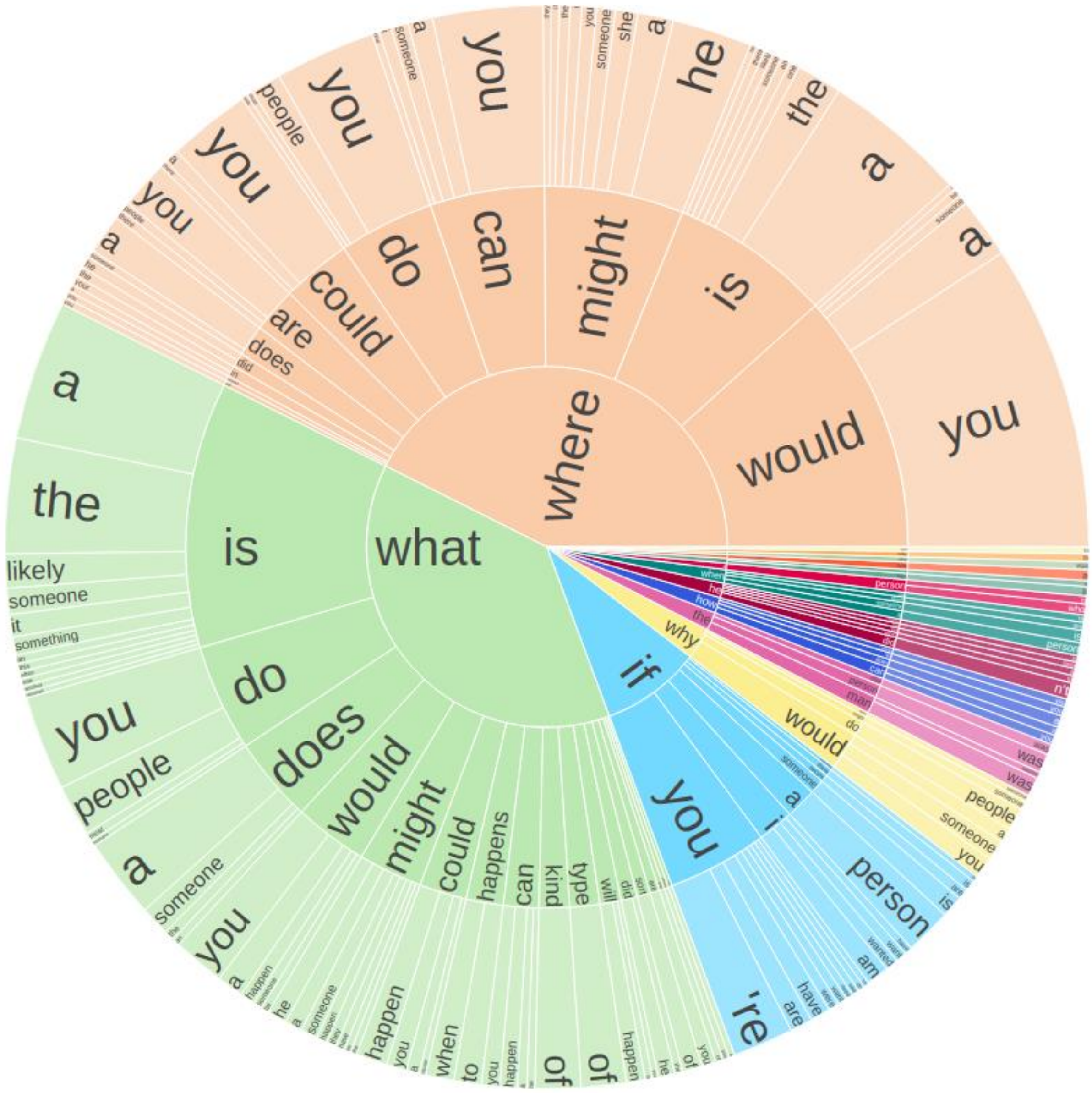
RACE



DROP

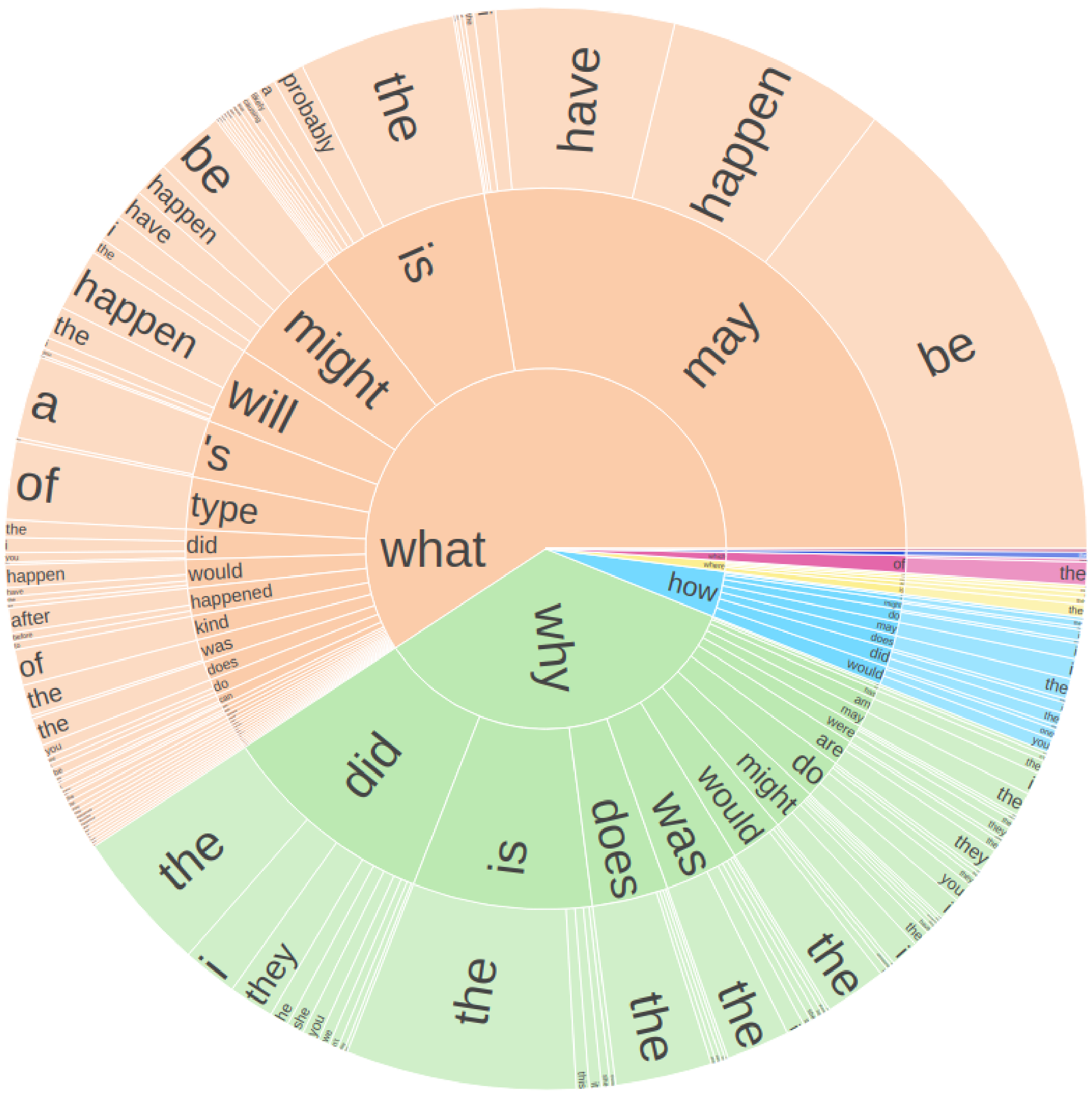

how
many
yards
more
years
points
field
percent
total
touchdowns
in
months
days
people
touchdown
of
times
games
which
player
team
group
players
age
happened
scored
had
threw
kicked
caught
scored
won
from
is
scored
group
first
who
scored
threw
caught
had
kicked
the
the
the
the
the
what
was
happened
the
second
first
more

