Extractive Question Answering with Soft Structural biases

Urchade Zaratiana[*†], **Nadi Tomeh**[†], **Pierre Holat**[*†], **Thierry Charnois**[†]
[*] FI Group, [†] LIPN, CNRS UMR 7030, France
{urchade.zaratiana,pierre.holah}@fi-group.com
{charnois,tomeh}@lipn.fr

Abstract

Extractive Question Answering (ExQA) is an NLP task that aims to accurately identify the location of the answer within a given document, given a question. It is typically achieved by independently predicting the start and end positions of the answer. While simple, it outperforms more structured approaches, such as jointly predicting the start and end positions, when working with large-scale data. However, when data is limited, the joint objective, which has a stronger inductive bias, proves to be more effective, but at the cost of increased computational complexity. We propose a new ExQA approach that combines the efficiency of independent prediction with the advantages of the joint objective by allowing for soft interactions between start and end positions. Our experiments show that our approach outperforms existing baselines in both high and low data settings.

1 Introduction

Extractive QA is an important task of Natural Language processing with numerous practical applications. Given a question and the document containing the answer, the objective of this task is to identify the precise positions of the start and end of the answer span within the document. In the past, extractive QA methods have relied on highly engineered architectures (Seo et al., 2016; Hu et al., 2017; Wang et al., 2017), primarily utilizing bidirectional long short-term memory networks and attention mechanism (Bahdanau et al., 2015). However, with the advent of BERT (Devlin et al., 2019) and other pretrained transformer-based language models, the field has undergone a significant transformation. Upon its release, BERT set new standards for QA performance, significantly outperforming the previous best method on a variety of benchmarks. This was achieved through a relatively straightforward approach, in which the model first computes word representations and then independently predicts the start and end positions of the answer by computing start and end logits through linear projection. This approach has become the most widely used method for extractive QA (Liu et al., 2019; Joshi et al., 2020; He et al., 2021) due to its strong performance and computational efficiency, which is linear with respect to the length of the input.

The independent objective function used in BERT has proven to be effective in QA performance. However, it may be sub-optimal as it does not take into account the structure of the output. A joint objective, which considers the interaction between the start and end of the answer may be a more effective approach. Our experimental results suggest that a joint objective greatly improves the sample efficiency of a QA model and considerably improves performance in scenarios where there is limited data available. However, when large amounts of data are employed, we found a joint objective does not necessarily improve results and can even harm performance, making it less favorable to use in scenarios where data size is scaled. Moreover, another disadvantage of the joint objective is that it has a quadratic space and time complexity in the length of the sequence, since the score all spans in the input have to be calculated for both training and inference.

As an alternative, in this paper, we propose an approach that keeps the advantages of the joint objective and independent objective while eliminating their weaknesses. To do so, we model the dependency of the start and end positions in a soft manner, without explicit constraints on the objective. Specifically, instead of strictly imposing a structure through a specialized objective function, we allow the model to softly choose the type of interaction it needs to produce the output, which we refer to as "*soft structural bias*". To accomplish this, we slightly modify the independent modeling approach so that the parameters that compute the

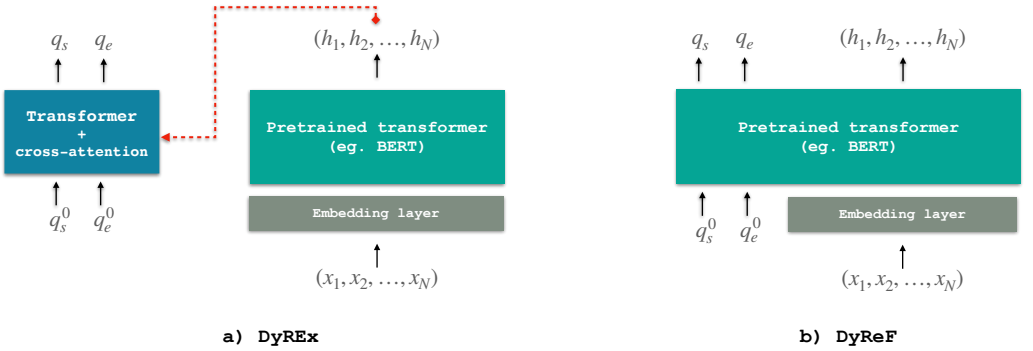

Figure 1: The figure illustrates the architecture of our proposed models. (a) *DyREx* uses an additional transformer layer to compute dynamic queries using self-attention and cross-attention. (b) *DyReF* computes dynamic representations without adding parameters by feeding initial queries and the input tokens to pretrained transformers.

