Abstract
The primary paradigm for multi-task training in natural language processing is to represent the input with a shared pre-trained language model, and add a small, thin network (head) per task. Given an input, a target head is the head that is selected for outputting the final prediction. In this work, we examine the behaviour of non-target heads, that is, the output of heads when given input that belongs to a different task than the one they were trained for. We find that non-target heads exhibit emergent behaviour, which may either explain the target task, or generalize beyond their original task. For example, in a numerical reasoning task, a span extraction head extracts from the input the arguments to a computation that results in a number generated by a target generative head. In addition, a summarization head that is trained with a target question answering head, outputs query-based summaries when given a question and a context from which the answer is to be extracted. This emergent behaviour suggests that multi-task training leads to non-trivial extrapolation of skills, which can be harnessed for interpretability and generalization.

1 Introduction
The typical framework for training a model in natural language processing to perform multiple tasks is to have a shared pre-trained language model (LM), and add a small, compact neural network, often termed head, on top of the LM, for each task (Clark et al., 2019; Liu et al., 2019b; Nishida et al., 2019; Hu and Singh, 2021). The heads are trained in a supervised manner, each on labelled data collected for the task it performs (Devlin et al., 2019). At inference time, the output is read out of a selected target head, while the outputs from the other heads are discarded (Figure 1).

What is the nature of predictions made by non-target heads given inputs directed to the target head? One extreme possibility is that the pre-trained LM identifies the underlying task, and constructs unrelated representations for each task. In this case, the output of the non-target head might be arbitrary, as the non-target head observes inputs considerably different from those it was trained on. Conversely, the pre-trained LM might create similar representations for all tasks, which can lead to meaningful interactions between the heads.

In this work, we test whether such interactions occur in multi-task transformer models, and if non-target heads decode useful information given inputs directed to the target head. We show that multi-head training leads to a steering effect, where the target head guides the behaviour of the non-target head, steering it to exhibit emergent behaviour, which can explain the target head’s predictions, or generalize beyond the task the non-target head was trained for.

We study the “steering effect” in three multi-head models (Figure 2). In a numerical reading comprehension task (Dua et al., 2019), the model is given a question and paragraph and either uses an extractive head to output an input span, or a generative head to generate a number using arithmetic...
operations over numbers in the input (Figure 2, left). Treating the extractive head as the non-target head, we observe that it tends to output the arguments to the arithmetic operation performed by the decoder, and that successful argument extraction is correlated with higher performance. Moreover, we perform interventions (Woodward, 2005; Elazar et al., 2021), where we modify the representation based on the output of the extractive head, and show this leads to predictable changes in the behaviour of the generative head. Thus, we can use the output of the non-target head to improve interpretability.

We observe a similar phenomenon in multi-hop question answering (QA) model (Yang et al., 2018), where a non-target span extraction head outputs supporting evidence for the answer predicted by a classification head (Figure 2, center). This emerging interpretability is considerably different from methods that explicitly train models to output explanations (Perez et al., 2019; Schuff et al., 2020).

Beyond interpretability, we observe non-trivial extrapolation of skills when performing multi-task training on extractive summarization (Hermann et al., 2015) and multi-hop QA (Figure 2, right). Specifically, a head trained for extractive summarization outputs supporting evidence for the answer when given a question and a paragraph, showing that multi-task training steers its behaviour towards query-based summarization. We show this does not happen in lieu of multi-task training.

To summarize, we investigate the behaviour of non-target heads in three multi-task transformer models, and find that without any dedicated training, non-target heads provide explanations for the predictions of target heads, and exhibit capabilities beyond the ones they were trained for. This extrapolation of skills can be harnessed for many applications. For example, teaching models new skills could be done by training on combinations of tasks different from the target task. This would be useful when labeled data is not available or hard to collect. Also, training an additional head that extracts information from the input could be applied as a general practice for model interpretability.

2 Multi-Head Transformer Models

The prevailing method for training models to perform NLP tasks is to add parameter-thin heads on top of a pre-trained LM, and fine-tune the entire network on labeled examples (Devlin et al., 2019).

Given a text input with \( n \) tokens \( x = (x_1, \ldots, x_n) \), the model first computes contextualized representations \( H = (h_1, \ldots, h_n) = L.M_{\theta}(x) \) using the pre-trained LM parameterized by \( \theta \). These representations are then fed into the output heads, with each head \( o \) estimating the probability \( p_{\psi_o}(y \mid H) \) of the true output \( y \) given the encoded input \( H \) and the parameters \( \psi_o \) of \( o \). The head that produces the final model output, termed the target head, is chosen either deterministically, based on the input task, or predicted by an output head classifier \( p(o \mid x) \). Predictions made by non-target heads are typically ignored. When \( p(o \mid x) \) is deterministic it can be viewed as an indicator...
function for the target head.

Training multi-head transformer models is done by marginalizing over the set of output heads $\mathcal{O}$, and maximizing the probability

$$ p(y \mid x) = \sum_{o \in \mathcal{O}} p(o \mid x) \cdot p(y \mid H, \psi_o), $$

where $p(y \mid H, \psi_o) > 0$ only if $y$ is in the output space of the head $o$.

