Bag-of-Words Transfer: Non-Contextual Techniques for Multi-Task Learning

Seth Ebner, Felicity Wang, Benjamin Van Durme **Johns Hopkins University**

Introduction Many architectures for multi-task learning (MTL) have been proposed to take advantage of transfer among tasks, often involving complex models and training procedures. We ask if the sentence-level representations learned in previous approaches provide significant benefit beyond that provided by simply improving word-based representations. To investigate the question, we consider three **Bag-of-Words Techniques** in **multi-task learning** on the tasks of sentiment analysis and textual entailment.

Unigram Generative Regularization

Reconstruct input using a language model conditioned on the label

- Uses no additional data
- Related to corresponding discriminative classification task
- Realized as an auxiliary loss term

For an arbitrary encoder network $q_{\phi_t}(y \mid x)$ and decoder network $p_{\boldsymbol{\theta}}(x \mid h)$, the loss function $\mathcal{L}_{\text{GMTL}}$ on a single example i *for dataset* t *is*:

$$-[\alpha_t \log q_{\phi_t}(y_i^{(t)} \mid x_i^{(t)}) + \beta_t \log p_{\boldsymbol{\theta}}(x_i^{(t)} \mid h_i^{(t)})]$$

- $x_i^{(t)}$, $y_i^{(t)}$: input and its label
- α_t , β_t : discriminative and reconstruction task weights
- $h_i^{(t)}$: conditioning vector for *controllable* text generation of the second sequence $\boldsymbol{x}_2$ $\boldsymbol{h} := [\boldsymbol{t}, \boldsymbol{y}', \boldsymbol{\pi}_1]$
- $\boldsymbol{t}$: one-hot encoding of the task index
- $\boldsymbol{y}' = \mathbf{L}_t \boldsymbol{y}$: task-specific label projection transforming potentially disparate label spaces of different sizes to the same $\mathbf{L}_t \in \mathbb{R}^{l \times |\mathcal{Y}_t|}$
- : trainable task-specific parameters $\boldsymbol{\pi}_1$ $\boldsymbol{x}_1$
- : input encoding of the first sequence $\boldsymbol{x}_2$, on which we condition of the reading of

Pooling Encoder (DAN)

Deep Averaging Network (Iyyer et al., 2015)

- Competitive performance to LSTMs and CNNs on textual similarity, textual entailment, and sentiment classification
- Syntactically oblivious
- Fast and small

Dataset	Model	Epoch	# Params.	Metric
MultiNLI$^{2.5\%}$	ARS (r)	268 s (C)	362,608	**49.20**
	DAN	35 s (C)	241,408	47.69
Topic-5	ARS (r)	93 s (G)	423,918	0.914
	DAN	75 s (C)	362,718	**0.900**

Table 1: Comparisons of mean training epoch times and number of trainable architecture parameters(i.e., trainable non-word-embedding parameters) in the reimplemented ARS model and the DAN model in the MTL setting for the MultiNLI and Topic-5 datasets. (C) denotes time run on a CPU, (G) denotes time run on a GPU.

Pre-trained Word Embeddings

GloVe (Pennington et al., 2014)

- Transfer learning: embeddings derived from 6B tokens of English from Wikipedia and Gigaword
- Type-level, non-contextual representations
- Good initialization for word embeddings

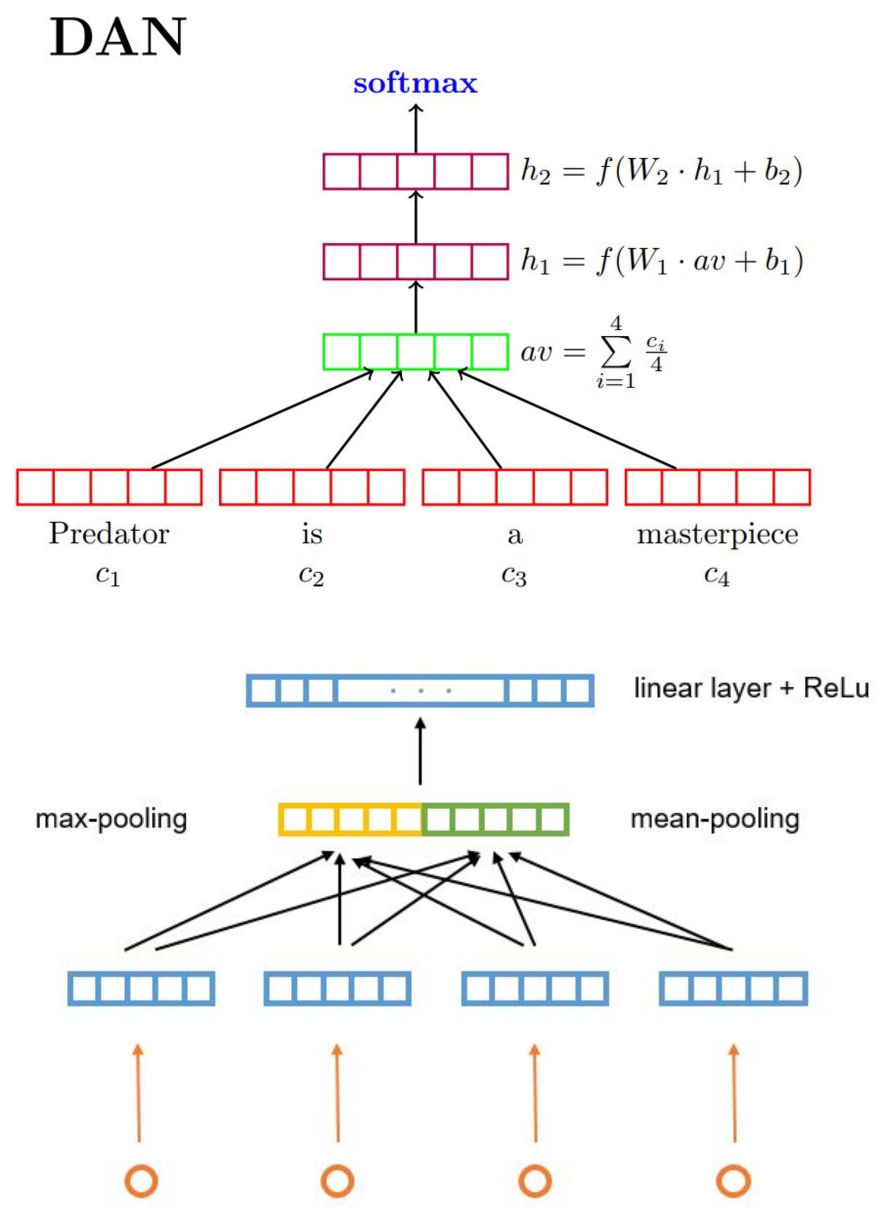

Figure 1: Original DAN model and our modification: We use the concat[mean-pooling, max-pooling] and then a linear projection and ReLu activation, with a word dropout rate of 0.1

Datasets

Following (Augenstein et al., 2018), we experiment with 8 two-sequence-input text classification datasets.

Dataset	# Labels	# Train	Seq 1	Seq 2	Task	Auxiliary tasks
MultiNLI$^{2.5\%}$	3	10,001	Hypothesis	Premise	Natural language inference	Topic-5
ABSA-L	3	2,618	Aspect	Review	Aspect-based sentiment analysis, laptop domain	Topic-5
ABSA-R	3	2,256	Aspect	Review	Aspect-based sentiment analysis, restaurant domain	Topic-5, ABSA-L, Target
Target	3	5,623	Target	Text	Target-dependent sentiment analysis	FNC-1, MultiNLI$^{2.5\%}$, Topic-5
Stance	3	3,209	Target	Tweet	Stance detection	FNC-1, MultiNLI$^{2.5\%}$, Target
Topic-2	2	5,177	Topic	Tweet	Topic-based sentiment analysis, binary	FNC-1, MultiNLI$^{2.5\%}$, Target
Topic-5	5	7,236	Topic	Tweet	Topic-based sentiment analysis, fine-grained	FNC-1, MultiNLI$^{2.5\%}$, ABSA-L, Target
FNC-1	4	39,741	Headline	Document	Fake News Detection	

Table 2: Size of label set, number of training examples, content of sequences, task description and auxiliary tasks of each dataset.

Results

	MultiNLI$^{2.5\%}_{\uparrow}$	ABSA-L$_{\uparrow}$	ABSA-R$_{\uparrow}$	Target$_{\uparrow}$	Stance$_{\uparrow}$	Topic-2$_{\uparrow}$	Topic-5$_{\downarrow}$
Metric	Acc	Acc	Acc	F_1^M	F_1^{FA}	ρ^{PN}	MAE^M
ARS STL (baseline)	49.25	76.74	67.47	64.01	41.1	63.92	0.919
ARS MTL (baseline)	49.39	74.94	82.25	65.73	44.12	80.74	0.859
ARS MTL (best)	49.94*	75.66*†	83.71*†	66.42*	46.26*	80.74	0.803*†
ARS STL (r)	47.71	73.16	72.99	62.44	25.05	63.91	0.903
ARS MTL (r)	49.20	75.03	79.39	63.61	29.30	61.26	0.914
STL DAN (w)	38.82	**74.03**	80.79	63.35	34.31	64.15	0.907
GSTL DAN (w)	41.70	73.53	78.58	63.45	**35.17**	65.09	0.906
MTL DAN (w)	**47.69**	**74.03**	79.86	61.44	31.77	65.42	0.900
MTL DAN + GloVe (w)	43.04	68.91	**81.84**	**63.53**	30.96	**67.85**	**0.856**
GMTL DAN (w)	39.35	69.29	78.23	61.95	25.70	59.88	0.927
GMTL DAN + GloVe (w)	40.41	69.29	80.21	63.01	26.36	61.17	0.958

Table 3: Test results. Acc: accuracy; F_1^M: macro-averaged F_1; F_1^{FA}: macro-averaged F_1 of "favour" and "against" classes; ρ^{PN}: macro-averaged recall, averaged across topics; MAE^M: macro-averaged mean absolute error, averaged across topics. ↑/↓ next to each task name indicates that higher/lower score is better. "STL": single-task setting; "MTL": multi-task setting; "(r)": reimplementation of baseline bi-directional RNN model from ARS (no Label Embedding Layer or Label Transfer Network). *: model uses LEL; †: model uses LTN. Models using only BOW representations are marked with (w). Best results from BOW experiments (bottom section) are **bolded**.

Conclusions

- BOW Techniques often outperform baseline, competitive with best ARS models
- DAN encoder facilitates transfer across tasks
- GloVe embeddings serve as good initialization
- DAN encoder is fast to train compared to bi-RNN
- Unigram Generative Regularization often improves STL performance but hurts MTL
  - Training with similar datasets is more helpful using UGR
  - But additional datasets are not always available

TFMTL

Try out our codebase TFMTL, a flexible, general, **T**ensor**F**low-based **M**ulti-**T**ask **L**earning full-pipeline framework for text classification tasks on Github! Simply modify configurations in a JSON file and everything else (dataset downloading, preprocessing, architectures, auxiliary tasks, hyper-parameters, etc.) is taken care of.

TFMTL@GitHub