start and end position logits (i.e. the queries q_s and q_e in Equation 1) can interact with each other through attention mechanisms. This enables the queries to decide the information they need from each other without explicit instructions, and even the possibility to ignore each other. Additionally, since the dependency of the start and end positions may vary depending on the input, we allow the query parameters to aggregate information from the input. Our approach thus keeps the linear complexity of the independent objective, while at the same time has good performance one few data. Furthermore, as we do not impose any structure (i.e we let the model decide itself), our approach also has greater performance when scaled to large data.

In this paper, we propose two variants of our proposed approach. The first variant, we called *DyREx* (**Dy**namic Query **R**epresentation for **ExQA**), computes the start and end interactions using a transformer layer with cross-attention. Specifically, the queries interact with each other using self-attention and the queries are made dependent on the input using cross-attention. While *DyREx* is computationally efficient (linear complexity with input size), it adds additional parameters (transformer layer) which may not be desirable. To address this, we propose a second variant called *DyReF* (**Dy**namic Query **Re**presentation for **F**ree), which computes the interactions in a "Free" manner without adding any parameters. This is achieved by feeding the initial query representation along with the input token embeddings into a pretrained transformer layer such as Bert, allowing the queries to interact with each others as well as with the input sequence, without any additional parameters. Both *DyREx* and *DyReF* are simple to incorporate into existing ExQA models, which mainly employ the Independent objective.

We conduct extensive experiments to demonstrate the superiority of our proposed approaches on a variety of benchmark extractive QA datasets. We found that our approach achieved significant improvements over the independent baseline, especially in the low-data regime. Additionally, we performed a series of ablation studies to investigate the important interactions by trying several attention masks (causal, bidirectional, independent). Our studies revealed two key findings: making the queries dependent on the input is highly beneficial, and making the queries interdependent further improves the results. Furthermore, by visualizing the attention maps, we found that the queries attend to three principal regions: they attend to each other, the region containing the question, and the region containing the answer positions.

The rest of this paper is organized as follows. In §(2) we provide the necessary background for understanding the baseline models. In §(3) we present our proposed models in details, followed by our experimental setup, results, and further experimental analysis in §(5). In §(8) we present an overview of related work. The last section concludes this paper.

2 Background

For all the models we present, we are given a set of input tokens $X = \{x_i\}_{i=1}^N$ which is the concatenation of the tokenized question Q and the passage P containing the answer. These tokens are encoded using a pretrained transformer language model, yielding a set of contextualized embeddings, $\boldsymbol{H} = \{\mathbf{h}_i\}_{i=1}^N \in \mathbb{R}^d$, d being the model dimension size. We now describe two models that we use as baselines: independent and a joint model.

Algorithm 1 PyTorch pseudocode for *DyREx*.

```
import torch

def dyrex(Bert, X, Q_0, Trans):
    """
    Bert: BERT model
    X: (Batch size, Length) input tokens
    Q_0: (1, 2, Dim) initial query vectors
    Trans: Transformer layer with cross-attention
    """
    # Token representation
    # (Batch size, Length, Dim)
    H = Bert(input_ids=X)

    # Repeat queries along batch axis
    # (Batch size, 2, Dim)
    Q_0 = Q_0.repeat(B, 1, 1)

    # Dynamic queries by cross-attention
    # (Batch size, 2, Dim)
    Q = Trans(query=Q_0, key=H, value=H)
    return Q
```

Algorithm 2 PyTorch pseudocode for *DyReF*.

```
import torch

def dyref(Bert, X, Q_0):
    """
    Bert: BERT model
    X: (Batch size, Length) input tokens
    Q_0: (1, 2, Dim) initial query vectors
    """
    # Token embeddings (Batch size, Length, Dim)
    H_0 = Bert.embedding_layer(input_ids=X)

    # Repeat queries along batch axis
    # (Batch size, 2, Dim)
    Q_0 = Q_0.repeat(B, 1, 1)

    # Concat queries and token embeddings
    # (Batch size, Length+2, Dim)
    C_0 = torch.cat((Q_0, H_0), dim=1)
    C = Bert.transformer_layers(input_embs=C_0)

    # final queries and token representations
    Q, H = C[:, :2], C[:, 2:]
    return Q, H
```

