Do Text-to-Text Learners Suffer from Multi-Task Conflict?

David Mueller ♦ Nicholas Andrews ♦ Mark Dredze

JOHNS HOPKINS
UNIVERSITY

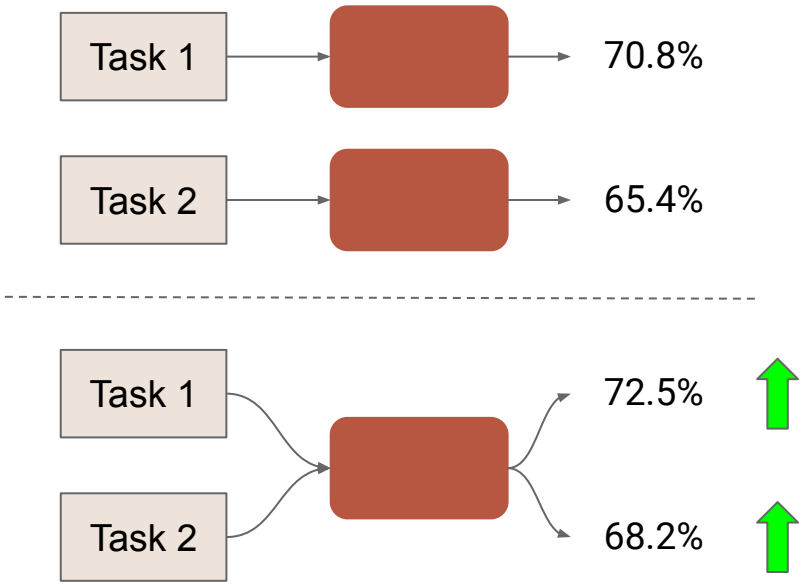
What Do We Mean & Why Does It Matter?

Multi-Task Learning: Why?

Multi-task learning aims to optimize multiple tasks jointly by using a *shared model*.

Why should we use multi-task learning?

In theory, information that is shared across tasks can yield a **stronger model** than what single-task objectives can achieve.



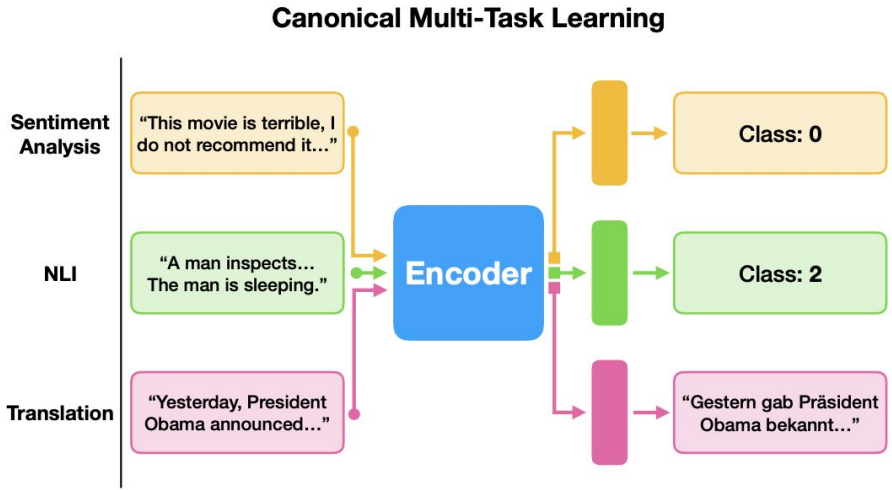
Multi-Task Learning: The Canonical Model

Canonically, we have:

- A shared input space $\mathcal{X}$
- Task-specific output space $\mathcal{Y}_k$.

To model this, we:

- Share an encoder which projects inputs from $\mathcal{X}$ into a low-dim representation space.
- Leverage task-specific **heads** that maps this representation space to predictions in $\mathcal{Y}_k$.



Multi-Task Learning: The Objective

Our goal is to minimize each task's loss by jointly training the parameters of the shared encoder θ , and the parameters of each task's head ϕ_k.