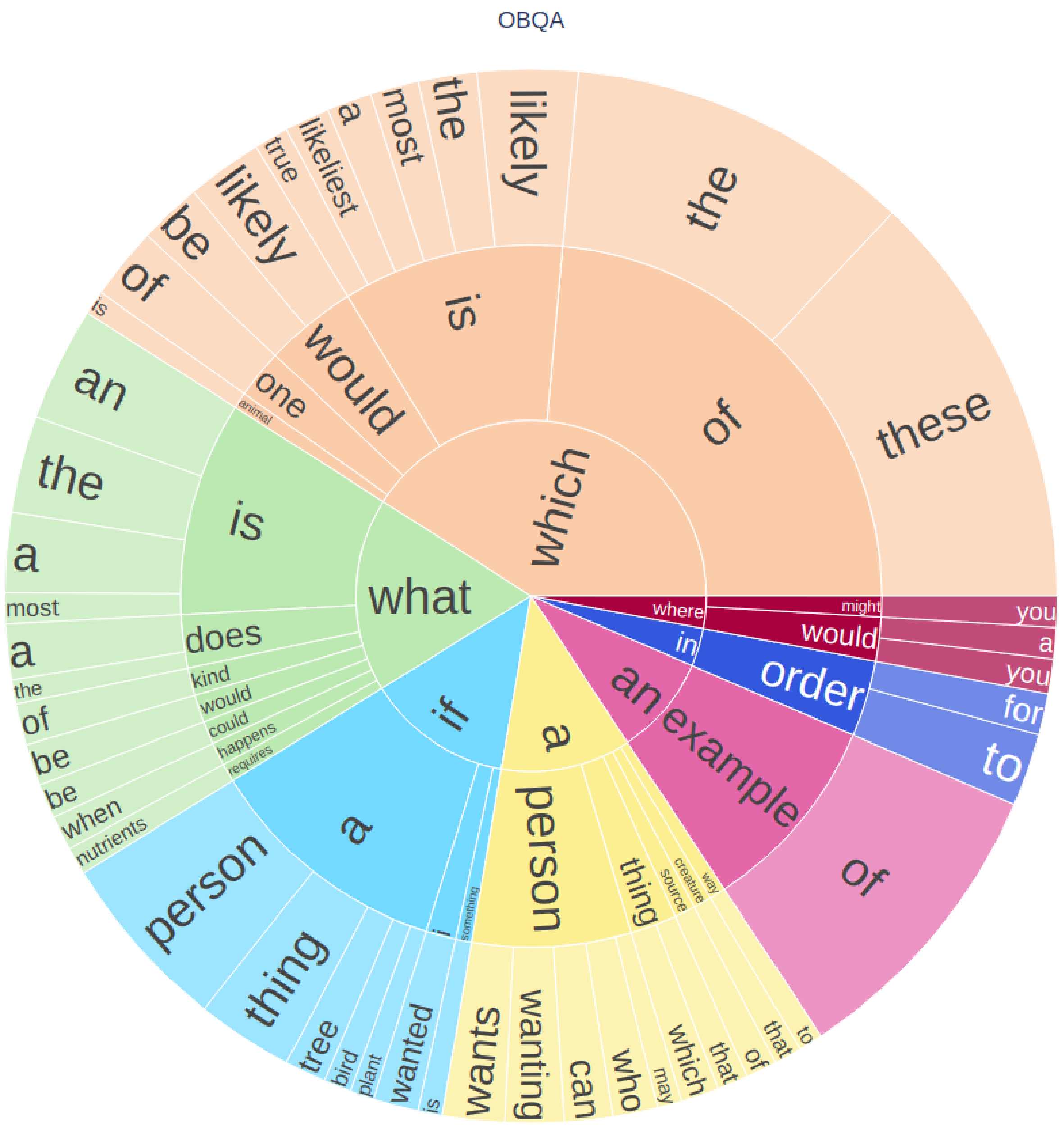
CSQA



what
where
if
why
how
the
he
when
is
do
does
would
might
could
happens
can
kind
type
will
did
are
might
is
would
you
a
the
likely
someone
it
something
people
he
she
're
person
have
am
wanted

CosmosQA


what
may
be
happen
have
is
the
probably
might
be
happen
have
will
happen
the
's
a
type
of
did
would
happened
kind
was
does
do
after
why
did
the
they
he
she
you
is
the
does
the
was
the
would
might
do
are
were
how
would
did
the
they
you
of
the


OBQA
which
of
these
the
is
likely
the
most
a
likeliest
true
would
likely
be
of
one
animal
is
what
is
an
the
a
most
does
a
the
kind
of
would
be
could
be
happens
when
requires
nutrients
if
a
person
thing
tree
bird
plant
wanted
is
something
a
person
wants
wanting
can
who
may
which
that
thing
source
creature
way
of
that
to
an example
of
in order
to
for
you
where
would
might
a
you

