



MULTI-TASK DEEP NEURAL NETWORKS FOR NATURAL LANGUAGE UNDERSTANDING

XIAODONG LIU, PENGCHENG HE, WEIZHU CHEN, JIANFENG GAO

Alberto BALDRATI, Giovanni BERTI

Supervisor: Prof. Paolo FRASCONI

Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Firenze

INDEX



Introduction

- Previous Multi Task Model
- BERT

Multi-Task Deep Neural Network

- Multi-Task Learning
- Datasets
- Architecture overview
- Task Specific Layers
- Training Procedure

Experiments

- Experimental setup
- Experimental results

Conclusions

INTRODUCTION



INTRODUCTION



- ▶ In Liu et al., 2019 is presented a Multi-Task Deep Neural Network (MT-DNN) for learning representations across multiple natural language understanding (NLU) tasks
- ▶ MT-DNN extends the models proposed in Liu et al., 2015 by incorporating a pre-trained bidirectional transformers language model, known as BERT (Devlin et al., 2019)
- ▶ MT-DNN obtains new state-of-the-art results on ten NLU tasks, including SNLI, SciTail, and eight out of nine GLUE tasks
- ▶ It is also shown that multi task learning allows a more domain adaptation with fewer data in SNLI and SciTail datasets



PREVIOUS MULTI TASK MODEL

- ▶ In [Liu et al., 2015](#) the authors showed how even simple neural networks can learn a more general language representation with the aid of multi-task learning in query classification and web search ranking
- ▶ The proposed Multi-Task DNN Model was made of three layers of where the first two of them were shared among tasks:
 - **Word Hash Layer** (l_1) maps one hot-word vectors, with an extremely high dimensionality, into a limited letter-trigram space with dimensionality about 50K
 - **Semantic-Representation Layer** (l_2): maps the letter-trigram input into a 300-dimensional vector by $l_2 = \tanh(\mathbf{W}_1 \cdot l_1)$
 - **Task-Specific Representation** (l_3) maps, the 300-dimensional semantic representation l_2 into the 128-dimensional task-specific representation by $l_3 = \tanh(\mathbf{W}_2^t \cdot l_2)$

PREVIOUS MULTI TASK MODEL ARCHITECTURE

- The last task specific representation is designed to either classify queries or rank web search output using the computed 128D input

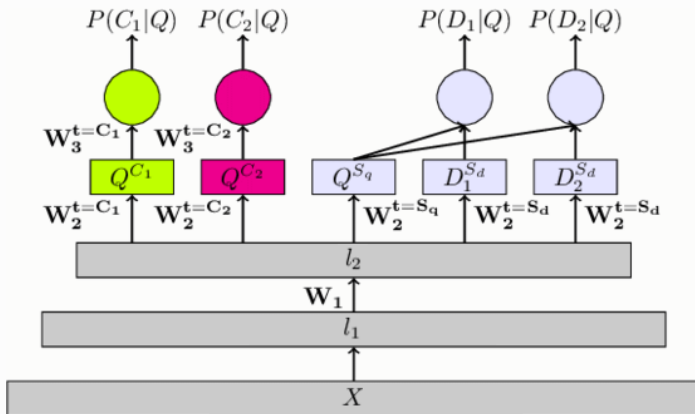


Figure: Previous MT-DNN architecture

BERT

- BERT Devlin et al., 2019, which stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers is a transformer-based language model
- BERT architecture is a multi-layer bidirectional Transformer encoder(Vaswani et al., 2017). It is designed to be pre-trained on unlabeled text data and its architecture allows every token to attend to every other token.