Objective: $\min_{\Theta} \sum_{k=1}^{K} \mathcal{L}_k(\Theta)$

where $\Theta = \{\theta, \phi_1, \ldots, \phi_K\}$

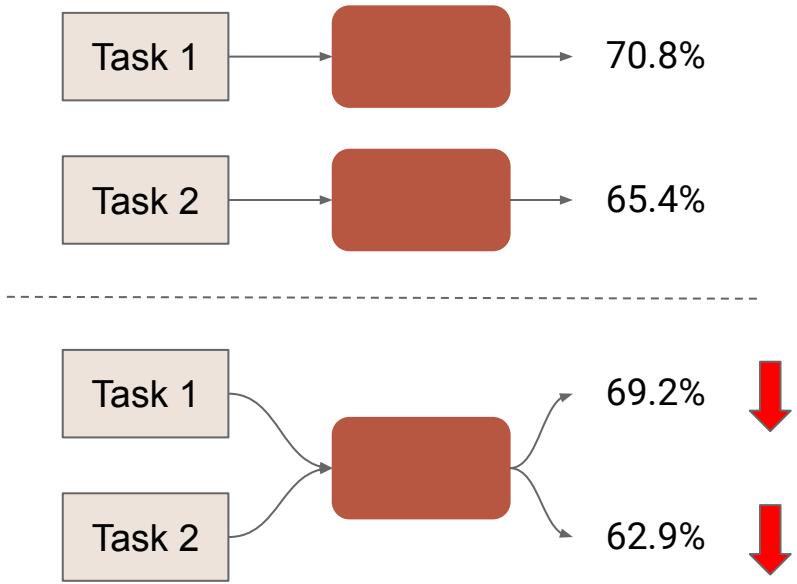
At each iteration, the parameters θ are updated with the *multi-task gradient*

$$\nabla_{\theta}^{MT} = \sum_{k=1}^{K} \nabla_{\theta} \mathcal{L}_k(\Theta)$$

Multi-Task Learning: Negative Transfer

Unfortunately, in practice multi-task learning can often result in *worse* performance than single-task models, i.e. **negative transfer**.

This is undesirable: intuitively, because deep neural networks are so underspecified, more data from related tasks should help us learn a more accurate model.



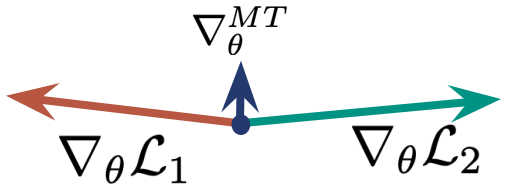
Multi-Task Learning: Task Conflict

Why does negative transfer occur?

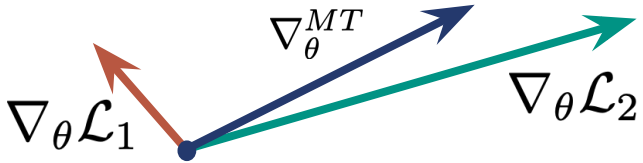
Common intuition is that **task conflict** is a leading cause of negative transfer.

Task conflict refers to significant differences in task gradients that can lead to certain tasks being under optimized, or can lead to optimization getting stuck in poor local minima.

Directional conflict can lead to a multi-task gradient that barely minimizes each task's loss.



Magnitude conflict can result in the gradients of certain tasks dominating the overall multi-task gradient.