2.1 Independent Modeling

The independent model predicts the start ($s \in 1 \dots N$) and end ($e \in 1 \dots N$) positions using the following estimator:

$$p(s = i|X) = \frac{\exp(\mathbf{q}_s^T \mathbf{h}_i)}{\sum_{i'=1}^{N} \exp(\mathbf{q}_s^T \mathbf{h}_{i'})}$$
$$p(e = j|X) = \frac{\exp(\mathbf{q}_e^T \mathbf{h}_j)}{\sum_{j'=1}^{N} \exp(\mathbf{q}_e^T \mathbf{h}_{j'})} \quad (1)$$

Where $\mathbf{q}_s$ and $\mathbf{q}_e \in \mathbb{R}^d$ are respectively the *start* and *end queries* that are randomly initialized and updated during training. The loss function is:

$$\mathcal{L}_{indep.} = -\log\left(p(s = i|X)p(e = j|X)\right)$$

Due to its simplicity and its linear complexity in the input sequence length, this approach is the most widely used in the literature (Devlin et al., 2019; Joshi et al., 2020; Yasunaga et al., 2022).

2.2 Joint Modeling

The joint model predicts the span probability directly, leveraging the idea that jointly estimating both the start and end positions is more expressive in terms of the range of distributions it can represent, and is less prone to label bias. To accomplish this, we utilize a model of the following form:

$$p(s = i, e = j|X) = \frac{\exp(\phi(\mathbf{h}_i, \mathbf{h}_j))}{\sum_{(i',j') \in \mathcal{S}} \exp(\phi(\mathbf{h}_{i'}, \mathbf{h}_{j'}))} \quad (2)$$

In the above equation, $\mathcal{S}$ denotes the set of all spans of the input sequence X, $\phi(\mathbf{h}_i, \mathbf{h}_j) \in \mathbb{R}$ represents the joint score of token i being the start and token j being the end of the answer. We adopt the following scoring function:

$$\phi(\mathbf{h}_i, \mathbf{h}_j) = \mathbf{q}_s^T \mathbf{h}_i + \mathbf{q}_e^T \mathbf{h}_j + \mathbf{h}_i^T \mathbf{W} \mathbf{h}_j \quad (3)$$

This scoring function incorporates both independent start and end scores ($\mathbf{q}_s^T \mathbf{h}_i$ and $\mathbf{q}_e^T \mathbf{h}_j$) as well as an interaction score modeled through a bilinear layer, represented as $\mathbf{h}_i^T \mathbf{W} \mathbf{h}_j$, where $\mathbf{W} \in \mathbb{R}^{d \times d}$ is a learned parameter of the operator. It is noteworthy that independent modeling is mathematically equivalent to using $\phi(\mathbf{h}_i, \mathbf{h}_j) = \mathbf{q}_s^T \mathbf{h}_i + \mathbf{q}_e^T \mathbf{h}_j$, as it only accounts for local scores. The loss function employed in this study is:

$$\mathcal{L}_{joint} = -\log p(s = i, e = j|X)$$

This loss function aims to maximize the probability of the gold span. Due to the pairwise interaction score, both computation of this loss and the inference process have quadratic time and space complexity with respect to the input length.

3 Dynamic Query Modeling

The main motivation behind the models we introduce in this section is to combines the efficiency and flexibility of independent approaches with inherent structural biases of present in the joint model. Furthermore, the joint model which scores interactions between start and end in the output layer may negatively impact performance on large datasets as our experiments show. Instead, we propose a dynamic representation approach that models the interdependence between start and end queries in a soft manner, enabling the simulation of various structural biases. For this purpose we employ attention mechanisms known for their expressive power and performance. We present two models based on this idea: *DyREx* and *DyReF*.