ARC

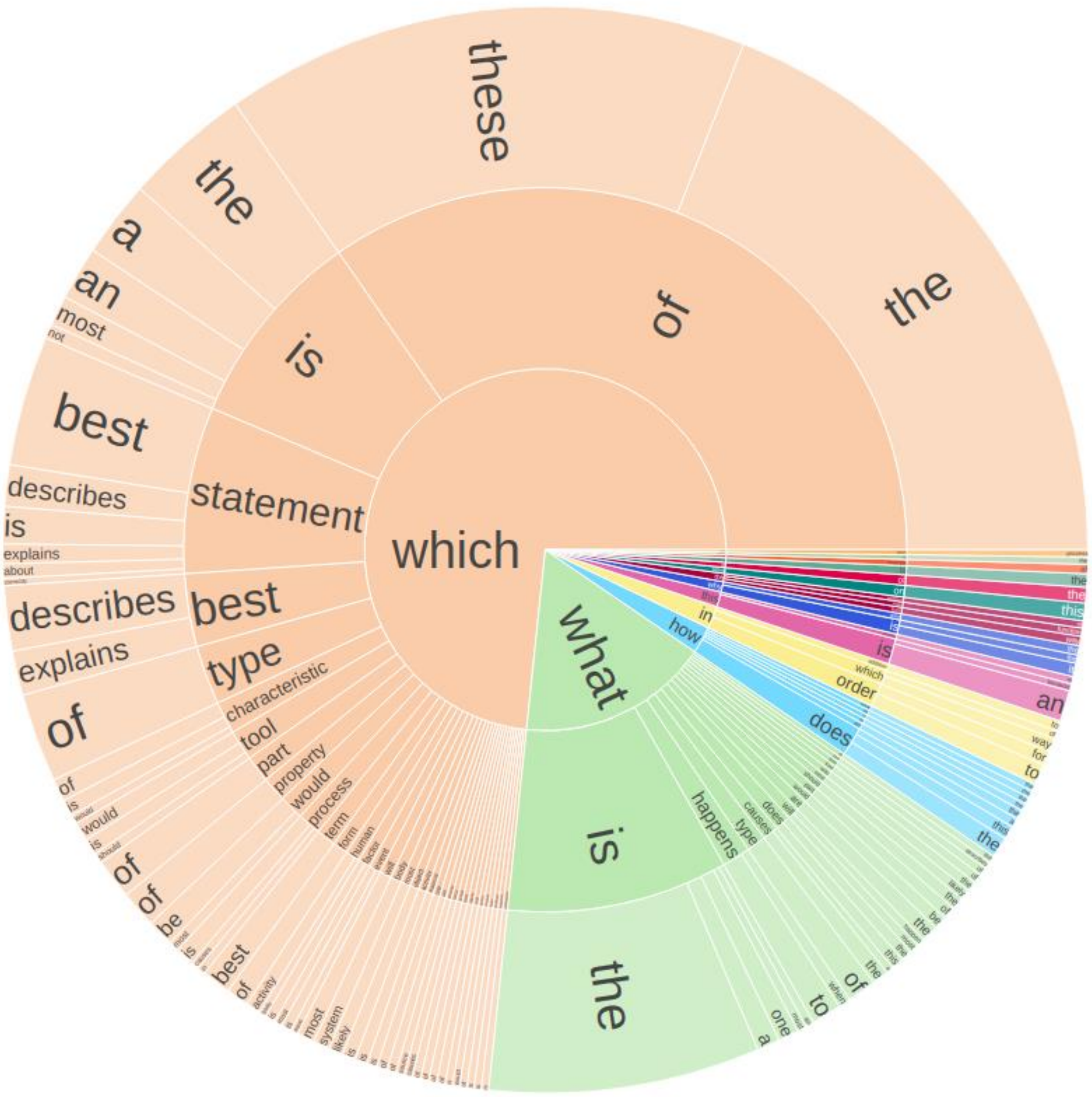

which
of
the
these
is
the
a
an
most
statement
best
describes
is
explains
best
describes
explains
type
of
characteristic
tool
part
property
would
process
what
is
the
how
in
does
order
happens
causes
type
to
of
one
an
the
this