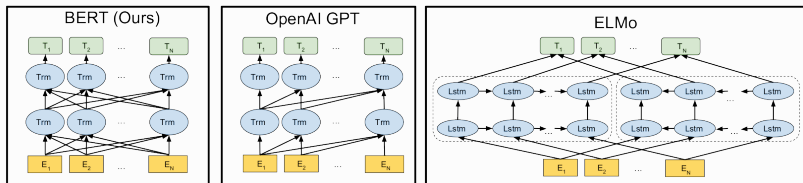


Figure: BERT comparison with other transformers-based language models

BERT

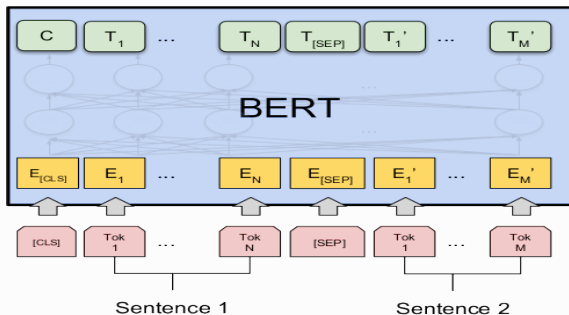
PRE-TRAINING



- ▶ BERT is pre-trained via two unsupervised tasks:
 - **Masked Language Modeling** consists in predicting correct words in sentences after they have been masked or replaced with another word. This masking task allows training of a fully bidirectional model, as opposed to traditional left-to-right conditional language models.
 - **Next Sequence Prediction** consists in training the model to output sentence relationships between pairs of clauses. More specifically, the model is trained to classify with one of IsNext/NotNext labels. This kind of classification is fundamental to many important natural language inference tasks
- ▶ This pre-trained model is then used as a backbone when fine tuning for downstream tasks

BERT

OPERATING SCHEME



Input	<div> <div>[CLS]</div> <div>my</div> <div>dog</div> <div>is</div> <div>cute</div> <div>[SEP]</div> <div>he</div> <div>likes</div> <div>play</div> <div>#ing</div> <div>[SEP]</div> </div>										
Token Embeddings	$E_{[CLS]}$	E_{my}	E_{dog}	E_{is}	E_{cute}	$E_{[SEP]}$	E_{he}	E_{likes}	E_{play}	$E_{\#ing}$	$E_{[SEP]}$
	+	+	+	+	+	+	+	+	+	+	+
Segment Embeddings	E_A	E_A	E_A	E_A	E_A	E_A	E_B	E_B	E_B	E_B	E_B
	+	+	+	+	+	+	+	+	+	+	+
Position Embeddings	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}



MULTI-TASK DEEP NEURAL NETWORK

MT-DNN

NLU AND MULTI-TASK TRAINING



- ▶ MT-DNN extends the models proposed in Liu et al., 2015 by incorporating the pre-trained bidirectional transformer-based language model BERT
- ▶ Learning multiple tasks jointly has been shown to be effective in training language models, either by leveraging data from related tasks or by learning more general language representations
- ▶ The reference training tasks are based on General Language Understanding Evaluation (GLUE) benchmark, plus SNLI and SciTail for evaluating domain adaptation
- ▶ MTL and language model pretraining are complementary technologies, and can be combined to improve the learning of text representations to boost the performance of various NLU tasks

DATASETS



- ▶ **GLUE** (Wang et al., 2019) (General Language Understanding Evaluation) is a benchmark consisting in many different NLU tasks, including question answering, text similarity and textual entailment
- ▶ **SNLI** (Bowman et al., 2015) (Stanford Natural Language Inference) dataset contains 570k premise-hypothesis sentence pairs. It is mainly used for textual entailment in NLI.
- ▶ **SciTail** (Khot et al., 2018) is a textual entailment dataset analogous to SNLI, with the difference that answer candidates are collected from relevant web sentences retrieved from a large corpus, this making the task more difficult.
- ▶ GLUE tasks were used for multi-task learning while SNLI and SciTail were tested in a domain adaptation setting