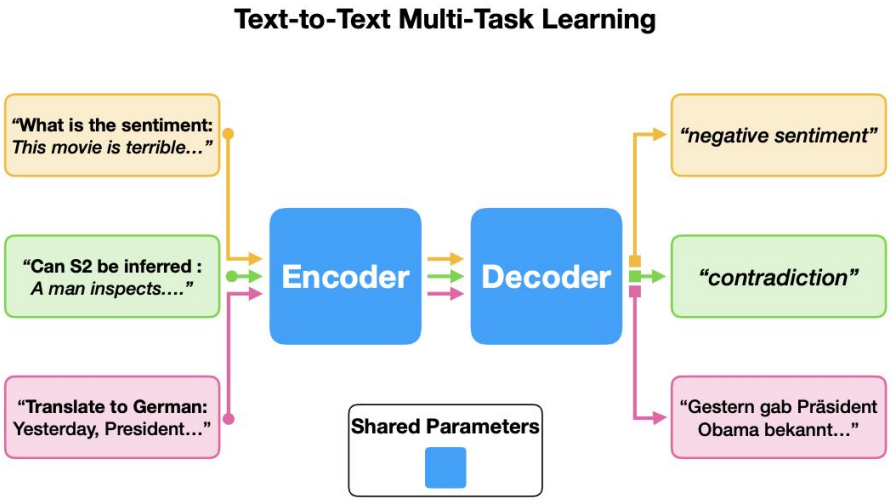


Multi-Task Learning: The Text-to-Text Model

Recently, the field of NLP has gained an interest in Text-to-Text models as multi-task learners.

It has become fairly clear that including *many supervised tasks* as signal during large language model pre-training is an effective way to improve your pre-trained model.

But these models are a departure from Canonical MTL in some key ways.



The Three Factors

What changes when we move from the Canonical model to the Text-to-Text model? In particular, what changes may affect task-conflict, and thus negative transfer?

1. **Text-to-Text Framing**
   - All tasks use LM decoders and LM losses, e.g. the output space of all tasks is homogenized.
2. **No Task-Specific Parameters**
   - Parameters in the encoder and decoder are shared across all tasks. No task-specific parameters.
3. **Prompts & Output Spaces**
   - Each task leverages *task prompts*, sequence of vocabulary which *specify* the task to be performed.

The Core Question

Do the 3 factors that we identify have a significant impact on either multi-task conflict or negative transfer? Is there an *inherent advantage* to modeling multiple tasks as a text-to-text objective?

Why Does It Matter?

Models such as T0, ExT5, and FLAN leverage *hundreds* of tasks as a *pre-training objective*. However, very rarely is their efficacy as a *multi-task model* evaluated. If we are interested in training a truly multi-task model, should we still be using Text-to-Text models? Despite the prevalence of multi-task learning in recent NLP research, the answers to this questions are unknown.

Why Does It Matter?

The standard multi-task objective is an effective pre-training objective, but that *doesn't mean it's the best pre-training objective*. If task conflict and negative transfer are just as problematic for Text-to-Text learners, LLMs may still benefit from more sophisticated multi-task optimization techniques as a pre-training objective.

JOHNS HOPKINS UNIVERSITY

Factors 1 & 2 [Architectural Factors]

Experimental Setup: Datasets

For our experiments we separately consider 2 distinct multi-task settings.

GLUE is a standard, if overused, multi-task benchmark in NLP.

- 7 Classification Tasks
  - 1 Acceptability Task
  - 3 Similarity / Paraphrasing
  - 3 Inference Tasks
- 1 Regression Task
  - Sentiment

DecaNLP consists of 10 tasks, which are much more diverse in output space.

- 2 Classification Tasks
  - Sentiment
  - Inference
- 3 Sequence to Sequence Tasks
  - Translation
  - Summarization
  - Semantic Parsing
- 4 Span-Labeling Tasks

Experimental Setup: The Canonical Model

We use a *pre-trained* T5 model as a starting point for all models we consider.

We begin with **Canonical Models** which leverage a pre-trained T5 Encoder as the shared encoder across all tasks. This Encoder projects inputs from $\mathcal{X}$ into a low-dimensional representation space $\mathbb{R}^{d \times w}$, where d is the dimension, and w is the sequence length.

These representations are passed to *task-specific heads*: $h_{\phi_k} : \mathbb{R}^{d \times w} \rightarrow \mathcal{Y}_k$. For classification tasks, these heads may be simple linear layers. For e.g. the translation task, this head will be a language model decoder.

Experimental Setup: Factor 1 (Text-to-Text-ID)

Factor 1 refers to re-framing multi-task learning as a Text-to-Text problem.

Rather than modeling task-specific output spaces $f_k : \mathcal{X} \rightarrow \mathcal{Y}_k$ each tasks output space is the same as it's input space: $f_k : \mathcal{X} \rightarrow \mathcal{X}$

However, we keep task-specific heads. Each task has a separate LM decoder which maps from $\mathbb{R}^{d \times w}$ to $\mathcal{X}$.

Each task still has task-specific parameters. We call these models **Text-to-Text-ID Models**, because they have *Independent Heads*.

To specify tasks we use the default prompts suggested with T5 for GLUE, and the default prompts that were used when initially building the DecaNLP dataset for DecaNLP.

Experimental Setup: Factor 2 (Text-to-Text)

Factor 2 refers to the removal of all task-specific parameters.

Naturally, to enable this in many settings we already require Factor 1 (a homogenous output space and loss for all tasks). We start from our Text-to-Text-ID architecture and unify each task head into a single LLM decoder.

The only difference between this model and Text-to-Text-ID models are that the decoder parameters are now shared across all tasks, in addition to the encoder parameters.

This model represents the model used in the original T5 paper: a fully unified **Text-to-Text Model** for multi-task learning.

Negative Transfer

To measure negative transfer in our multi-task models, we require single-task models as baselines. Negative transfer, or transfer in general, is measured as the *difference on in-domain test-set performance*.

In our experiments, **we're looking for significant differences between transfer across Canonical and Text-to-Text models.** Because these models do not necessarily achieve comparable performance on all tasks, we will look at *relative* negative transfer, e.g. the percentage of increase or decrease from single-task performance.

However, we note that most tasks achieve comparable single-task performance across Canonical and Text-to-Text models. Exceptions include STS-B (a regression task) and span-labeling tasks which suffer from the removal of constraints on the output.

		GLUE									
Architecture	Training	CoLA	SST	MSRPC	STSB	QQP	MNLI	MNLI-mm	QNLI	RTE	**Avg**
Canonical	Single-Task	36.20	90.29	80.80	87.46	88.69	80.34	80.72	87.96	61.25	78.06
Canonical	Multi-Task	30.27	89.98	82.60	87.58	87.89	78.21	78.44	85.84	64.98	76.20
Text-to-Text	Single-Task	36.28	90.44	82.92	64.27	88.83	80.60	81.27	86.87	59.33	77.55
Text-to-Text-ID	Multi-Task	28.64	90.11	84.62	62.08	86.91	77.44	78.14	84.72	65.70	73.15
Text-to-Text	Multi-Task	35.12	90.71	85.21	67.92	87.51	78.98	79.73	86.30	68.47	75.55

		DecaNLP									
Architecture	Training	SST	MNLI	IWSLT	CNN/DM	Seq2SQL	SQUAD	QA-SRL	QA-ZRE	Wino	**Avg**
Canonical	Single-Task	88.74	80.40	23.55	40.22	60.28	66.57	67.34	54.85	33.93	53.41
Canonical	Multi-Task	90.92	81.15	21.77	39.48	57.25	67.29	70.07	65.79	36.90	58.96
Text-to-Text	Single-Task	89.69	80.06	23.74	40.16	60.26	73.00	63.45	47.43	27.38	56.13
Text-to-Text-ID	Multi-Task	91.61	81.10	23.05	39.88	58.52	72.53	62.90	55.57	31.85	57.44
Text-to-Text	Multi-Task	90.33	80.08	22.41	39.54	58.13	72.72	73.85	59.33	47.62	60.45

Transfer across Canonical and Text-to-Text Architectures (GLUE & DecaNLP)

Takeaways

Factors 1 & 2 seem to have little effect on which tasks suffer and which tasks benefit in multi-task learning (e.g. negative transfer remains largely consistent).

This is perhaps not surprising: it suggests that the relationship between tasks during learning is intrinsic to the distribution of the tasks, and is robust to e.g. significant architectural changes.

However, it *also* suggests that Text-to-Text learners are not magical: they exhibit the same multi-task behaviors as canonical models.

Measuring Conflict

In addition to negative transfer, we would like to study how *task conflict* is affected by a shift in architecture from Canonical to Text-to-Text.

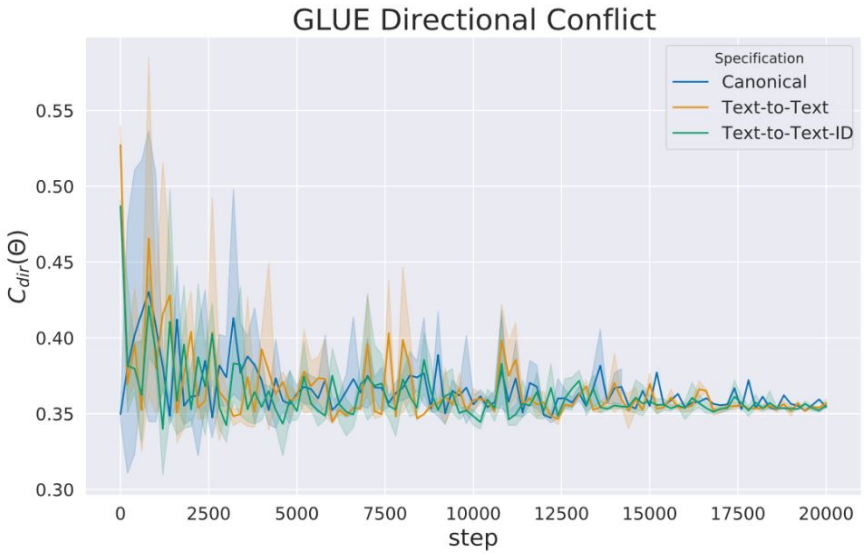
We study both directional conflict $C_{dir}(\Theta)$ and magnitude conflict $C_{mg}(\Theta)$ in the encoder of Canonical, Text-to-Text-ID, and Text-to-Text models, and plot their trajectories during training.

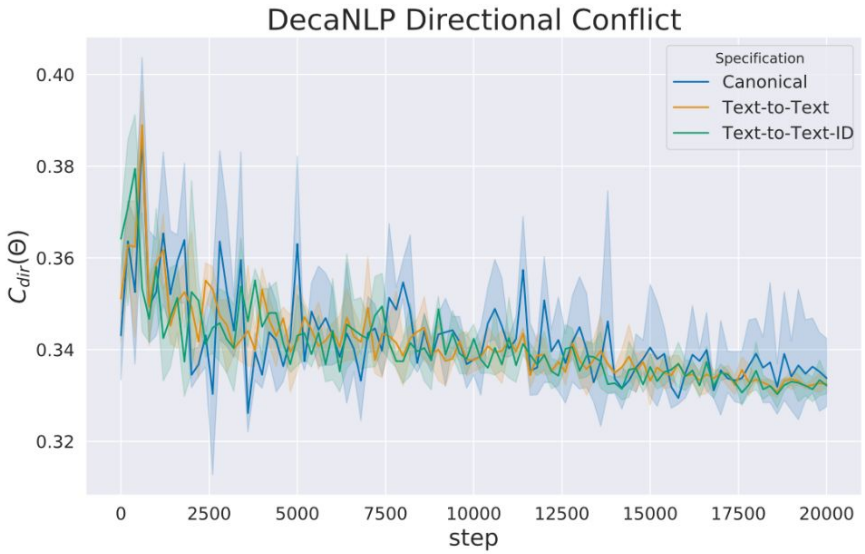
Directional conflict is measured as the norm of the mean normalized gradient over all tasks (the average pairwise cosine similarity, up to a constant).

$$C_{dir}(\Theta) = \left|\left| \frac{1}{K} \sum_{k=1}^{K} \frac{\nabla_\theta \mathcal{L}_k(\cdot)}{||\nabla_\theta \mathcal{L}_k(\cdot)||_2} \right|\right|_2^2 - \frac{1}{K}$$

Magnitude conflict is simply measured as the covariance of task gradient norms.