Table 43

Most frequent starting trigrams in questions, per dataset

Dataset	**Most frequent question start trigrams**
OBQA	1) ('which', 'of', 'these') - 122 times 2) ('which', 'of', 'the') - 100 times 3) ('an', 'example', 'of') - 87 times 4) ('if', 'a', 'person') - 52 times 5) ('what', 'is', 'an') - 33 times
RACE	1) ('which', 'of', 'the') - 9183 times 2) ('according', 'to', 'the') - 4760 times 3) ('what', 'is', 'the') - 3405 times 4) ('what', "'s", 'the') - 1962 times 5) ('what', 'can', 'we') - 1813 times
DROP	1) ('how', 'many', 'yards') - 8740 times 2) ('how', 'many', 'more') - 6167 times 3) ('how', 'many', 'years') - 5613 times 4) ('how', 'many', 'points') - 4536 times 5) ('how', 'many', 'field') - 3660 times
CSQA	1) ('where', 'would', 'you') - 423 times 2) ('where', 'is', 'a') - 198 times 3) ('what', 'is', 'a') - 193 times 4) ('what', 'is', 'the') - 159 times 5) ('where', 'can', 'you') - 152 times
CosmosQA	1) ('what', 'may', 'be') - 3981 times 2) ('why', 'is', 'the') - 1877 times 3) ('what', 'may', 'happen') - 1793 times 4) ('what', 'may', 'have') - 1454 times 5) ('what', 'is', 'the') - 1280 times
ARC	1) ('which', 'of', 'the') - 620 times 2) ('which', 'of', 'these') - 509 times 3) ('what', 'is', 'the') - 263 times 4) ('which', 'statement', 'best') - 126 times 5) ('which', 'is', 'the') - 122 times

Appendix B

Examples of retrieved facts for OpenBookQA using the vector-based approach:

Question: An example of camouflage is when an organism looks like what?

A) clouds
B) local flora
C) buildings
D) oceans

Similar facts:

1) An example of camouflage is when an organism looks like its environment
2) An example of camouflage is an organism looking like leaves
3) An example of camouflage is when something has the same color as its environment
4) An example of camouflage is when something is the same color as its environment
5) An example of camouflage is when something changes color in order to have the same color as its environment
6) metamorphosis is a stage in the life cycle process of some animals
7) disguise means change appearance to hide
8) a solution is made of one substance dissolved in another substance
9) an ecosystem contains nonliving things
10) hibernation is used for conserving resources by some animals

Question: How does darkness impact photosynthesis?

A) positively
B) increases absorption
C) very poorly
D) increases endurance

Similar facts:

1) darkness has a negative impact on photosynthesis
2) cloudy means the presence of clouds in the sky
3) as distance from a source of light increases, that source of light will appear dimmer
4) the phases of the Moon change the appearance of the Moon
5) cutting down trees has a negative impact on an organisms living in an ecosystem
6) as the amount of rain increases in an environment, available sunlight will decrease in that environment
7) extreme temperatures cause an organism 's energy levels to decrease
8) a plant requires photosynthesis to grow
9) as light pollution increases, seeing the stars will be harder
10) clouds produce rain

Appendix C

Examples of retrieved facts for OpenBookQA using the token-based approach:

Question: Pushing a little kid on a swing is easy because they are light. It will require more strength to push when

A) when he loses weight
B) when he eats chicken
C) seventy-four plus five
D) he's older and heavier

Query: Pushing a little kid on a swing is easy because they are light. It will require more strength to push when

1) pushing an object requires force
2) as the mass of an object increases, the force required to push that object will increase
3) pushing a button sometimes completes a circuit
4) flowing liquid can push objects
5) if a flexible container is pushed on then that container will change shape
6) pushing on the pedals of a bike causes that bike to move
7) tectonic plates being pushed together causes earthquakes
8) a squid produces thrust by pushing water out of its body
9) if an object reflects more light then that object is more easily seen
10) a rocket engine is used to produce thrust by pushing gases out at a high speed

Question: Which is a pollinating creature?