3.1 *DyREx*

Our first approach for incorporating structural bias to the ExQA model learns $\mathbf{q}_s$ and $\mathbf{q}_e$ dynamically in the context of a given Q and P. We begin by initializing the start and end query representations, $\mathbf{q}_s^0$ and $\mathbf{q}_e^0$, which are concatenated and passed through a transformer layer to obtain dynamic representations $\mathbf{q}_s$ and $\mathbf{q}_e$:

$$\mathbf{Q} = \texttt{Trans}(\mathbf{Q}^0, \mathbf{H}) \tag{4}$$

with $\mathbf{Q}^i = [\mathbf{q}_e^i, \mathbf{q}_s^i]$ the concactenated queries at layer i and $\mathbf{H}$ the input token representations, and `Trans` being a K-layers transformer layer with cross-attention. More specifically, the i-th layer of the transformer consists of a bi-directional self-attention module $\texttt{self-att}_i$ applied between the queries to model the interdependence between the start and the end positions of the answer, a cross-attention $\texttt{cross-att}_i$ which updates the query representations by aggregating information from the input sequence embeddings so that the queries become dependent on the input, and a two-layer point-wise feedforward network $\texttt{FFN}_i$ with GeLU activation (Hendrycks and Gimpel, 2016):

$$\begin{aligned}
\tilde{\mathbf{Q}}^i &= \texttt{self-att}_i(\texttt{Q} = \mathbf{Q}^i, \texttt{K} = \mathbf{Q}^i, \texttt{V} = \mathbf{Q}^i) \\
\widehat{\mathbf{Q}}^i &= \texttt{cross-att}_i(\texttt{Q} = \tilde{\mathbf{Q}}^i, \texttt{K} = \mathbf{H}, \texttt{V} = \mathbf{H}) \\
\mathbf{Q}^{i+1} &= \texttt{FNN}_i(\widehat{\mathbf{Q}}^i)
\end{aligned} \tag{5}$$

Furthermore, an `Add-Norm` (skip connection (He et al., 2016) plus layer normalization (Ba et al., 2016)) are inserted after each of the components as in Vaswani et al. (2017), but we do not show it here for better readability. Moreover, both `self-att` and `cross-att` layers are multi-head scaled dot-product attention from Vaswani et al. (2017), and the embedding dimension and the number of attention heads of the decoder layers are the same as for the token representation layer. The design of *DyREx* is depicted in Figure 1, and its Pytorch implementation is provided in Algorithm 1.

3.2 *DyReF*

In practice, *DyREx* is compuatationally efficient and provides strong results. However, it adds non-negligible number of parameters with the addition of K-Layer transformer which might be undesirable. Motivated by this, we introduce *DyReF*, which retains the same performance as *DyREx* without adding any parameter to the initial model. To do so, *DyReF* first concatenates the initial queries $\mathbf{Q}^0$ to the output of the pretrained transformer (e.g. BERT) *embedding layer* $\mathbf{H}^0$. They are then passed to the transformer layers of the model to obtain the representations of the queries and the input tokens:

$$[\mathbf{Q}, \mathbf{H}] = \texttt{BERT}([\mathbf{Q}^0, \mathbf{H}^0]) \tag{6}$$

Where $\mathbf{Q} = [\mathbf{q}_s, \mathbf{q}_e]$ are the dynamic queries of start and end positions obtained from the model. The architecture of *DyReF* is illustrated in Figure 1, and its corresponding Pytorch pseudo-code can be found in Algorithm 2.

3.3 Modeling with *DyREx* and *DyReF*

For both *DyREx* and *DyReF*, to compute the start and the end answer position probabilities, we use the same estimator and objective function as the independent model in Equation (1). This choice is jusrified by the fact that interactions between start and end are taken into account in the query representations in *DyREx* and *DyReF*, therefore there is no need for further joint modeling. To validate this hypothesis, we integrate *DyREx* and *DyReF* into the joint model in Equation (2) and compare both models. Our experiments confirm that no further improvements are obtained by this combination.

4 Experimental Setup

4.1 Data

In our experiments, we use several widely-used English extractive question-answering datasets, including SQuAD (Rajpurkar et al., 2016), HotpotQA (Yang et al., 2018), NewsQA (Trischler et al., 2016), TriviaQA (Joshi et al., 2017), and Natural Questions (Kwiatkowski et al., 2019). These datasets are sourced from various sources, such as Wikipedia articles, news articles, and web snippets. Each dataset consists of a triple of "Question, Passage, and Answer", and the task is to predict the precise position of the answer (start and end positions) within the passage given the question. The dataset statistics are provided in Table 1 of the MRQA shared task paper (Fisch et al., 2019)[1].