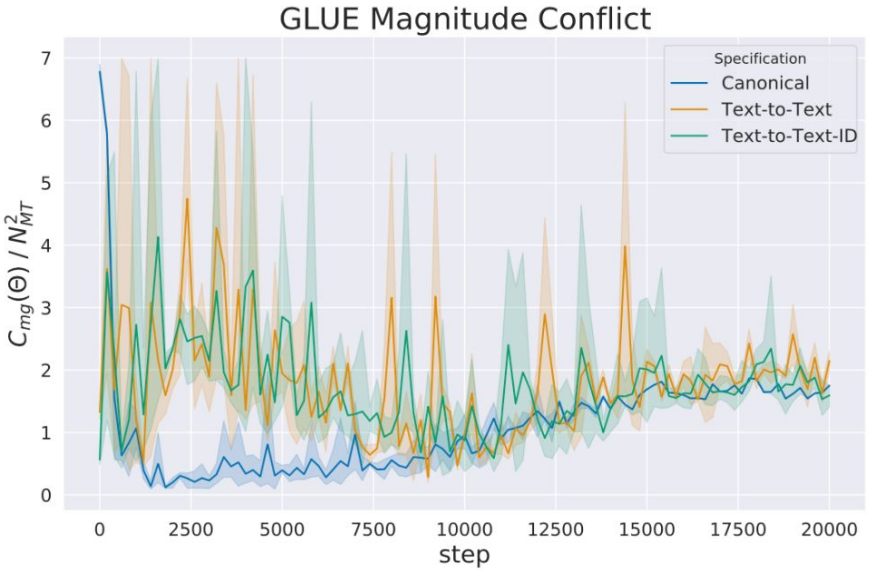
$$C_{mg}(\Theta) = \frac{1}{K} \sum_{k=1}^{K} \left(||\nabla_\theta \mathcal{L}_k(\cdot)||_2^2 - ||\nabla_\theta^{MT}||_2^2 \right)^2$$

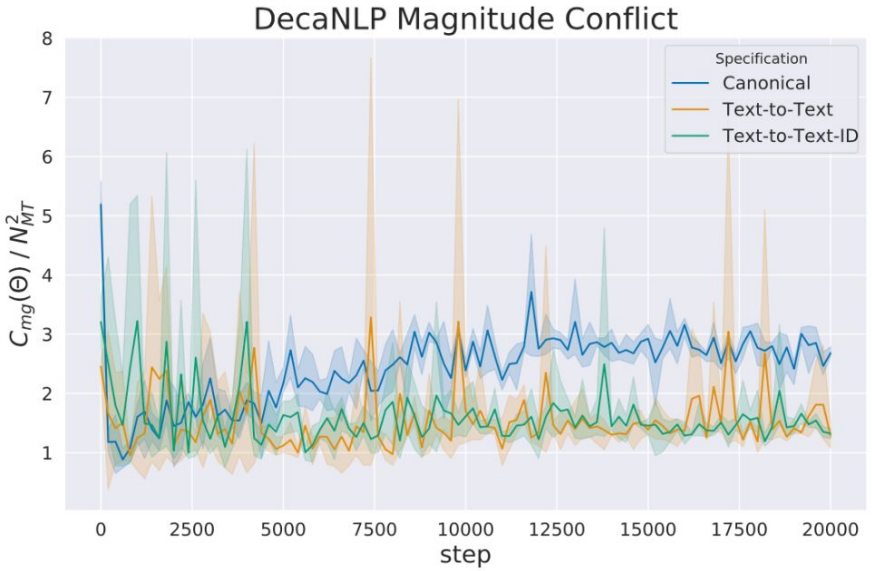




Canonical and Text-to-Text models suffer roughly similar levels of directional conflict in the encoder.