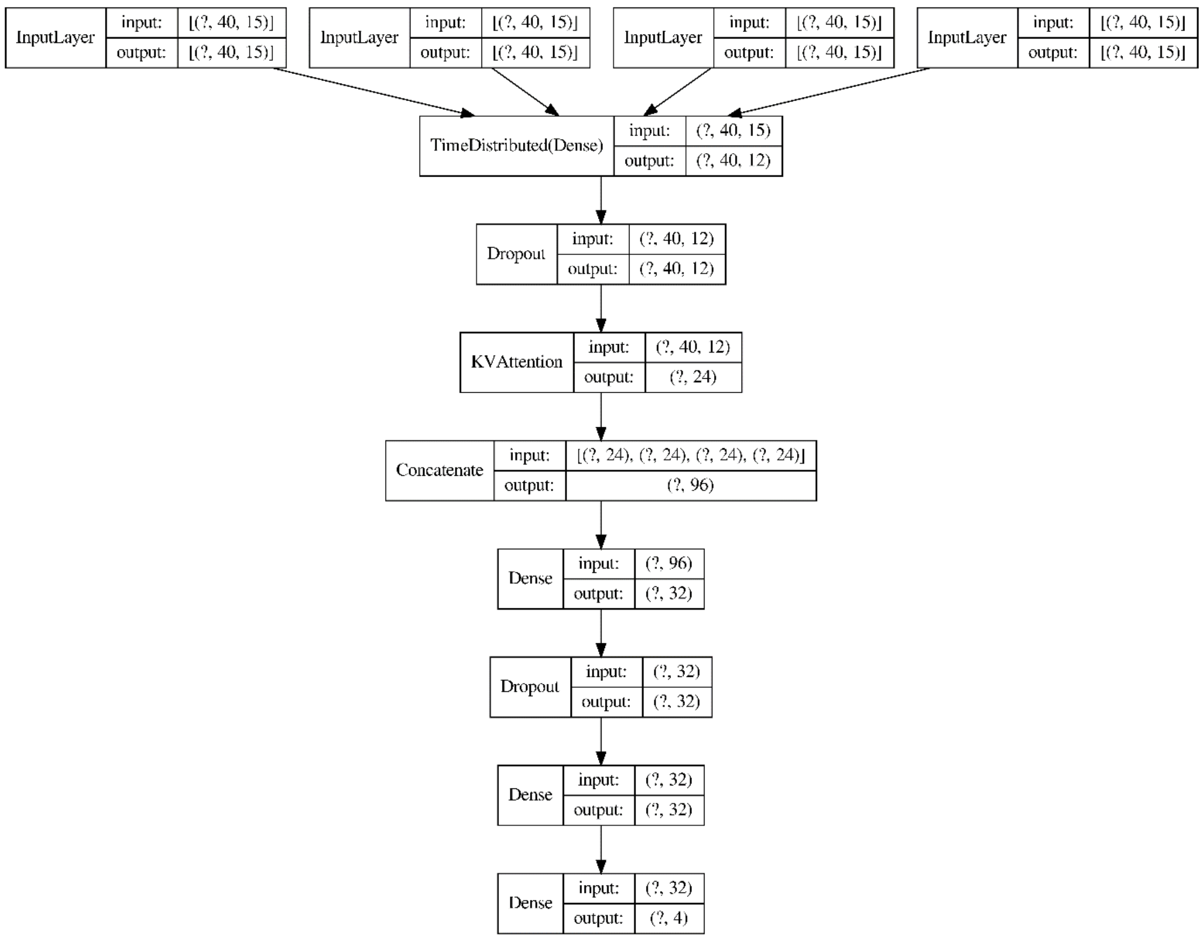
A) hyenas
B) lions
C) honey makers
D) rhinos

Query: Which is a pollinating creature?

1) pollination requires pollinating animals
2) A bee is a pollinating animal
3) An insect is a pollinating animal
4) A bird is a pollinating animal
5) seed production requires pollination
6) plant reproduction requires pollination
7) the type of material through which sound passes changes the speed at which sound travels
8) nectar is used for attracting pollinators by plants
9) melting point means temperature at which a solid melts
10) a planet is exposed to the heat of the star around which it revolves

Appendix D

The combiner architecture used by ZeroQA (exactly as in the Attentive Ranker [62], same layers, same hyperparameters). The network is responsible for transferring the acquired knowledge from the base datasets to the downstream dataset. The network has 4 inputs, a tensor for each answer. The tensor represents the scores predicted by models trained on the base datasets and maybe other scores.



The output is a probability distribution over the 4 possible candidate answers. Notice that the neural network is symmetric with respects to the inputs (we can permute them, and the scores will be permuted in the same way).

Appendix E

ARC questions that are similar to questions in the base datasets (TF-IDF is the similarity proxy):

a) Most similar ARC questions to RACE:

ARC: “A lunar eclipse occurs when the Moon passes through Earth's shadow. A lunar eclipse can only occur during a”
RACE: “When a total lunar eclipse occurs, _ .”

ARC: “Sydney is learning about the sources of energy that can run out. Which energy source will run out the fastest?”
RACE: “The sun is an endless source of energy, and it will not run out of energy for _ .”

ARC: “Earth orbits the Sun once a year. About how many times does the moon orbit Earth in a year?”
RACE: “It takes Earth _ to orbit the sun in one year”

ARC: “Today, almost all cars have seat belts. How does improving the design of seat belts help people the most?”
RACE: “Seat belts _”

ARC: “When atmospheric carbon dioxide increases, some of the carbon dioxide dissolves in the ocean causing the ocean to become”
RACE: “Which of the following is NOT the result of oceans absorbing heat and carbon dioxide?”

b) Most similar ARC questions to DROP:

ARC: “Mrs. Moyers teaches her students about different energy sources. Which example comes from a renewable source of energy?”
DROP: “What is Ireland's best source of renewable energy?”

ARC: “Mars is an inner planet and Jupiter is an outer planet. Which best describes the relationship between the two planets?”
DROP: “Which stars have planets that are more massive than Jupiter?”

ARC: “Planets in our solar system have different solar years. Which statement explains the cause of an Earth solar year?”
DROP: “How many days in a solar year date?”

ARC: “The Sun is classified as a star. Which characteristic identifies the Sun as a star?”
DROP: “Which stars are sun-like?”

ARC: “Compared to other stars in our galaxy, which is the best description of our Sun?”
DROP: “Which star mentioned are smaller than our sun?”

c) Most similar ARC questions to OpenBookQA:

ARC: “If two objects are two meters apart, which of these changes will increase the gravitational force between the two objects?”
OBQA: “the gravitational pull between two objects increases as they are”

ARC: “Mammals must eliminate waste products that their bodies produce. Which organ helps mammals eliminate bodily waste?”
OBQA: “Elimination of their body's waste is”

ARC: “Cells in the body use oxygen (O2) for cellular respiration. Which is the result of cellular respiration?”
OBQA: “When a cell takes in in oxygen an use cellular respiration it will then expire”

ARC: “Students combined baking soda and vinegar to demonstrate a chemical reaction. What indicates that a chemical reaction occurred?”
OBQA: “A chemical reaction to vinegar or to baking soda can be caused by adding”

ARC: “Electrical energy may sometimes be converted to mechanical energy. Which of these appliances converts electrical energy into mechanical energy?”
OBQA: “a battery converts chemical energy into electrical energy to power a”

d) Most similar ARC questions to CSQA:

ARC: “Why is it winter in North America when it is summer in South America?”
CSQA: “South America is in winter when North America is in summer, this is because it is located where?”

ARC: “A light bulb transforms electrical energy into light energy. A light bulb also transforms electrical energy into”
CSQA: “Where are light bulbs frequently found?”

ARC: “The Moon revolves around the Earth because of a non-contact force. What force of Earth keeps the Moon in orbit?”
CSQA: “What force keeps objects on the surface of the earth?”

ARC: “Why is it better to wear a white T-shirt than a dark blue T-shirt in the summer?”
CSQA: “Where might a t-shirt be kept?”

ARC: “Salt and pepper are placed together in a container. When the container is shaken, the salt and pepper become a”
CSQA: “You'd add pepper and salt to what liquid meal if it's bland?”

e) Most similar ARC questions to CosmosQA:

ARC: “When ice cream is left out of a freezer, the ice cream changes from a ___.”
CosmosQA: “Why was it good that there was Ice Cream in the freezer?”

ARC: “Planets in our solar system have different solar years. Which statement explains the cause of an Earth solar year?”
CosmosQA: “What may happen before three of our solar system’s planets are discovered?”

ARC: “Electrical energy may sometimes be converted to mechanical energy. Which of these appliances converts electrical energy into mechanical energy?”
CosmosQA: “What was zapping my energy and mood?”

ARC: “A light bulb transforms electrical energy into light energy. A light bulb also transforms electrical energy into”
CosmosQA: “Why did he remove the light bulbs from the flat?”

ARC: “Young frogs do not resemble adult frogs. Which term is given to this pattern of development in frogs?”
CosmosQA: “What will happen to the frog?”