For a head $o$, we denote by $S_o$ the set of examples $(x, y)$ such that $y$ is in the output space of $o$, and by $S_\ell$ the other training examples. The sets $S_o$ and $\hat{S}_o$ may consist of examples from different tasks (e.g., question answering and text summarization), or of examples from the same task but with different output formats (e.g., yes/no vs. span extraction questions). Our goal is to evaluate the predictions of $o$ on examples from $\hat{S}_o$, for which another head $o'$ is the target head, and the relation between these outputs and the predictions of $o'$.

In the next sections, we will show that the predictions of $o$ interact with those of $o'$. We will denote by $o_t$ the target head, and by $o_s$ the steered head.

3 Overview: Experiments & Findings

This section provides an overview of our experiments, which are discussed in detail in §4, §5, §6.

Given a model with a target head $o_t$ and a steered head $o_s$, it is important to understand the behaviour of $o_s$ on inputs where $o_t$ provides the prediction. To this end, we focus on head combinations, where $o_s$ is expressive enough to explain the outputs of $o_t$, but unlike most prior work aiming to explain by examining model outputs (Perez et al., 2019; Schuff et al., 2020; Wang et al., 2019), $o_s$ is not explicitly trained for this purpose. Concretely, our analysis covers three settings, illustrated in Figure 2 and summarized in Table 1.

The first setting (Figure 2 left, and §4) considers a model with generative and extractive heads, trained on the DROP dataset (Dua et al., 2019) for numerical reasoning over text. Surprisingly, we observe that the arguments for the arithmetic computation required for the generative head to generate its answer often emerge in the outputs of the extractive head. The second setting (Figure 2 middle, and §5) considers a model with a classification head outputting ‘yes’/’no’ answers, and a span extraction head, trained on the HOTPOTQA dataset (Yang et al., 2018) for multi-hop reasoning. The outputs of the extractive head once again provide explanations in the form of supporting facts from the input context. The last setting (Figure 2 right, and §6) considers a model with two extractive heads, one for span extraction and another for (sentence-level) extractive summarization. Each head is trained on a different dataset; HOTPOTQA for span extraction and CNN/DAILYMAIL for summarization (Hermann et al., 2015). We find that the summarization head tends to extract the supporting facts given inputs from HOTPOTQA, effectively acting as a query-based summarization model.

We now present the above settings. Table 1 summarizes the main results. We denote by $\text{FFNN}_{m \times n}$ a feed-forward neural network with $l$ layers that maps inputs of dimension $m$ to dimension $n$.

4 Setting 1: Emerging Computation Arguments in Span Extraction

We start by examining a combination of generative and extractive heads (Figure 2, left), and analyze the spans extracted from the input when the generative head is selected to output the final answer.

4.1 Experimental Setting

Model We take GenBERT (Geva et al., 2020), a BERT-base model fine-tuned for numerical reasoning, and use it to initialize a variant called MsGenBERT, in which the single-span extraction head is replaced by a multi-span extraction (MSE) head introduced by Segal et al. (2020), which allows extracting multiple spans from the input. This is important for supporting extraction of more than one argument. MsGenBERT has three output heads: The multi-span head, which takes $H \in \mathbb{R}^{d \times n}$, and uses the BIO scheme.

| Target head ($o_t$) | Steered head ($o_s$) | Dataset(s) | Emergent behaviour |
|---------------------|----------------------|------------|--------------------|
| Generation          | Multi-span extraction| DROP       | $o_s$ extracts numerical arguments for $o_t$ |
| Classification      | Single-span extraction| HOTPOTQA   | $o_s$ extracts supporting facts for $o_t$ |
| Single-span extraction| Extractive summarization| HOTPOTQA, CNN/DAILYMAIL | $o_s$ performs query-based summarization |

Table 1: A summary of the main findings in each of the settings investigated in this work.
Table 2: Evaluation results of M\textsc{seg}ENBERT. DRO\textsubscript{P} \text{F}_{1} and DRO\textsubscript{Span} \text{F}_{1} were computed on the DRO\textsubscript{P} development set and its subset of examples with span answers, respectively. All other scores are on the 400 annotated examples from DRO\textsubscript{P}. Correlation is between recall and F\textsubscript{1} scores. \(l\) refers to the number of linear layers in \(o_{\text{mse}}\).

Table 3: Annotated example from DRO\textsubscript{P}. Crowdsourcing was asked to extract the arguments (in bold) from the passage required to compute the answer.

Table 3: Annotated example from DRO\textsubscript{P}. Crowdsourcing was asked to extract the arguments (in bold) from the passage required to compute the answer. For arguments and spans which include a number as well as text, only the numeric sub-strings were considered when preforming the comparison between \(\mathcal{P}\) and \(\mathcal{G}\).

\textbf{Evaluation metrics} Given a list \(\mathcal{P}\) of extracted spans by \(o_{\text{mse}}\) and a list of annotated arguments \(\mathcal{G}\), we define the following metrics for evaluation:\footnote{For arguments and spans which include a number as well as text, only the numeric sub-strings were considered when preforming the comparison between \(\mathcal{P}\) and \(\mathcal{G}\).} We check argument recall by computing the fraction of arguments in \(\mathcal{G}\) that are also in \(\mathcal{P}\). We can then compute average recall over the dataset, and the proportion of questions with a perfect recall of 1.0 (first column in Table 2). Similarly, we compute precision by computing the fraction of arguments in \(\mathcal{P}\) that are also in \(\mathcal{G}\) and then the average precision over the dataset.