4.2 Hyperparameters

In this study, we used standard hyperparameter configurations commonly found in the literature to ensure reproducibility. To achieve

[1] https://arxiv.org/abs/1910.09753

Train size	Models	Datasets					
		SQuAD	HotpotQA	TriviaQA	NewsQA	NaturalQs	Average
256	*Indep.*	65.74	53.23	28.49	35.80	41.87	45.03
	+DyREx	70.75	**57.08**	**41.66**	43.77	45.57	**51.77**
	+DyReF	71.08	55.08	40.53	44.56	**46.91**	51.63
	Joint	69.55	53.42	43.48	44.79	43.33	50.91
	+DyREx	70.01	55.52	29.17	43.58	44.85	48.63
	+DyReF	**72.25**	46.19	41.01	**46.45**	45.7	50.32
512	*Indep.*	69.72	58.70	45.39	43.24	48.36	53.08
	+DyREx	77.19	61.15	52.57	49.20	**55.37**	59.10
	+DyReF	**77.24**	62.27	52.59	**52.26**	54.51	**59.77**
	Joint	76.18	61.87	54.24	49.18	53.94	59.08
	+DyREx	76.54	**63.88**	52.15	49.79	53.24	59.12
	+DyReF	75.03	57.08	**55.77**	51.53	53.4	58.56
1024	*Indep.*	74.01	62.54	51.87	50.61	53.42	58.49
	+DyREx	79.42	**67.95**	57.59	54.26	**61.67**	64.18
	+DyReF	**80.35**	66.40	56.77	**56.71**	61.15	**64.28**
	Joint	79.12	65.27	58.31	55.0	59.84	63.51
	+DyREx	79.17	66.47	56.74	54.68	59.72	63.36
	+DyReF	79.36	64.97	**58.38**	55.55	61.45	63.94
Full	*Indep.*	90.64	79.95	76.31	68.02	77.79	78.54
	+DyREx	91.01	80.55	77.37	**68.53**	**78.58**	**79.21**
	+DyReF	**91.04**	80.55	**77.39**	68.27	78.44	79.12
	Joint	90.60	80.12	75.52	67.07	77.61	78.18
	+DyREx	91.04	**80.62**	75.52	67.97	77.66	78.56
	+DyReF	90.60	80.25	76.92	67.97	77.75	78.70

Table 1: **Main results**. We present experimental results using SpanBert (Joshi et al., 2020) for token representation, on diverse datasets with varying sizes from 256 to full data.

this, we made use of the default hyperparameter configuration provided by the HuggingFace Transformers library (Wolf et al., 2020) (for the Independent model, we employed the `AutoModelForQuestionAnswering` class from the library). We used the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 3e-5, and employed a warm-up stage for the first 10% of the training steps, and then linearly decreased the learning rate for the remainder of the training steps. The batch size was set to 12, and we trained for a maximum of 5 epochs for full-sized datasets. For few-shot settings, we trained the models for a maximum of either 2500 steps or 10 epochs. As the *DyReX* variant of our model adds transformer layers for computing the dynamic queries. We found that using three layers transformer provide a good efficiency-performance trade-off. We use the PyTorch framework (Paszke et al., 2019) to implement our models and loaded pre-trained models from the Transformers library (Wolf et al., 2020). The training of all models was conducted on a server equipped with V100 GPUs. The total GPU hours required for all experiments described in this paper, as well as additional preliminary experiments, amounted to approximately 2500 hours.

5 Results

The results of our experiments are reported in Table 1. Our experiments were conducted using a variety of datasets, spanning a range of data sizes from 256 training examples to full datasets. For both the independent and the joint models, we report the performance results with and without *DyREx* and *DyReF*.

5.1 Independent vs Joint

In this section, we investigate the comparative performance of independent and joint modeling approaches. Our results indicate that, in cases of limited data, the joint model exhibits a superior performance, as demonstrated by the +6 point increase in F1 score on the SQuAD dataset when using only 512 training examples. This finding highlights the utility of incorporating more inductive bias in data-scarce scenarios. On the other hand, when utilizing full datasets, the independent model emerges as the optimal choice on average. This phenomenon aligns with established knowledge in the field of machine learning, which posits that stronger inductive bias improves sample efficiency, yet assuming less knowledge may be more advantageous at larger scales - as observed with convolutional and transformer models for image literature (Dosovitskiy et al., 2020; Touvron et al., 2020; d'Ascoli et al., 2021).

5.2 Soft structural models

In the current sub-section, we examine the performance of joint and independent models when augmented with the use of *DyREx* and *DyReF* as soft structural biases.