Directional Conflict in the Encoder of Multi-Task Models (GLUE & DecaNLP)





Magnitude conflict does exhibit noticeable differences between Canonical and Text-to-Text models. However, these differences apparently have little bearing on negative transfer (at least in the settings we consider).

Magnitude Conflict in the Encoder of Multi-Task Models (GLUE & DecaNLP)

Takeaways so far.

Factors 1 & 2 appear to have little effect on both task conflict *and* negative transfer.

The shift from Canonical to Text-to-Text models for multi-task learning does little *on it's own* to mitigate task conflict and negative transfer.

Factor 1 *alone* does seem harmful, overall, to task-performance, indicating that surprisingly sharing *all parameters in the decoder* is a good thing.

Factor 3 [Prompts & Output Spaces]

What is Factor 3?

Factors 1 & 2 represent *architectural changes* to the way we model multiple tasks. However, Factor 3 assumes that we are modeling each task in a text-to-text way, and in instead concerned with *specification*: how do we tell the model which task it's performing?

Factor 3 is concerned with the specific *vocabulary items* we select as both task prompts and task outputs.

JOHNS HOPKINS UNIVERSITY

What is Factor 3?

Intuitively, the semantics & overlap of the task descriptions and output spaces may have some bearing on negative transfer or conflict. For instance, a pre-trained model with some semblance of semantic knowledge may be able to leverage semantically rich task descriptions during training to infer task relatedness.

The Input Space (Task Prompts)

In our experiments we use fixed, *non-semantic* task identifiers to specify tasks in text-to-text models (e.g. SST: …). However, we could use a more descriptive set of prompts for each task. In fact, we could use a *diverse* set of prompts for each task.

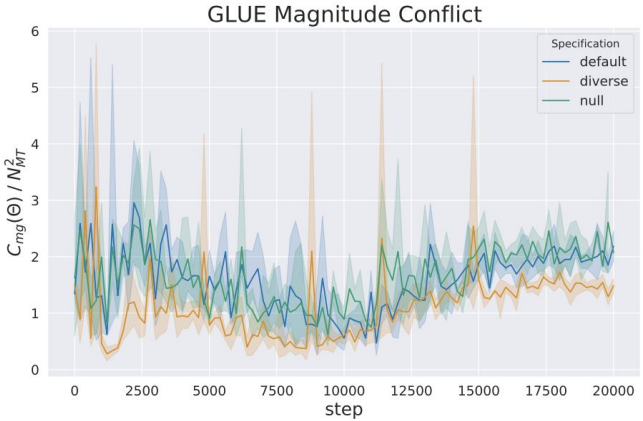
Models trained on diverse prompts (e.g. T0) have been shown to have stronger zero-shot generalization than models trained on fixed task prompts, suggesting a stronger understanding of task semantics.

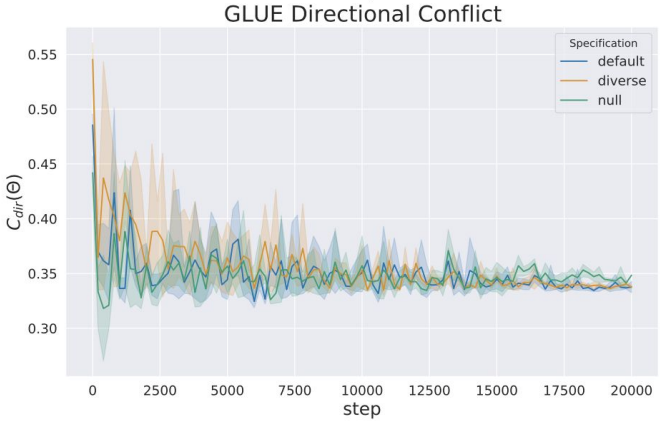
If we train on diverse task prompts, does an improved understanding of task descriptions lead to an improvement in negative transfer?

Prompt	CoLA	SST	MSRPC	STSB	QQP	MNLI	MNLI-mm	QNLI	RTE	**Sum**
Single-Task (Null Prompt)	36.28	90.44	82.92	64.27	88.83	80.60	81.27	86.87	59.33	77.55
Null Prompt	29.56	90.34	83.46	48.20	86.88	76.71	76.40	85.08	50.99	50.99
Default Prompt	35.12	90.71	85.21	67.92	87.51	78.98	79.73	86.30	68.47	75.55
Diverse Prompt	32.39	89.94	79.96	66.30	87.31	77.22	77.87	86.25	65.61	73.65

Null prompts are generally very poor in multi-task learning. *Some amount of specification* must occur. However, having *diverse* specifications is actually harmful.

The Effects of Diverse Prompts





Interestingly, we do see a slight difference in *magnitude* conflict: diverse prompts result in *lower* magnitude conflict, suggesting that it may even out task losses. However, again, we do not see a strong correlation between magnitude conflict and negative transfer.

Additionally, we see that diverse prompts have little effect on directional conflict, again suggesting that directional relationships between tasks is intrinsic to the distribution of the task data itself.

The Effects of Diverse Prompts

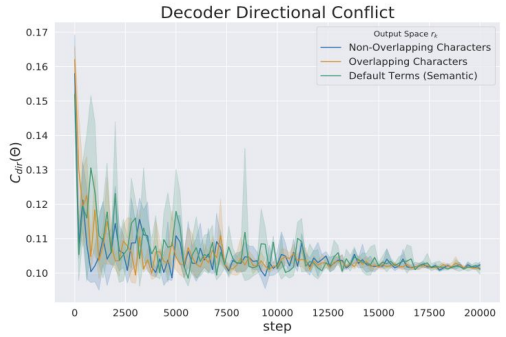
Takeaways

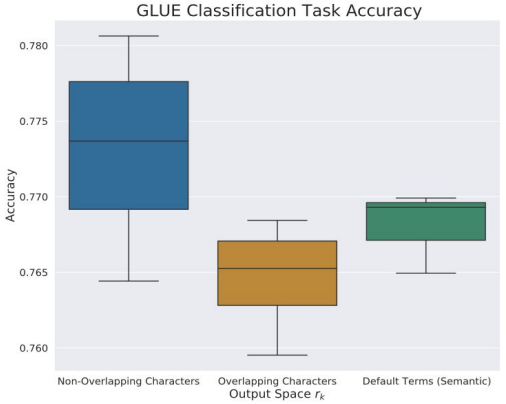
Diverse task specification can improve zero-shot generalization by (ostensibly) improving semantic understanding of task descriptions. However, from a multi-task perspective, using diverse prompts across tasks can increase negative transfer. Is there a tradeoff between “understanding task prompts” (i.e. zero-shot generalization) and in-domain performance (negative transfer)?

Overlap In The Output Space

Finally, we study how overlap in the output space may affect multi-task learning. We study artificial tokens (random characters) for each class in a classification task. We consider characters that are unique across all tasks, and characters that are shared across all tasks (e.g. the letter “a” means “equivalent” for SST and “entailed” for NLI).

Interestingly, we see that non-overlapping characters perform better than overlapping characters *and* the default (semantic) terms. Distinct vocabulary terms for individual tasks





Takeaways

From the perspective of negative transfer, having distinct well-defined (not necessarily semantic) prompts and output spaces for all tasks alleviates negative transfer. While semantically rich tokens are important for zero-shot generalization, they seem to impair performance on in-domain test-sets.

Conclusions & Future Directions

Modeling MTL as a Text-to-Text problem has little to no bearing on the actual performance of the multi-task model. Regardless of architecture, the relation between tasks dominates trends.

There are a couple tricks in Text-to-Text models that can alleviate negative transfer, but they are at odds with zero-shot generalization.

Despite heavy use from the MTL community, common metrics of task conflict seem meaningless. Rarely did they predict trends in generalization.