\textbf{4.2 Results} Table 2 presents the results. Comparing M\textsc{seg}ENBERT to M\textsc{se}BERT, where the model was trained without \(o_{\text{gen}}\) only on span extraction examples, we observe that multi-task training substantially changes the behaviour of the extractive head. First, M\textsc{seg}ENBERT dramatically improves the extraction of computation arguments: recall increases from 0.2→0.56, precision goes up from 0.32→0.6, and the fraction of questions with perfect recall reaches 0.41. The number of extracted spans also goes up to 2.1, despite the fact that most span extraction questions are a single span. The performance of M\textsc{se}BERT on span questions is similar to M\textsc{seg}ENBERT, showing that the difference is not explained by performance degradation. We

(Revised)
also measure the number of extracted spans on out-of-distribution math word problems, and observe similar patterns (details are in Appendix B.2).

Moreover, model performance, which depends on $o_{\text{gen}}$, is correlated with the recall of predicted spans, extracted by $o_{\text{mse}}$. The Spearman correlation between model $F_1$ and recall for MSE\textsc{gen}BERT is high at 0.351 (Table 2) and statistically significant (p-value $5.6e^{-13}$), showing that when the computation arguments are covered, performance is higher.

These findings illustrate that multi-task training leads to emergent behaviour in the extractive head, which outputs computation arguments for the output of the generative head. We now provide more fine-grained analysis.

**Distribution of extracted spans** On average, MSE\textsc{gen}BERT extracts 2.12 spans per example, which is similar to 1.95 spans extracted by annotators. Moreover, the average ratio $|p|/|w|$ is 1.2, indicating good correlation at the single-example level. Table 4 shows example outputs of $o_{\text{mse}}$ vs. the annotated arguments for the same questions. The full distributions of the number of extracted spans by MSE\textsc{gen}BERT compared to the annotated spans are provided in Appendix B.1.

**Parameter sharing across heads** We conjecture that the steering effect occurs when the heads are strongly tied, with most of their parameters shared. To examine this, we increase the capacity of the FFNN in $o_{\text{mse}}$ from $l = 1$ layer to $l = 2, 4$ layers, and also experiment with a decoder whose parameters, unlike GEN\textsc{bert}, are not tied to the encoder.

We find (Table 2) that reducing the dependence between the heads also diminishes the steering effect. While the models still tend to extract computation arguments, with much higher recall and precision compared to MSe\textsc{bert}, they output 1.2 spans on average, which is similar to the distribution they were trained on. This leads to higher precision, but much lower recall and fewer cases of perfect recall. Overall model performance is not affected by changing the capacity of the heads.

**Table 4**: Example outputs by $o_{\text{mse}}$ on DROP in comparison to the annotated computation arguments.

|       | Annotated arguments | Predicted arguments |
|-------|---------------------|---------------------|
| Full match | 26, 48              | 26, 48              |
| $o_{\text{mse}}$ missing | 31.7, 20.1          | 31.7                |
| $o_{\text{mse}}$ excessive | 1923, 1922          | 1923, 1937, 1922    |

4.3 Influence of Extracted Spans on Generation

The outputs of $o_{\text{mse}}$ and $o_{\text{gen}}$ are correlated, but can we somehow control the output of $o_{\text{gen}}$ by modifying the value of span tokens extracted by $o_{\text{mse}}$? To perform such intervention, we change the cross-attention mechanism in MSE\textsc{gen}BERT’s decoder. Typically, the keys and values are both the encoder representations $H$. To modify the values read by the decoder, we change the value matrix to $H_{i+j}$:

$$\text{MultiHeadAttention}(Q, H, H_{i+j})$$

where in $H_{i+j}$ the positions of the representations $h_i$ and $h_j$ are swapped (illustrated in Figure 3). Thus, when the decoder attends to the $i$'th token, it will get the value of the $j$'th token and vice versa.

We choose the tokens $i, j$ to swap based on the output of $o_{\text{mse}}$. Specifically, for every input token $x_k$ that is a digit, we compute the probability $p_k^\oplus$ by $o_{\text{mse}}$ that it is a beginning of an output span. Then, we choose the position $i = \arg \max_k p_k^\oplus$, and the position $j$ as a random position of a digit token. As a baseline, we employ the same procedure, but swap the positions $i, j$ of the two digit tokens with the highest outside ($\odot$) probabilities.

We focus on questions where MSE\textsc{gen}BERT predicted a numeric output. Table 5 shows in how many cases each intervention instance changed the model prediction (by $o_{\text{gen}}$). For 40.6% of the questions, the prediction was altered due to the intervention in the highest probability $\oplus$ token, compared to only 0.03% (2 cases) by the baseline intervention. This shows that selecting the token based on $o_{\text{mse}}$ affects whether this token will lead to a change.