Independent with soft structural bias Our results, as presented in the accompanying table, demonstrate that the integration of *DyREx* and *DyReF* with the independent model significantly enhances its performance when compared to the independent model alone. In this configuration, the performance of *DyREx* and *DyReF* is comparable to that of the joint model, and even surpasses it on average. Similarly, when using full datasets, the independent model augmented with either *DyREx* or *DyReF* yields the best performance. Notably, while the performance of the joint model plateaus with the use of full datasets, *DyREx* and *DyReF* are able to further improve the performance of the independent model.

Joint with soft structural bias In contrast, when considering the joint model with the integration of *DyREx* and *DyReF* as soft structural biases, our results reveal that the performance improvement observed with the independent model is not replicated. Specifically, the addition of *DyREx* and *DyReF* does not necessarily lead to the performance improvement of the joint model. This may be attributed to the potential redundancy or conflicting nature of the information provided by the joint model and the structural biases, as the integration of *DyREx* or *DyReF* in some instances even negatively impact the performance of the joint model (which is never happening with the independent model).

	Training	**Inference**
Indep.	42.44	136.11
+DyREx	40.12	132.97
+DyReF	41.25	133.60
Joint	36.88	102.43
+DyREx	33.20	100.06
+DyReF	35.57	101.11

Table 2: Training and Inference speed throughput in samples per second. The test was done using a V100 32G GPUs.

5.3 Efficiency analysis

In this sub-section, we conduct an efficiency analysis of the models in question. The table 2 reports the training and inference throughput of the models, measured in number of samples per second. For this study, we run the models using a V100 GPU with 32GB of memory. As can be observed, the independent model exhibits a significant advantage in terms of efficiency compared to the joint model. Furthermore, we also note that the incorporation of *DyREx* and *DyReF* does not significantly impact the overall efficiency of the models, as evidenced by the minor reductions in the throughput of the independent and joint models with *DyREx* and *DyReF* respectively.

6 Attention analysis

The attention mechanism is a crucial aspect in our work, as it enables the queries to interact with each other in a meaningful way. In this section, we conduct a thorough analysis of the attention mechanism to gain a better understanding of the model. To do this, we undertake two studies: 1) In the first study, we evaluate different attention masks to identify the most relevant interactions between the queries and the input sequence. 2) In the second study, we examine the attention map to determine which areas of the input the queries pay the most attention to, which can aid in understanding the model's decision-making process

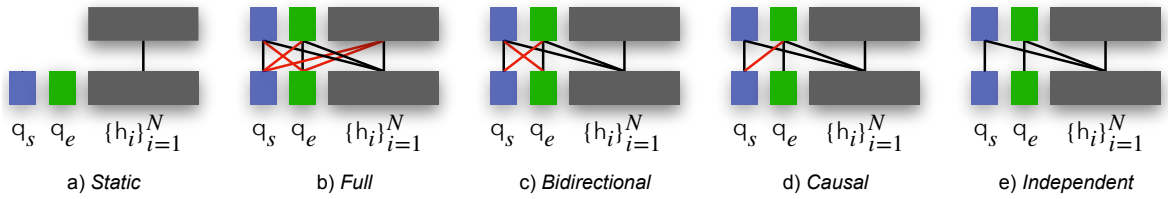

Figure 2: **Different attention masking for `DyReF`**. a) *Static* is the one employed by the *Indep.* ExQA. b) *Full* is a setting where both all the queries and all the tokens attend to each other. c) *Bidirectional* allow full interaction between the queries. d) *Causal* is the same as the bidirectional but start query does not attend to the end query. e) *Independent* the queries only depend on the input sequence but not attending to each other.

6.1 Attention mask variants

Motivation Our proposed models (*DyREx* and *DyReF*) use attention mechanism to allow queries to interact with each other, enabling them to model dependencies the start and end of anwer. This improves the ability to answer questions by aggregating relevant information through the Transformer layers. Here, We investigate the importance of different attention masks to better understand the impact of different interactions on model performance.

Masking strategies We evaluate four variants of our model, shown in Figure 2. The figure also displays the *Indep.* model using static queries and the *Full* model that allows full interaction between queries and tokens, exclusive to *DyReF*. All variants use dynamic queries but with different attention masks between queries: *Bidirectional* allows full attention between queries; *Causal* allows end query to attend start query but not vice versa; and *Independent* completely masks attention between queries making them independent of each other.

