Deep Automated Multi-task Learning

Davis Liang
d1liang@ucsd.edu
University of California, San Diego

Yan Shu
yashu@ucsd.edu
University of California, San Diego

Abstract

Multi-task learning (MTL) has recently contributed to learning better representations in service of various NLP tasks. MTL aims at improving the performance of a primary task, by jointly training on a secondary task. This paper introduces automated tasks, which exploit the sequential nature of the input data, as secondary tasks in an MTL model. We explore next word prediction, next character prediction, and missing word completion as potential automated tasks. Our results show that training on a primary task in parallel with a secondary automated task improves both the convergence speed and accuracy for the primary task. We suggest two methods for augmenting an existing network with automated tasks and establish better performance in topic prediction, sentiment analysis, and hashtag recommendation. Finally, we show that the MTL models can perform well on datasets that are small and colloquial by nature.

1 Introduction

Recurrent neural networks have demonstrated formidable performance in NLP tasks ranging from speech recognition (Hinton et al., 2012) to neural machine translation (Bahdanau et al., 2014; Wu et al., 2016). In NLP, multi-task learning has been found to be beneficial for *seq2seq* learning (Luong et al., 2015; Cheng et al., 2016), text recommendation (Bansal et al., 2016), and categorization (Liu et al., 2015).

Despite the popularity of multi-task learning, there has been little work done in generalizing the application of MTL to all sequential tasks. To accomplish this goal, we use the concept of automated tasks. Similar work in multi-task learning frameworks proposed in (Liu et al., 2016) and (Luong et al., 2015) are both trained on multiple labeled datasets. Though we have seen evidence of research using external unlabeled datasets in pretraining (Dai and Le, 2015) and semi-supervised multi-task frameworks (Ando and Zhang, 2005), to our knowledge there is no work dedicated to using tasks derived from the original dataset in multi-task learning with deep recurrent networks. With automated tasks, we are able to use MTL for almost any sequential task.

We present two ways of using automated multi-task learning: (1) the MRNN, a multi-tasking RNN where the tasks share an LSTM layer, and (2) the CRNN, a cascaded RNN where the network is augmented with a concatenative layer supervised by the automated task. Examples of either network are shown in Figure 1.

In summary, our main contributions are:

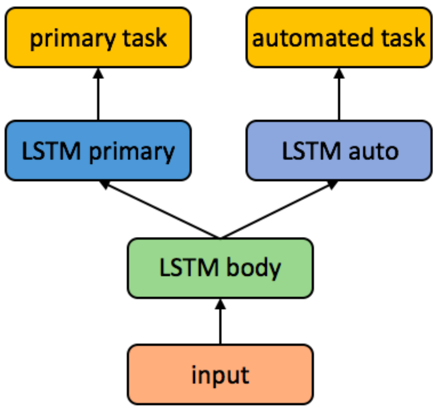
- We introduce the concept of automated tasks for multi-task learning with deep recurrent networks.
- We show that using the CRNN and the MRNN trained in parallel on a secondary automated task allows the network to achieve better results in sentiment analysis, topic prediction, and hashtag recommendation.

2 Automated Multi-task Learning

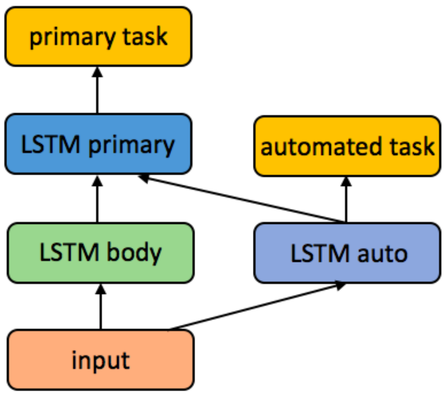
We generalize multi-task learning by incorporating automated tasks with our two MTL models: the CRNN and the MRNN. In the following subsections, we describe the automated tasks, the models, and their respective training methods.

2.1 Automated Tasks

The set of automated tasks we suggest include (1) next word prediction, (2) next character predic-



(a)



(b)

Figure 1: MRNN (a) and CRNN (b) model

tion, and (3) missing word completion.

For word and character generation, we trained a language model to predict the next word or character given the words or characters from the previous K steps. For the missing word completion task, we removed a random non-stop-word from each document and replaced it with a UNK placeholder. The removed word is fed into a word2vec model trained on Google News (Le and Mikolov, 2014) and the resulting vector is the target. We performed regression to minimize the mean squared error of predicting the missing word vector given the text. We generated predictions by finding the target word vector with the highest cosine similarity to the output vector.

2.2 MRNN

The multi-tasking RNN, MRNN, is an MTL model that we use to train our primary and automated tasks in parallel. The MRNN's initial layers are shared, and the later layers branch out to separate tasks. A basic example of an MRNN is shown in Figure 1.

The MRNN is constructed such that the primary task and automated task(s) share a body of units. This body is supervised by both the primary and automated task(s) and learns internal representations for both tasks.

2.3 CRNN

(Søgaard and Goldberg, 2016) showed that a higher-level task can benefit from making use of a shared representation learned by training on a lower-level task. Similarly, the CRNN assumes that the primary task has a hierarchical relationship with the automated task. A basic example of a CRNN is shown in Figure 1.

Specifically, we designed the CRNN to use the representations learned from an automated task as a concatenative input (Ghosh et al., 2016; Lipton et al., 2015) for the primary task. Furthermore, such a model can be supervised on an identical task at different network layers.

3 Experiments

We evaluate the performance of our models on binary sentiment analysis of the Rotten Tomato Movie Review dataset, topic prediction on the AG News dataset, and hashtag recommendation on a Twitter dataset. For each of these datasets, we compared the results from the MRNN and CRNN to a corresponding LSTM model. We separately tuned the hyper-parameters for each model with the validation sets and took the average results across the multiple runs. Note that the baseline LSTM models are 2-layered. Our MTL models and the LSTM baseline have the exact same number of parameters along the primary task stream.

In the following experiments, we use 512 LSTM cells for all models trained on the Rotten Tomato dataset and 128 LSTM cells for the AG News and Twitter datasets. Before each output layer, we have a single fully connected layer consisting of 512 hidden units for the Rotten Tomato dataset and 128 hidden units for the AG News and Twitter datasets. We use a batch size of 128 and apply gradient clipping with the norm set to 1.0 on all the parameters for all experiments.

We found that missing word completion is especially detrimental to our MTL models. We believe that removing a word from each document, which consists almost exclusively of short sequences, discards a large portion of the useful information.

Dataset	Doc. Count	Categories	Avg. WC
RTMR	10662	2	20
AGNews	127600	4	34
Twitter	5964	71	70*

Table 1: Dataset statistics. (*character count)

Thus, the quantitative results of the missing word completion experiments have been omitted from this paper. We hypothesize that missing word completion is more useful for datasets with longer documents where discarding individual words will not have a major effect on each document.

3.1 Data

The Rotten Tomato Movie Review (RTMR)[1] (Pang and Lee, 2005) dataset consists of 5331 positive and 5331 negative review snippets. The task is to predict review sentiment. The dataset is randomly split into 90% for the training and validation sets and 10% for test set (Dai and Le, 2015).

The AG News[2] (Zhang et al., 2015) dataset consists of 120,000 training and 7,600 testing documents. The task is to classify the documents into one of four topics. Following (Wang and Tian, 2016), we took 18,275 documents from the training set as validation data.

The Twitter dataset consists of 5,964 tweets. The task is to predict one of the 71 hashtag labels. We collected 300,000 tweets using the Twitter API. We removed all retweets, URLs, uncommon symbols, and emojis. We lowercased all the characters in the tweets. We kept the tweets with the 71 most popular English hashtags, and removed the hashtags from the tweets. We used an 80/10/10 split of the remaining data. Although Twitter's Developer Policy prevents us from releasing the dataset, the entire data collection pipeline will be made available upon publication.

4 Rotten Tomatoes

4.1 Training Details

The primary task for the Rotten Tomatoes dataset is sentiment analysis. We used word generation as the automated task. The input is a 300-dimensional word2vec vector for each word. The primary task output consists of two softmax units, representing a positive or negative review. The automated task output is next word prediction of the word2vec representation, and hence is a 300 unit tanh layer. For LSTM we use a learning rate of 0.0001. For the MTL models, we need to tune the learning rate hyper-parameter of the automated task. Instead of tuning the primary and automated task hyper-parameters separately, we found an alternative method for tuning the learning rates using the following equation where lr_{actual} is the only learning rate hyper-parameter. lr_{actual} is optimized on the validation set.

$$lr_{\text{prim}}(\text{epoch}) = \text{epoch} * \left(\frac{lr_{\text{actual}}}{\text{total Epochs}}\right) \qquad (1)$$
$$lr_{\text{auto}}(\text{epoch}) = lr_{\text{actual}} - lr_{\text{prim}}(\text{epoch})$$

We apply this type of learning rate modulation in order to simulate network pre-training on the automated task in the earlier epochs, learn shared representations in the intermediate epochs through multi-task learning, and train more exclusively on the primary task during the later epochs. We used an lr_{actual} of 0.01.

The MTL and LSTM models both use word-level word2vec representations trained on Google News (Le and Mikolov, 2014). The primary sentiment analysis task is trained using Adam optimizer (Kingma and Ba, 2014) on cross-entropy loss while the automated word generation task is trained using mean-squared error. We continue to use Adam optimizer in the rest of our experiments.

4.2 Results

We compare our experimental results with (1) SA-LSTM (Dai and Le, 2015), an LSTM initialized with a sequence auto-encoder, and (2) the adversarial model (Miyato et al., 2016), an LSTM-based text classification model with perturbed embeddings. We choose these two models because they are both LSTM-based and are thus comparable to our models. Non-LSTM models, such as convolutional neural networks, have been able to achieve higher accuracy on sentiment analysis with the Rotten Tomatoes dataset (Kim, 2014). All of our networks beat the variant of the SA-LSTM that does not use outside data for pre-training. However, the adversarial (Miyato et al., 2016) and SA-LSTM (Dai and Le, 2015) models, using external unlabeled datasets, outperform our MTL models. With the MRNN, we achieve a 1.5% gain in accuracy over SA-LSTM, and 1% over the vanilla LSTM network. With the CRNN, we achieve similar results compared to the vanilla LSTM network. We hypothesize that the reason the CRNN under-performs the MRNN is due to the lack of a clear hierarchy between sentiment analysis and

[1] cs.cornell.edu/people/pabo/movie-review-data/
[2] di.unipi.it/gulli/AG_corpus_of_news_articles.html

Dataset	Model	Accuracy
RTMR	SA-LSTM (2015)	79.7%
RTMR	SA-LSTM (2015)*	83.3%
RTMR	Adversarial (2016)*	83.4%
RTMR	LSTM	80.2%
RTMR	**CRNN**	**80.1%**
RTMR	**MRNN**	**81.2%**
AGNews	SC-LSTM-I (2016)	92.05%
AGNews	LSTM	91.59%
AGNews	**CRNN**	**92.19%**
AGNews	**MRNN**	**91.93%**
Twitter	LSTM	57.8%
Twitter	**CRNN**	**61.4%**
Twitter	**MRNN**	**62.0%**

Table 2: Experimental results. (*trained on external unlabeled dataset)

word generation. We suspect that sentiment analysis is primarily keyword based and cannot fully take advantage of the automated language model task. Additionally, we found that the MTL models can be trained with much higher learning rates than a standard LSTM, allowing for convergence in many fewer epochs. The MRNN model converged within the first 10 epochs, whereas the LSTM model required approximately 30 epochs to converge.

5 AG News

5.1 Training Details

For the AG News experiment, the primary task is topic prediction and the automated task is word generation. The input to the model is the 300-dimensional word2vec representations of the words from the documents. The primary task output uses a softmax layer with 4 units. The automated task output is represented by a tanh layer with 300 units. The learning rate for the LSTM is 0.001. For the MRNN, the learning rates undergo the same linear function as in the Rotten Tomatoes experiment where lr_{actual} is 0.01.

5.2 Results

For AG News dataset, we compare our experiment result with skip-connected LSTM (Wang and Tian, 2016), the previous state-of-the-art model on this dataset. The CRNN outperforms state-of-the-art by 0.14% and MRNN by 0.26%. We believe the CRNN beats the MRNN due to a hierarchical relationship between topic prediction and word generation. We suspect that topic prediction, which relies on a holistic understanding of a document, can effectively take advantage of the language model.

6 Twitter

We ran an experiment showing that our models can perform well in challenging environments with little data. We used a small dataset of 5,964 tweets. We performed regression on the word2vec representation of the hashtag given the tweet. We chose regression over classification of one-hot targets because our chosen hashtags are inherently non-orthogonal and can benefit from semantic representations in vector space. We trained three models: an LSTM model, the MRNN, and the CRNN.

6.1 Training Details

For the Twitter experiment, the primary task is hashtag recommendation and the automated task is character prediction. We use character prediction as the automated task due to the large amount of misspellings and colloquialisms in tweets.

The input to the model is the 66-dimensional one-hot encoding of the characters corresponding to the ASCII characters that we kept during preprocessing. The primary task output is a tanh layer with 300 units. The automated task output uses a softmax layer with 66 units. For all the models we chose a fixed learning rate of 0.001 based on our observation that different learning rates have little effect on the relative trend between the models on this particular task. A constant, equal learning rate allows us to compare the accuracy curves of each network against epochs run.

Since several of the hashtags are very similar to each other (i.e. #Capricorn and #Scorpio), we marked a prediction as correct if the predicted semantic vector's top 5% (top 4) closest cosine distance words contained the target hashtag.

6.2 Results

With the MRNN, we achieve a 4.2% gain in accuracy over the LSTM in the Twitter dataset. With the CRNN, we achieve a 3.6% gain in accuracy. Additionally, we have shown in Figure 2 that both the MRNN and CRNN models converge faster than the LSTM model; both MTL models take approximately half of the number of epochs to reach 50% accuracy using the same constant learning rate.

7 Conclusion

In this paper, we showed that automated multitask learning models can consistently outperform the LSTM in sentiment analysis, topic prediction,

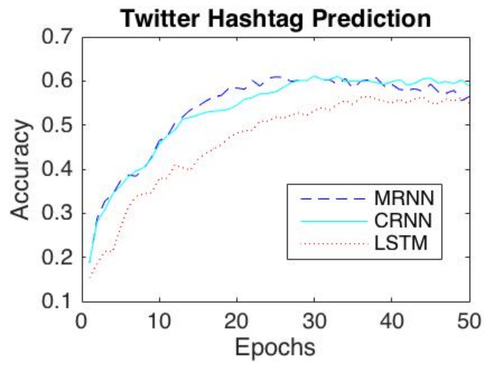


Figure 2: Hashtag prediction in Twitter.

and hashtag recommendation. Note that the concept of automated tasks can be extended to non-NLP sequence tasks such as image categorization with next row prediction as the automated task. Because automated MTL can be integrated into an existing network by adding a new branch to a pre-existing graph, we can substitute bidirectional LSTMs (Schuster and Paliwal, 1997), GRUs (Gulcehre et al., 2014), and vanilla RNNs for LSTMs in our MTL models. We will experiment on these variations in the future.

References

R. Ando and T. Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Trapit Bansal, David Belanger, and Andrew McCallum. 2016. Ask the gru: Multi-task learning for deep text recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 107–114. ACM.

Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Semi-supervised learning for neural machine translation. *arXiv preprint arXiv:1606.04596*.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, pages 3079–3087.

Shalini Ghosh, Oriol Vinyals, Brian Strope, Scott Roy, Tom Dean, and Larry Heck. 2016. Contextual lstm (clstm) models for large scale nlp tasks. *arXiv preprint arXiv:1602.06291*.

Caglar Gulcehre, Kyunghyun Cho, Razvan Pascanu, and Yoshua Bengio. 2014. *Learned-norm pooling for deep feedforward and recurrent neural networks*, part 1 edition, volume 8724 LNAI of *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Springer Verlag.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

P. Kingma and J. Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196.

Zachary C Lipton, Sharad Vikram, and Julian McAuley. 2015. Generative concatenative nets jointly learn to write and classify reviews. *arXiv preprint arXiv:1511.03683*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Deep multi-task learning with shared memory. *arXiv preprint arXiv:1609.07222*.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *HLT-NAACL*, pages 912–921.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*.

Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2016. Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 115–124. Association for Computational Linguistics.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual*

Meeting of the Association for Computational Linguistics, volume 2, pages 231–235. Association for Computational Linguistics.

Yiren Wang and Fei Tian. 2016. Recurrent residual learning for sequence classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, page 938943. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.