More interestingly, we test whether we can predict the change in the output of $o_{\text{gen}}$ by looking at the two digits that were swapped. Let $d$ and $d'$ be the values of digits swapped, and let $n$ and $n'$ be the numeric outputs generated by $o_{\text{gen}}$ before and after the swap. We check whether $|n-n'| = |d-d'| \times 10^e$ for some integer $e$. For example, if we swap the

---

\textsuperscript{2}GEN\textsc{bert} uses digit tokenization for numbers.
digits ‘7’ and ‘9’, we expect the output to change by 2, 20, 0.2, etc. We find that in 543 cases out of 2,460 (22.1%) the change in the model output is indeed predictable in the non-baseline intervention, which is much higher than random guessing, that would yield 10%.

Last, we compare the model accuracy on predictable and unpredictable cases, when intervention is not applied to the examples. We observe that exact-match performance is 76% when the change is predictable, but only 69% when it is not. This suggests that interventions lead to predictable changes with higher probability when the model is correct.

Overall, our findings show that the spans extracted by \( o_{sse} \) affect the output of \( o_{gen} \), while spans \( o_{sse} \) marks as irrelevant do not affect the output. Moreover, the (relative) predictability of the output after swapping shows that the model performs the same computation, but with a different argument.

## 5 Setting 2: Emerging Supporting Facts in Span Extraction

We now consider a combination of an extractive head and a classification head (Figure 2, middle).

### 5.1 Experimental Setting

**Model**  We use the BERT-base \textsc{Reader} model introduced by Asai et al. (2020), which has two output heads: A single-span extraction head, which predicts for each token the probabilities for being the start and end position of the answer span:

\[
o_{sse} := FFN_{d \times 2}(H).
\]

The second head is a classification head for the answer type: yes, no, span, or no-answer:

\[
o_{type} := FFN_{d \times 1}(h_{CLS}).
\]

Implementation details are in Appendix A.2.

|       | B-swap changed | B-swap unchanged | O-swap changed | O-swap unchanged |
|-------|----------------|------------------|----------------|-----------------|
| 0.03  | 0.0            | 0.03             | 40.62          | 59.35           |
| 40.65 | 59.35          | 100              |                |                 |

Table 5: Intervention results on \( o_{gen} \) outputs. Percentage out of 6,051 DROP’s development examples for which swapping \( B \) (or \( O \)) tokens changed (or did not change) the output of \textsc{MseGenBERT}.

### Evaluation metrics
Let \( F \) be the set of annotated supporting facts per question and \( P \) be the top-\( k \) output spans of \( o_{sse} \), ordered by decreasing probability. We define Recall@\( k \) to be the proportion of supporting facts covered by the top-\( k \) predicted spans, where a fact is considered covered if a predicted span is within the supporting fact sentence and is not a single stop word (see Table 7).\(^3\)

We use \( k = 5 \) and report the fraction of questions where Recall@5 is 1 (Table 6, first column), to measure the cases where \( o_{sse} \) covers all relevant sentences in the first few predicted spans.

Additionally, we introduce an InverseMRR metric, based on the MRR measure, as a proxy for precision. We take the rank \( r \) of the first predicted span in \( P \) that is not a supporting fact from \( F \), and use \( 1 - \frac{1}{r} \) as the measure (e.g., if the rank of the first non overlapping span is 3, the reciprocal is 1/3 and the InverseMRR is 2/3). If the first predicted span is not in a supporting fact, InverseMRR is 0; if all spans for \( k = 5 \) overlap, InverseMRR is 1.

### Data
We fine-tune a \textsc{Reader} model on the gold paragraphs of \textsc{HotpotQA} (Yang et al., 2018), a dataset for multi-hop QA. Specifically, we feed the model question-context pairs and let it predict the answer type with \( o_{type} \). If \( o_{type} \) predicts span then the output by \( o_{sse} \) is taken as the final prediction, otherwise it is the output by \( o_{type} \). Therefore, \( o_{sse} \) is trained only on examples with an answer span.

Examples in \textsc{HotpotQA} are annotated with supporting facts, which are sentences from the context that provide evidence for the final answer. We use the supporting facts to evaluate the outputs of \( o_{sse} \) as explanations for questions where the gold answer is \textit{yes} or \textit{no}.

![Table 6: Evaluation results of \textsc{Reader} model on the gold paragraphs of \textsc{HotpotQA} (Yang et al., 2018), a dataset for multi-hop QA. Specifically, we feed the model question-context pairs and let it predict the answer type with \( o_{type} \). If \( o_{type} \) predicts span then the output by \( o_{sse} \) is taken as the final prediction, otherwise it is the output by \( o_{type} \). Therefore, \( o_{sse} \) is trained only on examples with an answer span. Examples in \textsc{HotpotQA} are annotated with supporting facts, which are sentences from the context that provide evidence for the final answer. We use the supporting facts to evaluate the outputs of \( o_{sse} \) as explanations for questions where the gold answer is \textit{yes} or \textit{no}.

\[\begin{array}{cccc}
\text{Data} & \text{Model} & \text{Evaluation metrics} & \text{Results} \\
\hline
\text{DROP} & \text{BERT} & \text{Recall@5} & 0.605 \text{ (1)} \\
\text{DROP} & \text{BERT} & \text{MRR} & 0.867 \\
\text{DROP} & \text{BERT} & \text{F1} & 72.9 \\
\end{array}\]

\(^3\)We do not define coverage as the fraction of tokens in a supporting fact that the span covers, because supporting facts are at the sentence-level, and often times most of the tokens in the supporting fact are irrelevant for the answer.
Table 7: Example questions from HOTPOTQA and the top-5 spans extracted by the READER model.

5.2 Results

Results are presented in Table 6. Comparing READER and READERonly sse, the Recall@5 and InverseMRR scores are substantially higher when using multi-task training, with an increase of 10.9% and 4.5%, respectively, showing again that multi-task training is the key factor for emerging explanations. Example questions with the spans extracted by READER are provided in Table 7.

As in §4, adding an additional layer to $o_{ssse}$ (READER$_{l=2}$) decreases the frequency of questions with perfect Recall@5 (0.605 → 0.568). This shows again that reducing the dependency between the heads also reduces the steering effect. It is notable that the performance on HOTPOTQA is similar across the different models, with only a slight deterioration when training only the extraction head ($o_{ssse}$). This is expected as READERonly sse is not trained with yes/no questions, which make up a small fraction of HOTPOTQA.

6 Setting 3: Emerging Query-based Summaries

In §4 and §5, we considered models with output heads trained on examples from the same data distribution. Would the steering effect occur when output heads are trained on different datasets? We now consider a model trained to summarize text and answer multi-hop questions (Figure 2, right).

6.1 Experimental Setting

Model We create a model called READERSUM as follows: We take the READER model from §5, and add the classification head presented by Liu and Lapata (2019), that summarizes an input text by selecting sentences from it. Sentence selection is done by inserting a special [CLS] token before each sentence and training a summarization head to predict a score for each such [CLS] token from the representation $h_{CLS}$:

$$o_{sum} := FFNN_{d×1}(h_{CLS}).$$

The sentences are ranked by their scores and the top-3 highest score sentences are taken as the summary (top-3 because choosing the first 3 sentences of a document is a standard baseline in extractive summarization (Nallapati et al., 2017; Liu and Lapata, 2019)). Implementation details are in A.3.

Data The QA heads ($o_{ssse}$, $o_{type}$) are trained on HOTPOTQA, while the summarization head is trained on the CNN/DAILYMAIL dataset for extractive summarization (Hermann et al., 2015). We use the supporting facts from HOTPOTQA to evaluate the outputs of $o_{sum}$ as explanations for predictions of the QA heads.

Evaluation metrics Annotated supporting facts and the summary are defined by sentences from the input context. Therefore, given a set $T$ of sentences extracted by $o_{sum}$ ($|T| = 3$) and the set of supporting facts $F$, we compute the Recall@3 of $T$ against $F$.

6.2 Results

Results are summarized in Table 8. When given HOTPOTQA examples, READERSUM extracts summaries that cover a large fraction of the supporting facts (0.79 Recall@3). This is much higher compared to a model that is trained only on the extractive summarization task (READERSUM$_{only sum}$ with 0.69 Recall@3). Results on CNN/DAILYMAIL show that this behaviour in READERSUM does not stem from an overall improvement in extractive summarization, as READERSUM performance is slightly lower compared to READERSUM$_{only sum}$. 

|               | Recall@3 | F1   | ROUGE 1/2 |
|---------------|----------|------|-----------|
| READERSUM    | 0.79     | 71.6 | 42.7/19.2 |
| READERSUM$_{only sum}$ | 0.69     | -    | 43.6/19.9 |
| RANDOM       | 0.53     | -    | 32.1/10.9 |
| LEAD3        | 0.60     | -    | 40.6/17.1 |
| READERSUM$_{masked}$ | 0.66     | -    | 42.7/19.2 |

Table 8: Evaluation results of READERSUM. Recall@3 and F1 scores were computed over the development set of HOTPOTQA, and ROUGE over CNN/DAILYMAIL.
To validate this against other baselines, both ReaderSum and ReaderSum only sum achieved substantially better Recall@3 scores compared to a baseline that extracts three random sentences from the context (RANDOM with 0.53 Recall@3), and summaries generated by taking the first three sentences of the context (LEAD3 with 0.6 Recall@3).

Overall, the results show multi-head training endows $o_{sum}$ with an emergent behavior of query-based summarization, which we evaluate next. Example summaries extracted by ReaderSum for HOTPOTQA are provided in Appendix C.

**Influence of questions on predicted summaries**

We run ReaderSum on examples from HOTPOTQA while masking out the questions, thus, $o_{sum}$ observes only the context sentences. As shown in Table 8 (ReaderSum masked), masking the question leads to a substantial decrease of 13 Recall@3 points in comparison to the same model without masking (0.79→0.66).

Since our model appends a [CLS] token to every sentence, including the question (which never appears in the summary), we can rank the question sentence based on the score of $o_{sum}$. Computing the rank distribution of question sentences, we see (Figure 4) that the distributions of ReaderSum only sum and ReaderSum are significantly different, and that questions are ranked higher in ReaderSum. This shows that the summarization head puts higher emphasis on the question in the multi-head setup.

Overall, these results provide evidence that multi-head training pushes $o_{sum}$ to perform query-based summarization on inputs from HOTPOTQA.

**7 Related Work**

Transformer models with multiple output heads have been widely employed in previous works (Hu and Singh, 2021; Aghajanyan et al., 2021; Segal et al., 2020; Hu et al., 2019; Clark et al., 2019). To the best of our knowledge, this is the first work that analyzes the outputs of the non-target heads.

Previous work used additional output heads to generate explanations for model predictions (Perez et al., 2019; Schuff et al., 2020; Wang et al., 2019). Specifically, recent work has explored utilization of summarization modules for explainable QA (Nishida et al., 2019; Deng et al., 2020). In the context of summarization, Xu and Lapata (2020) have leveraged QA resources for training query-based summarization models. Hierarchies between NLP tasks have also been explored in multi-task models not based on transformers (Segaard and Goldberg, 2016; Hashimoto et al., 2017; Swayamdipta et al., 2018). Contrary to previous work, the models in this work were not trained to perform the desired behaviour. Instead, explanations and generalized behaviour emerged from training on multiple tasks.

A related line of research has focused on developing probes, which are supervised network modules that predict properties from model representations (Conneau et al., 2018; van Aken et al., 2019; Tenney et al., 2019; Liu et al., 2019a). A key challenge with probes is determining whether the information exists in the representation or is learned during probing (Hewitt and Liang, 2019; Tamkin et al., 2020). Unlike probes, steered heads are trained in parallel to target heads rather than on a fixed model. Moreover, steered heads are not designed to decode specific properties from representations, but their behaviour naturally extends beyond their training objective.

Our findings also relate to explainability methods that highlight parts from the input via the model’s attention (Wiegreffe and Pinter, 2019), and extract rationales through unsupervised training (Lei et al., 2016). The emerging explanations we observe are based on the predictions of a head rather than on internal representations.

**8 Conclusions and Discussion**

We show that training multiple heads on top of a pre-trained language model creates a steering effect, where the target head influences the behaviour of another head, steering it towards capabilities beyond its training objective. In three multi-task settings, we find that without any dedicated training, the steered head often outputs explanations for the model predictions. Moreover, modifying the
input representation based on the outputs of the steered head can lead to predictable changes in the target head predictions.

Our findings provide evidence for extrapolation of skills as a consequence of multi-task training, opening the door to new research directions in interpretability and generalization. Future work could explore additional head combinations, in order to teach models new skills that can be cast as an extrapolation of existing tasks. In addition, the information decoding behaviour observed in this work can serve as basis for developing general interpretability methods for debugging model predictions.

A natural question that arises is what head combinations lead to a meaningful steering effect. We argue that there are two considerations involved in answering this question. First, the relation between the tasks the heads are trained on. The tasks should complement each other (e.g. summarization and question answering), or the outputs of one task should be expressive enough to explain the outputs of the other task, when applied to inputs of the other task. For example, extractive heads are particularly useful when the model’s output is a function of multiple input spans. Another consideration is the inputs to the heads. We expect that training heads with similar inputs (in terms of length, language-style, etc.) will make the underlying language model construct similar representations, thus, increasing the probability of a steering effect between the heads.

Acknowledgements

We thank Ana Marasović and Daniel Khashabi for the helpful feedback and constructive suggestions, and the NLP group at Tel Aviv University, particularly Maor Ivgi and Elad Segal. This research was supported in part by The Yandex Initiative for Machine Learning, and The European Research Council (ERC) under the European Union Horizons 2020 research and innovation programme (grant ERC DELPHI 802800). This work was completed in partial fulfillment for the Ph.D degree of Mor Geva.

References

Armen Aghajanyan, Anchit Gupta, Akshat Shrivas-tava, Xilun Chen, Luke Zettlemoyer, and Sonal Gupta. 2021. Muppet: Massive multi-task rep-
resentations with pre-finetuning. arXiv preprint arXiv:2101.11038.

Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. 2020. Learning to retrieve reasoning paths over wikipedia graph for question answering. In International Conference on Learning Representations.

Kevin Clark, Minh-Thang Luong, Urvashi Khandelwal, Christopher D. Manning, and Quoc V. Le. 2019. BAM! born-again multi-task networks for natural language understanding. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 5931–5937, Florence, Italy. Association for Computational Linguistics.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#^*$ vector: Probing sentence embeddings for linguistic properties. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Yang Deng, Wai Lam, Yuexiang Xie, Daoyuan Chen, Yaliang Li, Min Yang, and Ying Shen. 2020. Joint learning of answer selection and answer summary generation in community question answering. Proceedings of the AAAI Conference on Artificial Intelligence, 34(05):7651–7658.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In North American Association for Computational Linguistics (NAACL), pages 4171–4186, Minneapolis, Minnesota.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In North American Chapter of the Association for Computational Linguistics (NAACL).

Yanai Elazar, Shauli Ravfogel, Alon Jacovi, and Yoav Goldberg. 2021. Amnesic Probing: Behavioral Explanation with Amnesic Counterfactuals. Transactions of the Association for Computational Linguistics, 9:160–175.