Results Table 3 shows that all queries (full, bidirectional and causal) perform similarly for *DyREx* and *DyReF*, the difference is only marginal. However, in all the settings, having independent queries always perform worse, indicating the benefit of keeping attention between start and end queries. Futhermore,

6.2 Attention visualization

In addition to the attention mask study, we visualize the attention scores of the transformer layer to gain insight into the model's interactions with the input sequence. The visualization is shown in Figure 3. It was generated using a trained QA model on the SQuAD dataset, using SpanBert and the *DyReF* approach.

`self-att`	EM	F1
a) Static (*indep.*)	83.08 ± 0.09	90.64 ± 0.10
DyREx		
c) *Bidirectional*	83.35 ± 0.07	$\mathbf{91.01 \pm 0.03}$
d) *Causal*	$\mathbf{83.47 \pm 0.17}$	91.00 ± 0.05
e) *Independent*	83.14 ± 0.44	90.87 ± 0.09
DyReF		
b) *Full*	$\mathbf{83.43 \pm 0.18}$	$\mathbf{91.04 \pm 0.04}$
c) *Bidirectional*	83.37 ± 0.26	$\mathbf{91.04 \pm 0.05}$
d) *Causal*	$\mathbf{83.43 \pm 0.07}$	90.99 ± 0.07
e) *Independent*	83.03 ± 0.08	90.88 ± 0.01

Table 3: **Different masking variants for the self-attention layers**. This study is performed on SQuAD dataset using SpanBert for token representation. Results are averaged across three random seeds.

Method To ease interpretation, we aggregate all the attention matrices from SpanBert's 12 layers and 12 attention heads into a single one by applying 2D max pooling over the head and layer dimensions of the stacked attention matrices. This results in a single attention matrix of dimension $(2+N) \times (2+N)$, where N is the length of the input sequence and 2 is the number of queries. We visualize only the query-query and query-token attention and drop token-token interaction, resulting in an attention map of $2 \times (2+N)$ in Figure 3.

Analysis The attention visualization is shown in Figure 3. We used two different inputs to test the model's consistency. Both inputs share the same passage but have different questions and answers. We observe that the queries attend to three main regions: each other, the region containing the question, and the region containing the answer positions. This demonstrate that the queries already have knowledge of the answer span position before making the final prediction, which may explain the effectiveness of these approaches.

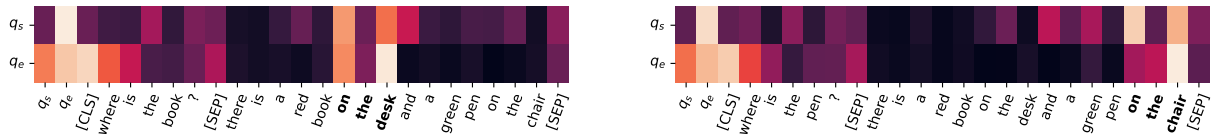

Figure 3: **Attention visualization**. Figure displays tokens attended to by start and end queries ($\mathbf{q}_s$ and $\mathbf{q}_e$) in a trained QA model on SQuAD using SpanBert. Attention maps created by elementwise-max of attention values across 12 attention heads and 12 layers. Two examples, using the same passage but different questions, are provided to demonstrate attention consistency. Tokens of answer spans are in boldface. The first example (left) is from Cui et al. (2022).

7 Related Works

Extractive QA Extractive question answering (ExQA) is a widely-studied NLP task with various real-world applications. Advancements in this field have been driven by the availability of large-scale annotated datasets such as SQuAD (Rajpurkar et al., 2016). Early ExQA models, such as BidAF (Seo et al., 2016), Match-LSTM (Hu et al., 2017), and QaNet (Yu et al., 2018), among others, were highly specialized and required complex engineering. The introduction of BERT (Devlin et al., 2019) revolutionized the ExQA field by introducing a simple yet effective approach: concatenating the input question and text passage and computing the start and end span answer using learned query parameters, referred to as *Indep.* ExQA in this paper. Since its introduction, this approach has remained the dominant approach for extractive QA (Liu et al., 2019; Yang et al., 2019; Joshi et al., 2020; Yamada et al., 2020; He et al., 2021; Yasunaga et al., 2022). In this paper, we extend the *Indep.* ExQA by utilizing dynamic query representation while maintaining the same number of parameters and a negligible computational cost. Furthermore, frameworks such as Splinter (Ram et al., 2021) and ReasonBERT (Deng et al., 2021) have recently produced state-of-the-art results on extractive QA, but their main contribution is the introduction of effective pretext tasks for improving ExQA models. These models continue to use the *Indep.* approach for fine-tuning, thus they are orthogonal to our work and can be used in conjunction with it to further improve the results. Similar to us, (Fajcik et al., 2021) have explored the use of different objective the extractive QA. However, they analysis focus on scenario with large data.