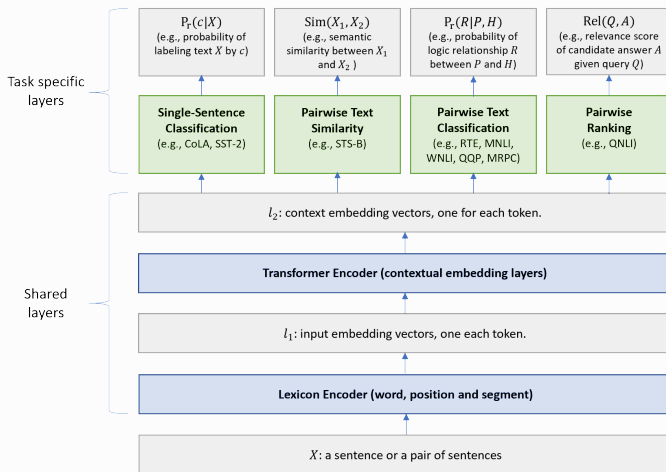
GLUE TASK CATEGORIES



- ▶ **Single Sentence Classification:** Given a sentence, the model labels it using one of the predefined class labels. (CoLA, SST-2)
- ▶ **Pairwise Text Similarity:** Given a pair of sentences, predict a semantic similarity score between the two sentences. (STS-B)
- ▶ **Pairwise Text Classification:** Given a pair of sentences, predict a semantic relationship between the two sentences (i.e. in RTE and MNLI whether there is an entailment relationship; in QQP and MRPC whether the two sentences are semantically equivalent)
- ▶ **Pairwise Ranking:** Given a query and an answer, predict a relevance score by assessing whether the second sentence contains the correct answer to the query. (QNLI)

MT-DNN

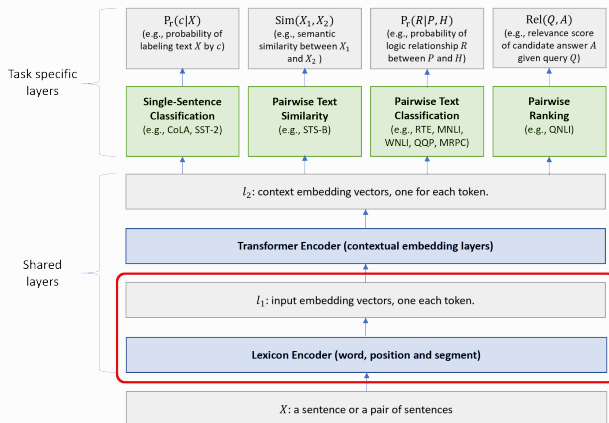
ARCHITECTURE OVERVIEW



MT-DNN

LEXICON ENCODER

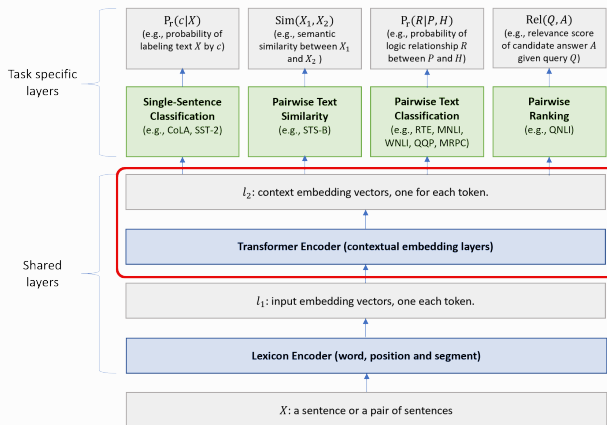
- The **Lexicon Encoder** (l_1) maps sequences of tokens (X) into input embedding vectors by summing their word, segment and positional embeddings.
WordPiece is used to compute word embeddings.



MT-DNN

TRANSFORMER ENCODER

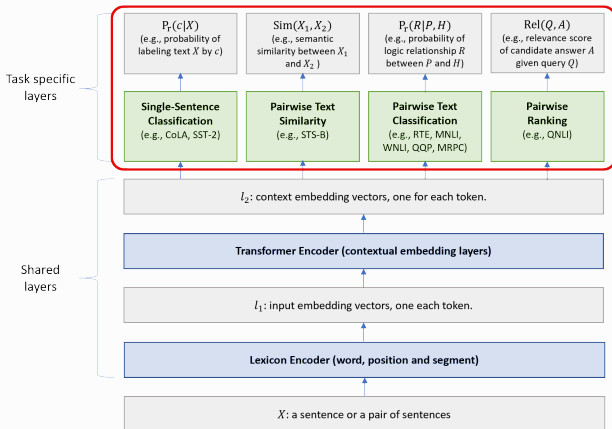
- The **Transformer Encoder** (l_2) maps representation vectors from l_1 into contextual embedding vectors using a multi-layer transformer architecture



MT-DNN

TASK-SPECIFIC LAYERS

- Different **task-specific layers** are fine-tuned along with the transformer model. Each task has its own parameters while the underlying transformer is shared.





TASK SPECIFIC LAYERS

SINGLE SENTENCE CLASSIFICATION

- ▶ [CLS] token is an extra token produced by the l_1 embedding at the beginning of the input. Its l_2 embedding C is used for classification purposes.
- ▶ The attention mechanism allows C to attend to all other token embeddings, thus encoding the overall meaning of the sentence
- ▶ This layer takes C as an input \mathbf{x}
- ▶ It is a dense layer with a parameter matrix of size $H \times k$ (where k equals the number of output classes) with a *softmax* activation function, where H is the hidden size of the transformer module:

$$P_r(c | X) = \text{softmax}(\mathbf{W}^T \cdot \mathbf{x})$$

- ▶ CoLA and SST-2 tasks fall into this category



TASK SPECIFIC LAYERS

PAIRWISE TEXT SIMILARITY

- ▶ Here C can be seen as the semantic representation of the input sentence pair (X_1, X_2)
- ▶ This layer takes C as an input \mathbf{x}
- ▶ It is a dense layer with a parameter vector of size H , where H is the hidden size of the transformer module:

$$\text{Sim}(X_1, X_2) = \mathbf{w}^T \cdot \mathbf{x}$$

- ▶ STS-B task falls into this category



TASK SPECIFIC LAYERS

PAIRWISE TEXT CLASSIFICATION



17

- ▶ Takes as input two sentence embeddings. We will refer the first as premise $P = (p_1, \dots, p_m)$ and the second one as hypothesis $H = (h_1, \dots, h_n)$ and outputs the logical relationship between P and H .
- ▶ The design of such layer follows the answer module of the stochastic answer network (SAN) ([Liu et al., 2018](#))
- ▶ SAN's answer module uses multi-step reasoning. Rather than directly predicting the entailment given the input, it maintains a state and iteratively refines its predictions.
- ▶ MNLI, RTE, WNLI, QQP, MRPC tasks fall into this category.



TASK SPECIFIC LAYERS

PAIRWISE TEXT CLASSIFICATION

- ▶ We denote as $\mathbf{M}^p \in \mathbb{R}^{d \times m}$ the output of the transformer encoder of premise P input, similarly we denote $\mathbf{M}^h \in \mathbb{R}^{d \times n}$ the output of the transformer encoder of premise H input.
- ▶ We perform K -step reasoning on such matrix representation. At the beginning the initial state \mathbf{s}^0 is the summary of \mathbf{M}^h :

$$\mathbf{s}^0 = \alpha^T \cdot \mathbf{M}^h \quad \text{where } \alpha = \text{softmax}(\mathbf{w}_1^T \cdot \mathbf{M}^h)$$

- ▶ A reasoning step k in the range of $\{1, 2, \dots, K-1\}$ the state is defined by

$$\mathbf{s}^k = \text{GRU}(\mathbf{s}^{k-1}, \mathbf{x}^k)$$

- ▶ \mathbf{x}^k is computed from the previous state \mathbf{s}^{k-1} and \mathbf{M}^p :

$$\mathbf{x}^k = \beta^T \cdot \mathbf{M}^p \quad \text{and } \beta = \text{softmax}(\mathbf{s}^{k-1} \cdot \mathbf{W}_2^T \cdot \mathbf{M}^p)$$

TASK SPECIFIC LAYERS

PAIRWISE TEXT CLASSIFICATION



19

- ▶ A one-layer classifier is used to determine the relation at each step k :

$$p_r^k = \text{softmax}(\mathbf{W}_3^T \cdot [\mathbf{s}^k; \mathbf{x}^k; |\mathbf{s}^k - \mathbf{x}^k|; \mathbf{s}^k \odot \mathbf{x}^k])$$

- ▶ At last, the final output is determined by averaging the K scores:

$$P_r = \text{avg}([p_r^0, p_r^1, \dots, p_r^{K-1}])$$

- ▶ Each P_r is a probability distribution over all logical relationships
- ▶ During training *stochastic prediction dropout* (Liu et al., 2017) is applied before the averaging operation. During decoding all K outputs are considered in the averaging operation.

TASK SPECIFIC LAYERS

RELEVANCE RANKING



20

- ▶ Here the C can be seen as the semantic representation of the input sentence pair (Question, Answer)
- ▶ This layer takes C as an input \mathbf{x}
- ▶ It is a dense layer with a parameter vector of size H , with a sigmoid activation function, where H is the hidden size of the transformer module:

$$\text{Rel}(Q, A) = \text{sigmoid}(\mathbf{w}^T \cdot \mathbf{x})$$

- ▶ QNLI task falls into this category



TRAINING PROCEDURE

LOSSES

- **Single Sentence Classification** and **Pairwise Text Classification**: crossentropy loss

$$-\sum_c \mathbb{1}(X = c) \log(P_r(c | X))$$

- **Text Similarity**: square loss

$$(y - \text{Sim}(X_1, X_2))^2$$

- **Relevance Ranking**: given a query Q we extract a list of candidate answers \mathcal{A} , where only one answer A^+ is correct and all other answers are incorrect. We then compute

$$P_r(A | Q) = \text{softmax}(\text{Rel}(Q, \mathcal{A}))$$

And use negative log likelihood as the loss function

$$-\sum_{(Q, A^+)} \log(P_r(A^+ | Q))$$



TRAINING PROCEDURE

LOSSES

- ▶ **Single Sentence Classification** and **Pairwise Text Classification**: crossentropy loss

$$-\sum_c \mathbb{1}(X = c) \log(P_r(c | X))$$

- ▶ **Text Similarity**: square loss

$$(y - \text{Sim}(X_1, X_2))^2$$

- ▶ **Relevance Ranking**: In the case of QNLIv2, due to the different dataset structure, binary crossentropy loss was used instead, turning the ranking problem into a binary classification problem
 - Since QNLIv1 expired on 2019-01-30 due to dataset issues, we perform all of our experiments on QNLIv2 and treat QNLI as a binary classification problem



TRAINING PROCEDURE

MULTI-TASK TRAINING ALGORITHM

```

1 Initialize model parameters  $\Theta$  randomly.
2 Pre-train the shared layers
3 for  $t$  in  $1, 2, \dots, T$  do
4   | Pack the dataset  $t$  into mini-batches:  $D_t$ .
5 end
6 for  $epoch$  in  $1, 2, \dots, epoch_{max}$  do
7   | 1. Merge all the datasets:  $D = D_1 \cup D_2 \dots \cup D_T$ 
8   | 2. Shuffle  $D$ 
9   | for  $b_t$  in  $D$  do
10  |   | //  $b_t$  is a mini-batch of task  $t$ .
11  |   | 3. Compute loss :  $L(\Theta)$ 
12  |   |    $L(\Theta) = \text{Crossentropy loss}$  for classification
13  |   |    $L(\Theta) = \text{Square loss}$  for regression
14  |   |    $L(\Theta) = \text{Ranking loss}$  for ranking
15  |   | 4. Compute gradient:  $\nabla L(\Theta)$ 
16  |   | 5. Update model:  $\Theta = \Theta - \epsilon \nabla L(\Theta)$ 
17  | end
18 end
  
```

TRAINING PROCEDURE

TASK FINE-TUNING



- ▶ Since the Multi-Task-Learning of MT-DNN uses all GLUE tasks, it is possible to directly apply the model trained to each GLUE task individually
- ▶ However in tasks such as CoLA where the available dataset is very small MT-DNN tends to underfit
- ▶ To alleviate this we further fine-tune the MT-DNN model on each task
- ▶ It is shown that such fine-tuning step improves the overall performance of the model

TRAINING PROCEDURE

DOMAIN ADAPTATION



- ▶ In the case of SNLI and SciTail we perform *domain adaptation*: after training with Multi-Task learning on GLUE task the model is fine-tuned without having seen these datasets before
- ▶ This measures how general the representations learned through multi task learning are and how flexible the system is
- ▶ Both SNLI and SciTail are *Pairwise Text Classification* tasks
- ▶ To test domain adaptation, a simple linear layer is added on top of the base MT-DNN model with C as input, with a *softmax* activation function

EXPERIMENTS





DATASETS AND METRICS

Corpus	Task	#Train	#Dev	#Test	#Label	Metrics
Single-Sentence Classification (GLUE)						
CoLA	Acceptability	8.5k	1k	1k	2	Matthews corr
SST-2	Sentiment	67k	872	1.8k	2	Accuracy
Pairwise Text Classification (GLUE)						
MNLI	NLI	393k	20k	20k	3	Accuracy
RTE	NLI	2.5k	276	3k	2	Accuracy
WNLI	NLI	634	71	146	2	Accuracy
QQP	Paraphrase	364k	40k	391k	2	Accuracy/F1
MRPC	Paraphrase	3.7k	408	1.7k	2	Accuracy/F1
Text Similarity (GLUE)						
STS-B	Similarity	7k	1.5k	1.4k	1	Pearson/Spearman corr
Relevance Ranking (GLUE)						
QNLI	QA/NLI	108k	5.7k	5.7k	2	Accuracy
Pairwise Text Classification						
SNLI	NLI	549k	9.8k	9.8k	3	Accuracy
SciTail	NLI	23.5k	1.3k	2.1k	2	Accuracy

EXPERIMENT SETUP



- ▶ Due to memory limitations we have performed experiments using BERT_{BASE} with bert - base - uncased configuration
- ▶ We have trained the MT-DNN model for **5** epochs
- ▶ We have fine-tuned the MT-DNN model for **10** epochs
- ▶ To assess the effect of Multi-Task-Learning we have also trained *Single-Task* version of the aforementioned model for **10** epochs
- ▶ We tested domain adaptation with random downsampling of the dataset, of sizes 0.1%, 1%, 10%, 100%. We have conducted such experiments **5** times and we have taken the average of the runs as the result



EXPERIMENTAL SETUP

- ▶ Our implementation of MT-DNN is based on the PyTorch implementation of BERT
- ▶ The following training hyperparameters are used:
 - **Optimizer:** Adamax
 - **Learning rate:** $5e-5$ with **linear warm-up** over 10% of the dataset
 - **Batch size:** 32^\dagger
 - **Dropout rate:** 0.1 for all task specific layers, except 0.3 for MNLI and 0.05 for CoLA
 - **Gradient clipping:** set to unitary norm
 - **Maximum sequence length:** 512 tokens
- ▶ † when batch size 32 exceed the GPU memory limit we dynamically halve the batch size



GLUE DEV SET RESULTS

Model	MNLI _{m/mm} Acc	QQP F1/Acc	RTE Acc	QNLI Acc	MRPC F1/Acc	CoLa MCC	SST-2 Acc	STS-B Acc
BERT _{LARGE}	86.3/86.2	88.0/91.1	71.1	92.4	89.5/85.8	61.8	93.5	89.6/89.3
ST-DNN	86.6/86.3	88.4/91.3	72.0	-	89.7/86.4	-	-	-
MT-DNN	87.1/86.7	89.2/91.9	83.4	92.9	91.0/87.5	63.5	94.3	90.7/90.6
ST-DNN_{ours}	82.3/83.1	86.4/89.8	66.0	91.2	89.1/84.0	53.3	93.2	88.6/88.1
MT-DNN_{ours} <small>no-fine-tune</small>	82.0/82.0	85.5/88.9	70.0	90.5	90.2/86.0	51.5	90.7	88.0/87.9
MT-DNN_{ours}	83.1/82.8	87.0/90.2	75.0	91.2	93.2/90.4	55.7	93.2	89.7/89.7

Table: Dev test results

GLUE TEST SET RESULTS

Model	CoLA MCC 8.5k	SST-2 Acc 67k	MRPC F1/Acc 3.7k	STS-B P/S Corr 7k	QQP F1/Acc 364k	MNLI _{m/mm} Acc 393k	QNLI Acc 108k	RTE Acc 2.5k	WNLI Acc 634	AX MCC	Score
BiLSTM+ELMo+Attn	36.0	90.4	84.9/77.9	75.1/73.3	64.8/84.7	76.4/76.1	-	56.8	65.1	26.5	70.5
Singletask Pretrain Transformer	45.4	91.3	82.3/75.7	82.0/80.0	70.3/88.5	82.1/81.4	-	56.0	53.4	29.8	72.8
GPT on STILTs	47.2	93.1	87.7/83.7	85.3/84.8	70.1/88.1	80.8/80.6	-	69.1	65.1	29.4	76.9
BERT _{LARGE}	60.5	94.9	89.3/85.4	87.6/86.5	72.1/89.3	86.7/85.9	92.7	70.1	65.1	39.6	80.5
MT-DNN _{no-fine-tune}	58.9	94.6	90.1/86.4	89.5/88.8	72.7/89.6	86.5/85.8	93.1	79.1	65.1	39.4	81.7
MT-DNN	62.5	95.6	91.1/88.2	89.5/88.8	72.7/89.6	86.7/86.0	93.1	81.4	65.1	40.3	82.7
BERT _{BASE}	52.1	93.5	88.9/84.8	87.1/85.8	71.2/89.2	84.6/83.4	90.5	66.4	65.1	34.2	78.3
ST-DNN_{ours}	50.8	92.9	86.6/81.0	83.5/81.8	70.1/88.3	82.1/82.4	90.8	63.9	65.1	34.4	76.8
MT-DNN_{ours} no-fine-tune	44.8	90.8	87.4/82.5	83.8/84.3	69.2/87.3	81.5/81.0	90.0	68.9	65.1	31.0	76.5
MT-DNN_{ours}	52.8	92.9	89.0/85.0	88.1/87.3	70.5/88.4	82.8/81.9	90.8	70.8	65.1	34.7	78.7
Human	66.4	97.8	86.3/80.8	92.7/92.6	59.5/80.4	92.0/92.8	91.2	93.6	95.9	-	87.1

Table: GLUE test set results



DOMAIN ADAPTATION

ADAPTATION WITH SMALL DATASETS

Model	0.1%	1%	10%	100%
-------	------	----	-----	------

SNLI Dataset (Dev Accuracy%)

#Training Data	549	5,493	54,936	549,367
BERT	52.5	78.1	86.7	91.0
MT-DNN	82.1	85.2	88.4	91.5
MT-DNN^{ours}	81.2	85.0	87.5	91.2

SciTail Dataset (Dev Accuracy%)

#Training Data	23	235	2,359	23,596
BERT	51.2	82.2	90.5	94.3
MT-DNN	81.9	88.3	91.1	95.7
MT-DNN^{ours}	83.6	86.6	92.1	95.1



DOMAIN ADAPTATION

BENCHMARK

Model	Dev	Test
SNLI Dataset (Accuracy%)		
GPT	-	89.9
BERT _{LARGE}	91.7	91.0
MT-DNN _{LARGE}	92.2	91.6
BERT _{BASE}	91.0	90.8
MT-DNN _{BASE}	91.5	91.1
MT-DNN_{ours}	91.2	91.0

SciTail Dataset (Accuracy%)		
GPT	-	88.3
BERT _{LARGE}	95.7	94.4
MT-DNN _{LARGE}	96.3	95.0
BERT _{BASE}	94.3	92.0
MT-DNN _{BASE}	95.7	94.1
MT-DNN_{ours}	95.1	93.6

CONCLUSIONS



CONCLUSIONS



- ▶ In this work we only manage to reproduce MT-DNN results with BERT_{BASE} as the transformer encoder layer
- ▶ In the original work the results were better due to the use of BERT_{LARGE} as the transformer encoder layer
- ▶ We demonstrated that, even in our configuration, Multi-Task learning augments the generalization capabilities of language models and improves performance wrt single task models
- ▶ In domain adaptation tests such improved generalization helps the model to be effective also with a small fraction of the original dataset

The background of the slide features a large, light blue watermark of the University of Florence seal. The seal is circular, depicting a seated figure (likely a saint or scholar) holding a book and a staff, with the Latin text 'FLORENTINA ACADEMIA STUDIORUM' around the top and 'UNIVERSITAS' at the bottom.

Code available at
<https://gitlab.com/reddeadrecovery/mt-bert>



REFERENCES I

- ▶ Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015).
A large annotated corpus for learning natural language inference.
In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics.

- ▶ Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019).
Bert: Pre-training of deep bidirectional transformers for language understanding.

- ▶ Khot, T., Sabharwal, A., and Clark, P. (2018).
Scitail: A textual entailment dataset from science question answering.

- ▶ Liu, X., Duh, K., and Gao, J. (2018).
Stochastic answer networks for natural language inference.
CoRR, abs/1804.07888.



REFERENCES II

- ▶ Liu, X., Gao, J., He, X., Deng, L., Duh, K., and Wang, Y.-y. (2015). Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 912–921, Denver, Colorado. Association for Computational Linguistics.
- ▶ Liu, X., He, P., Chen, W., and Gao, J. (2019). Multi-task deep neural networks for natural language understanding.
- ▶ Liu, X., Shen, Y., Duh, K., and Gao, J. (2017). Stochastic answer networks for machine reading comprehension. *CoRR*, abs/1712.03556.
- ▶ Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *CoRR*, abs/1706.03762.

REFERENCES III



- Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2019).
GLUE: A multi-task benchmark and analysis platform for natural language understanding.
In the Proceedings of ICLR.