Mor Geva, Ankit Gupta, and Jonathan Berant. 2020. Injecting numerical reasoning skills into language models. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 946–958, Online. Association for Computational Linguistics.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsu-ruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple NLP tasks. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing,
Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espenholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In Advances in Neural Information Processing Systems, volume 28. Curran Associates, Inc.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 2733–2743, Hong Kong, China. Association for Computational Linguistics.

Minghao Hu, Yuxing Peng, Zhen Huang, and Dongsheng Li. 2019. A multi-type multi-span network for reading comprehension that requires discrete reasoning. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 1596–1606, Hong Kong, China. Association for Computational Linguistics.

Ronghang Hu and Amanpreet Singh. 2021. Transformer is all you need: Multimodal multitask learning with a unified transformer. arXiv preprint arXiv:2102.10772.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hammaneh Hajishirzi. 2016. MAWPS: A math word problem repository. In Association for Computational Linguistics (ACL), pages 1152–1157, San Diego, California.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 107–117, Austin, Texas. Association for Computational Linguistics.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019a. Linguistic knowledge and transferability of contextual representations. 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 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019b. Multi-task deep neural networks for natural language understanding. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 4487–4496, Florence, Italy. Association for Computational Linguistics.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 3730–3740, Hong Kong, China. Association for Computational Linguistics.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In Proceedings of the AAAI Conference on Artificial Intelligence.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gülçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

Kosuke Nishida, Kyosuke Nishida, Masaaki Nagata, Atsushi Otsuka, Itsumu Saito, Hisako Asano, and Junji Tomita. 2019. Answering while summarizing: Multi-task learning for multi-hop QA with evidence extraction. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 2335–2345, Florence, Italy. Association for Computational Linguistics.

Ethan Perez, Siddharth Karamcheti, Rob Fergus, Jason Weston, Douwe Kiela, and Kyunghyun Cho. 2019. Finding generalizable evidence by learning to convince Q&A models. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 2402–2411, Hong Kong, China. Association for Computational Linguistics.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pages 101–108, Online. Association for Computational Linguistics.

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In Third Workshop on Very Large Corpora.

Hendrik Schuff, Heike Adel, and Ngoc Thang Vu. 2020. Fl is not Enough! Models and Evaluation Towards User-Centered Explainable Question Answering. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 7076–7095, Online. Association for Computational Linguistics.

Elad Segal, Avia Efrat, Mor Shoham, Amir Globerson, and Jonathan Berant. 2020. A simple and effective model for answering multi-span questions. In
A Implementation Details

A.1 Models Trained on DROP

We implement MSEGENBERT by taking GENBERT\(^3\) (Geva et al., 2020) and replacing its span-extraction head with the tagging-based multi-span extraction head\(^6\) by Segal et al. (2020).

All the variants of MSEGENBERT were initialized with the checkpoint of GENBERT\(_{ND+TD}\), that was fine-tuned on both numerical and textual data. For fine-tuning on DROP, we used the same hyperparameters used in Geva et al. (2020), specifically, a learning rate of \(3e^{-5}\) with linear warmup 0.1 and weight decay 0.01 for 30 epochs, and a batch size of 16.

A.2 Models Trained on HOTPOTQA Alone

To train READER models on HOTPOTQA, we used the official code\(^7\) by Asai et al. (2020). We fine-tuned the models on the gold paragraphs (without the distractor paragraphs) for 2 epochs with a learning rate of \(5e^{-5}\) and a batch size 16. All the other hyperparameters remained the same as in the implementation of Asai et al. (2020).

A.3 Models Trained on HOTPOTQA and CNN/DAILYMAIL

To train a model for both question answering and summarization, we used the official code\(^7\) by Asai et al. (2020), but adapted it to serve also for the extractive summarization task, by adding the classification head of BERTSUM \(^8\) (Liu and Lapata, 2019). BERTSUM summarizes an input context by selecting sentences (see §6). To allow this mechanism, we modify the inputs from HOTPOTQA and CNN/DAILYMAIL as follows: First, we split the context into sentences using Stanza (Qi et al., 2020). Then, a special [CLS] token is added at the beginning of each sentence and a [SEP] token is added at the end of it.

The model is fine-tuned for each task by training on one batch from each task at a time, i.e. a batch of HOTPOTQA followed by a batch of CNN/DAILYMAIL. To obtain the oracle summaries for CNN/DAILYMAIL, we used the greedy algorithm described in (Nallapati et al., 2016). The model trained with a learning rate of \(5e^{-5}\) and batch size 8 for 1 epoch.
Figure 5: The number of spans extracted by MSEGENDBERT $o_{mse}$ vs. the number of annotated arguments for the same questions.

B Distribution of Extracted Spans by MSEGENDBERT

B.1 Distribution of Extracted and Annotated Spans from DROP

Figure 5 shows the number of extracted spans by MSEGENDBERT compared to the annotated spans, for a subset of 400 examples from the development set of DROP. On average, MSEGENDBERT extracts a similar number of spans per example (2.12) compared to the spans extracted by annotators (1.95). However, MSEGENDBERT tends to over-predict one span compared to the annotated examples.

B.2 Distribution of Extracted Spans on Math-Word-Problems

To further test the emergent behaviour of MSEGENDBERT (§4), we compare the number of extracted spans on an out-of-distribution sample, by MSEGENDBERT and MSEGENDBERT-only mse that was trained without the decoder head ($o_{gen}$). Specifically, we run the models on MAWPS (Koncel-Kedziorski et al., 2016), a collection of small-size math word problem datasets. The results, shown in Figure 6, demonstrate the generalized behaviour of $o_{mse}$, which learns to extract multiple spans when trained jointly with the decoder.

Figure 6: The portion of examples per number of extracted spans by $o_{mse}$, for MSEGENDBERT that was trained on with and without $o_{gen}$. 
**Question:** Who is Bruce Spizer an expert on, known as the most influential act of the rock era? ("The Beatles")

**Context:** The Beatles were an English rock band formed in Liverpool in 1960. With members John Lennon, Paul McCartney, George Harrison and Ringo Starr, they became widely regarded as the foremost and most influential act of the rock era. Rooted in skiffle, beat and 1950s rock and roll, the Beatles later experimented with several musical styles, ranging from pop ballads and Indian music to psychedelia and hard rock, often incorporating classical elements and unconventional recording techniques in innovative ways. In 1963 their enormous popularity first emerged as “Beatlemania”, and as the group’s music grew in sophistication in subsequent years, led by primary songwriters Lennon and McCartney, they came to be perceived as an embodiment of the ideals shared by the counterculture of the 1960s. David “Bruce” Spizer (born July 2, 1955) is a tax attorney in New Orleans, Louisiana, who is also recognized as an expert on The Beatles. He has published eight books, and is frequently quoted as an authority on the history of the band and its recordings.

---

**Question:** Which Eminem album included vocals from a singer who had an album titled “Unapologetic”? ("The Marshall Mathers LP 2")

**Context:** “Numb” is a song by Barbadian singer Rihanna from her seventh studio album “Unapologetic” (2012). It features guest vocals by American rapper Eminem, making it the pair’s third collaboration since the two official versions of “Love the Way You Lie”. Following the album’s release, “Numb” charted on multiple charts worldwide including in Canada, the United Kingdom and the United States. “The Monster” is a song by American rapper Eminem, featuring guest vocals from Barbadian singer Rihanna, taken from Eminem’s album “The Marshall Mathers LP 2” (2013). The song was written by Eminem, Jon Bellion, and Bebe Rexha, with production handled by Frequency. “The Monster” marks the fourth collaboration between Eminem and Rihanna, following “Love the Way You Lie”, its sequel “Love the Way You Lie (Part II)” (2010), and “Numb” (2012). “The Monster” was released on October 29, 2013, as the fourth single from the album. The song’s lyrics present Rihanna coming to grips with her inner demons, while Eminem ponders the negative effects of his fame.

---

**Question:** Are both Dictyosperma, and Huernia described as a genus? ("yes")

**Context:** The genus Huernia (family Apocynaceae, subfamily Asclepiadoideae) consists of stem succulents from Eastern and Southern Africa, first described as a genus in 1810. The flowers are five-lobed, usually somewhat more funnel- or bell-shaped than in the closely related genus "Stapelia", and often striped vividly in contrasting colours or tones, some glossy, others matt and wrinkled depending on the species concerned. To pollinate, the flowers attract flies by emitting a scent similar to that of carrion. The genus is considered close to the genera "Stapelia" and "Hoodia". The name is in honour of Justin Heurnius (1587–1652) a Dutch missionary who is reputed to have been the first collector of South African Cape plants. His name was actually mis-spelt by the collector Dictyosperma is a monotypic genus of flowering plant in the palm family found in the Mascarene Islands in the Indian Ocean (Mauritius, Reunion and Rodrigues). The sole species, Dictyosperma album, is widely cultivated in the tropics but has been farmed to near extinction in its native habitat. It is commonly called princess palm or hurricane palm, the latter owing to its ability to withstand strong winds by easily shedding leaves. It is closely related to, and resembles, palms in the "Archontophoenix" genus. The genus is named from two Greek words meaning "net" and "seed" and the epithet is Latin for "white", the common color of the crownshaft at the top of the trunk.
Question: Which board game was published most recently, Pirate’s Cove or Catan? (“Pirate’s Cove”)

Context: The Settlers of Catan, sometimes shortened to Catan or Settlers, is a multiplayer board game designed by Klaus Teuber and first published in 1995 in Germany by Franckh-Kosmos Verlag (Kosmos) as Die Siedler von Catan. Players assume the roles of settlers, each attempting to build and develop holdings while trading and acquiring resources. Players are awarded points as their settlements grow; the first to reach a set number of points, typically 10, is the winner. The game and its many expansions are also published by Mayfair Games, Filosofia, Capcom, 999 Games, Καισσα, and Devir. Pirate’s Cove (in German, Piratenbucht) is a board game designed by Paul Randles and Daniel Stahl, originally published in Germany in 2002 by Amigo Spiele, illustrated by Markus Wagner and Swen Papenbrock. In 2003, Days of Wonder republished the game with a new graphic design from Julien Delval and Cyrille Daujean. In the game, players play pirate ship captains seeking treasure from islands and bragging rights from defeating other pirates in naval combat.