Inductive bias Recent research has challenged the notion that increased inductive bias is always beneficial. For instance, recent developments in deep learning have shown that the CNN architecture, which incorporates a strong inductive bias for image processing tasks, has been surpassed in performance by transformer models, which make fewer assumptions about the data (Dosovitskiy et al., 2020; Touvron et al., 2020).

It is worth noting that while hard inductive biases can act as a limitation, incorporating soft inductive biases can lead to improved performance. A recent example of this approach is the ConViT (d'Ascoli et al., 2021) model, which integrates the expressivity of transformers with the convolutional biases through a self-attention mechanism that incorporates a soft convolutional inductive bias. The resulting model has been shown to exhibit superior performance and data efficiency, similar to the findings of our paper.

8 Conclusion

In this paper, we investigated the use of various objective functions for extractive QA, specifically the independent and joint objectives. Our findings indicate that utilizing a strong inductive bias (joint objective) can lead to improved performance when data is limited. However, this advantage over the more efficient independent objective is diminished when large data sets are available. To address this limitation, we proposed the *DyREx* and *DyReF* models, which allow for flexible bias stimulation in a soft manner. Our experiments demonstrated that these models achieve competitive results in both high and low data settings. In future work, we plan to extend these models to other tasks, such as named entity recognition, where multiple span extraction is required.

Limitations

While our study provides empirical observations that a model with less structural knowledge performs better when the data size is scaled, we recognize that a robust theoretical grounding to explain

this phenomenon is not within the scope of this paper. Our focus was to establish these empirical patterns and relationships, offering a launching pad for more in-depth analysis in the future. However, rather than viewing this as a limitation, we posit it as an exciting opportunity for further exploration. Future research can dig deeper into understanding the underlying mechanisms that result in these findings, further extending and validating the empirical observations made in our study.

References

Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *ArXiv*, abs/1607.06450.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Yiming Cui, Wei-Nan Zhang, Wanxiang Che, Ting Liu, Zhigang Chen, and Shijin Wang. 2022. Multilingual multi-aspect explainability analyses on machine reading comprehension models. *iScience*, 25(5):104176.

Stéphane d'Ascoli, Hugo Touvron, Matthew L. Leavitt, Ari S. Morcos, Giulio Biroli, and Levent Sagun. 2021. Convit: improving vision transformers with soft convolutional inductive biases. *Journal of Statistical Mechanics: Theory and Experiment*, 2022.

Xiang Deng, Yu Su, Alyssa Lees, You Wu, Cong Yu, and Huan Sun. 2021. Reasonbert: Pre-trained to reason with distant supervision. In *EMNLP*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *ArXiv*, abs/2010.11929.

Martin Fajcik, Josef Jon, Santosh Kesiraju, and Pavel Smrz. 2021. Rethinking the objectives of extractive question answering. *ArXiv*, abs/2008.12804.

Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. 2019. MRQA 2019 shared task: Evaluating generalization in reading comprehension. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 1–13, Hong Kong, China. Association for Computational Linguistics.

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention. *ArXiv*, abs/2006.03654.

Dan Hendrycks and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *ArXiv*, abs/1606.08415.

Annie Hu, Cindy Wang, and Brandon Yang. 2017. Question answering using match-lstm and answer pointer.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving Pre-training by Representing and Predicting Spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural Questions: A Benchmark for Question Answering Research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text.

Ori Ram, Yuval Kirstain, Jonathan Berant, Amir Globerson, and Omer Levy. 2021. Few-shot question answering by pretraining span selection.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herv'e J'egou. 2020. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. Newsqa: A machine comprehension dataset.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and M. Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *ACL*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2020. Transformers: State-of-the-art natural language processing. In *EMNLP*.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. Luke: Deep contextualized entity representations with entity-aware self-attention. In *EMNLP*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *CoRR*, abs/1809.09600.

Michihiro Yasunaga, Jure Leskovec, and Percy Liang. 2022. Linkbert: Pretraining language models with document links. *ArXiv*, abs/2203.15827.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension.