UNIVERSITY OF CALIFORNIA, SAN DIEGO

Automated Multi-task Learning

A Thesis submitted in partial satisfaction of the
requirements for the degree
Masters of Science

in

Computer Science

by

Davis Liang

Committee in charge:

Professor Garrison Cottrell, Chair
Professor Julian McAuley
Professor Lawrence Saul

2017

The Thesis of Davis Liang is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

Chair

University of California, San Diego

2017

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

Thanks to Yan Shu, Gary Cottrell, Ben Cipollini, and Zachary Lipton for your continued guidance in my thesis, my research at UCSD, my research at CMU, and the great many questions I may ask in the near and far future.

Chapter 1, in part, has been submitted for publication of the material as it may appear in Deep Automated Multi-Task Learning, Liang, Davis; Shu, Yan; Garrison Cottrell, EMNLP 2017. The thesis author was the primary investigator and author of this paper.

Chapter 2, in part, has been submitted for publication of the material as it may appear in Deep Automated Multi-Task Learning, Liang, Davis; Shu, Yan; Garrison Cottrell, EMNLP 2017. The thesis author was the primary investigator and author of this paper.

Chapter 3, in part, has been submitted for publication of the material as it may appear in Deep Automated Multi-Task Learning, Liang, Davis; Shu, Yan; Garrison Cottrell, EMNLP 2017. The thesis author was the primary investigator and author of this paper.

Chapter 4, in part, has been submitted for publication of the material as it may appear in Deep Automated Multi-Task Learning, Liang, Davis; Shu, Yan; Garrison Cottrell, EMNLP 2017. The thesis author was the primary investigator and author of this paper.

Chapter 5, in part, has been submitted for publication of the material as it may appear in Deep Automated Multi-Task Learning, Liang, Davis; Shu, Yan; Garrison Cottrell, EMNLP 2017. The thesis author was the primary investigator and author of this paper.

Chapter 6, in part, has been submitted for publication of the material as it may appear in Deep Automated Multi-Task Learning, Liang, Davis; Shu, Yan; Garrison

Cottrell, EMNLP 2017. The thesis author was the primary investigator and author of this paper.

VITA

2016	B. S. in Electrical Engineering *magna cum laude*, University of California, San Diego
2016-2017	Graduate Teaching Assistant, University of California, San Diego
2017	M. S. in Computer Science, University of California, San Diego

PUBLICATIONS

Davis Liang, Yan Shu, Garrison Cottrell "Automated Multi-task Learning", *EMNLP (Under Review)*, 314, 2017.

ABSTRACT OF THE THESIS

Automated Multi-task Learning

by

Davis Liang

Masters of Science in Computer Science

University of California, San Diego, 2017

Professor Garrison Cottrell, Chair

Multi-task learning (MTL) has recently contributed to learning better representations in service of various natural language (NLP) tasks. MTL aims at improving the performance of a primary task by jointly training on a secondary task. This paper introduces to the field of deep recurrent neural networks the concept of automated tasks, which exploit the sequential nature of the original input data, as secondary tasks in an MTL model. Specifically, 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. Furthermore, we suggest two methods for

augmenting an existing network with automated tasks and establish better-than-baseline performance in topic prediction, sentiment analysis, and hashtag recommendation. Finally, we show that the MTL models can perform well on datasets like Twitter that are small and colloquial by nature. We claim that because every sequential dataset has associated automated tasks, automated MTL can be generalized to learn better representations for any sequential neural network model.

Chapter 1

Introduction

Recurrent neural networks (RNN) have demonstrated state-of-the-art performance in natural language processing (NLP) tasks ranging from speech recognition [9] to machine translation [1, 26]. For NLP, multi-task learning has been found to be beneficial for tasks including but not limited to *seq2seq* learning [18, 3], text recommendation [2], and categorization [17].

Despite the popularity of multi-task learning in deep learning, there has been little work done in generalizing the application of deep MTL to all sequential tasks. In general, data corpora do not have multiple labels and thus do not fit into the usual mold for multi-task learning. Similar work in multi-task learning frameworks proposed in [15] and [18] are both trained on multiply labeled datasets. Though we have seen evidence of research using external unlabeled datasets in pre-training [6], to our knowledge there is no work dedicated to using tasks derived from the original dataset in multi-task learning. To overcome this issue, we introduce the concept of automated tasks that exploit the sequential nature of the original input data as secondary tasks in an MTL model. With automated tasks, we are able to use MTL for almost any sequential task.

In addition to creating automated tasks, we present two multi-task learning models

to demonstrate the efficacy of automated multi-task learning: (1) the MRNN, a multi-tasking RNN where the tasks share a recurrent representation, and (2) the CRNN, a cascaded RNN where the network is augmented with a concatenative layer supervised by the automated task. Examples of these networks are given in the "Models" section.

Our main contributions are:

- We introduce the concept of automated tasks for multi-task learning.
- 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.
- We present a detailed analysis of the limits of our approaches, and suggest guidelines for when to use each type of model and automated task.

Chapter 1, in part, has been submitted for publication of the material as it may appear in Deep Automated Multi-Task Learning, Liang, Davis; Shu, Yan; Garrison Cottrell, EMNLP 2017. The thesis author was the primary investigator and author of this paper.

Chapter 2

Previous Work

In this section, we will discuss previous work in deep recurrent neural network architecture, multi-task learning, concatenative nets, and Word2vec in brief.

2.0.1 Recurrent Neural Networks

Classic recurrent neural network architecture offers a method for maintaining information over consecutive sequences of data. With the addition of gates and memory cells in LSTMs, Long Short Term Memory networks, information is allowed to persist even further, avoiding the long-term dependency problem found in classic recurrent network architectures [10]. In particular, the success of these networks can be attributed to the gated memory cells and a hidden state that is not iteratively squashed over time. Likewise, GRU, Gated Recurrent Units, have a combined input and forget gate (called the update gate) and a merged cell and hidden state [4]. The resulting GRU architecture is simpler than an LSTM while preserving much of its representational power. Bidirectional LSTMs have additional LSTM units connected in the opposite direction [22]. LSTMs in the forwards direction and backwards direction have their outputs concatenated and share an input, but are otherwise disconnected. Grid LSTMs have an additional memory

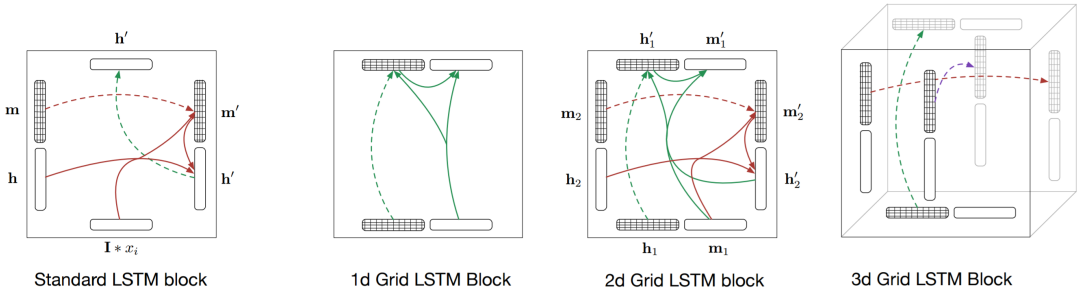

Figure 2.1: Grid LSTM. [11]

cell in each dimension [11]. For example, a two-dimensional grid LSTM will have an additional memory cell connected in the vertical depth direction. Three dimensional grid LSTMs will have two additional memory cells and so on. Figure 2.1 shows examples of the grid LSTM.

To introduce both simplicity and consistency, we use regular LSTMs in all our implementations. However, we note that any of the previously mentioned RNN architectures can be used instead.

2.0.2 Current State-of-the-art Models

In this section, we detail the architectures and results from previous work on sentiment analysis on the Rotten Tomatoes Movie Review Dataset and topic prediction on the AG News dataset.

For the Rotten Tomatoes Movie Review dataset, we focus on two models: the SA-LSTM and an adversarial model [6] [20]. For AG News, we will cover the recurrent residual model [25].

The sequence auto-encoder LSTM, SA-LSTM, is an LSTM initialized with a sequence auto-encoder which uses a standard sequence-to-sequence, or seq2seq, [24] model with the output replaced with the input sequence. A seq2seq model uses the encoding of an input sequence to decode an output sequence and is shown in 2.3. An

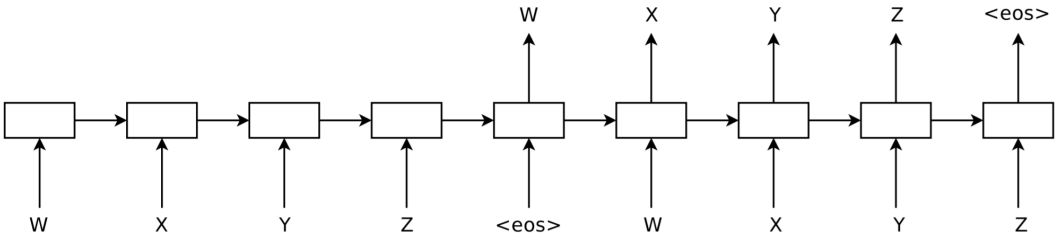

Figure 2.2: SA-LSTM. [6]

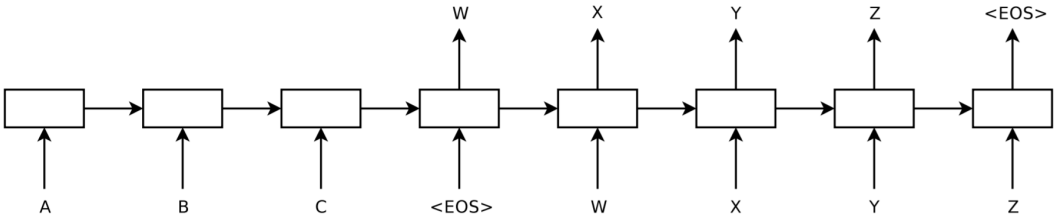

Figure 2.3: sequence-to-sequence model. [24]

example of the SA-LSTM is shown in 2.2. In their experiments, the sequence auto-encoder is trained to reproduce the full document after reading the document. The SA-LSTM exemplifies a method of pre-training to initialize hidden states and learn useful features prior to training on the actual task. This method, when paired with a large corpus of unlabeled data from Amazon reviews, achieved state-of-the-art results on sentiment analysis of the Rotten Tomatoes Movie Review dataset in 2015.

The adversarial model is the current state of the art model for sentiment analysis on the Rotten Tomatoes Movie Reviews dataset. Adversarial training helps regularize supervised learning algorithms. One shortcoming of adversarial training is that it requires making perturbations to the input vector, which is inappropriate for sparse high-dimensional inputs like the commonly used one-hot vector representations of text. Miyato et. al [20] extends adversarial training to text by applying perturbations to word embeddings rather than the original textual data.

The residual model is the current state-of-the-art model for topic prediction on

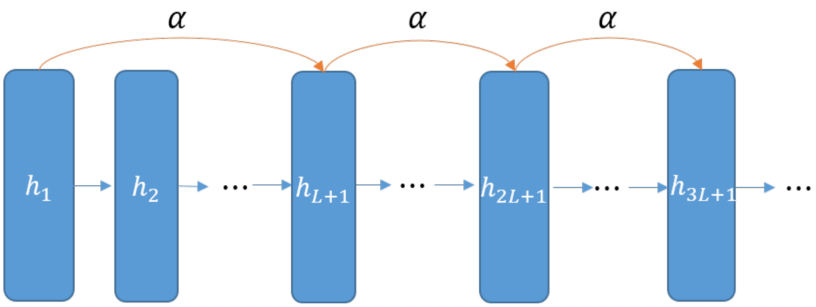

Figure 2.4: Skip-Connected LSTM. [25]

the AG News dataset. The residual RNNs have forward skip connections between non-adjacent RNN units. It has been shown that for sequence classification tasks, incorporating residual connections into recurrent structures yields similar accuracy to LSTMs with many fewer model parameters. Additionally, adding residual connections to an LSTM improves its performance above the baseline LSTM model. In theory, the addition of these skip connections allows the network to have an additional spatial shortcut paths for fluent information flow and efficient optimization for deep structures [8]. An example of the skip-connected recurrent residual network is shown in figure 2.4. In the figure, α can be set to 1 (identity mapping) or optimized as a parameter during training.

2.0.3 Multi-task Learning and Pre-training

In this section, we detail common methods for multi-task learning. First, we will explore work done in true multi-task learning where a dataset such as Amazon product reviews may have multiple labels. Next, we will explore pre-training on external datasets.

Classic MTL methods train multiple tasks in parallel with a shared representation to enforce inductive transfer, using what is learned for one problem to help another problem. Classic methods use datasets with multiple labels such as Part-Of-Speech tagging, Chunking, and Named Entity Recognition [5].

Previous work have used multiple datasets to train a network where both the

target and the inputs multiplex between datasets and tasks [16]. In this way, one is able to use multiple datasets in order to accomplish multi-task learning (as long as the input representations remain the same dimensionality). However, it was shown in the same paper that when the tasks came from the same corpus (i.e. multiply-labeled data sets), then improvements are more significant than other combinations [16].

Previous work has shown that pre-training on a language model and pre-training on a sequence autoencoder also improved network accuracy on the primary task [6]. Specifically, the network was initially trained an LSTM network as both a language model and a sequence autoencoder, using a seq2seq framework [24], before training on the primary task. Pre-training initializes the LSTM weights and hidden states and generally results in an overall improvement in network accuracy. The results show that pre-training on large external datasets allows for better results on a primary task on a separate dataset.

2.0.4 Concatenative Neural Networks

Concatenative networks have recently been shown to influence the output of recurrent neural network models by concatenating additional information to the input of the model [14]. A typical generative model, which specifies the joint probability distribution over observation and label sequences, is used for randomly generating observable data values. Adding a rating value to the input at each time-step alters the type of reviews that a generative language model produces. For example, by concatenating a high valued rating element to the input vector of a model, one can produce a review with a particularly high sentiment. In other words, concatenation of meaningful values to the input of a model gives additional information about what kind of output should be generated. An example of a concatenative model is shown in figure 2.5.

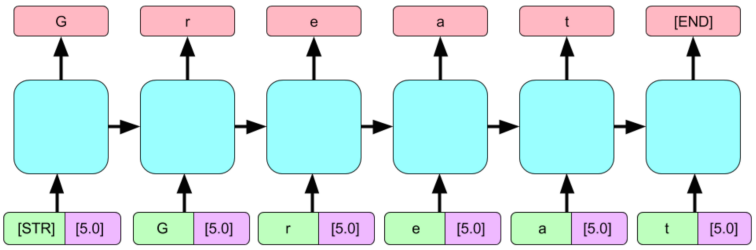

Figure 2.5: Concatenative Model. [14]

2.0.5 Word2Vec

Word2vec is a method for taking a corpus of text and learning vector representations for each word [13]. Word2vec uses a model that consists of an input layer that takes in the one-hot encoding of a word in the corpus, a hidden layer with linear units, and an output layer with softmax units that represents a word chosen within a certain window of the input word [19]. While the word2vec model learns a mapping from the input word to the output word, the input-to-hidden layer learns semantic vector embeddings for the input words. This method of assigning vector representations to each unique word in a text corpus results in a meaningful placement of words in a lower-dimensional word-vector space. A classic example demonstrates this by taking the learned word vector for king, subtracting the word vector for man, and adding the word vector for woman. The result word vector had high cosine similarity to the word vector for queen.

One obstacle for training a word2vec model is the high-dimensionality of the text corpus when represented in the usual one-hot-fashion. For example, if we wanted to map a text corpus with a vocabulary size of one million to a 300 dimensional vector space, we would have on the order of 300 million parameters.

Chapter 2, in part, has been submitted for publication of the material as it may appear in Deep Automated Multi-Task Learning, Liang, Davis; Shu, Yan; Garrison Cottrell, EMNLP 2017. The thesis author was the primary investigator and author of this paper.

Chapter 3

Automated Multi-task Learning

3.1 Automated Tasks

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

We define automated tasks as any task that exploits the sequential nature of the original input data, as secondary tasks in an MTL model. The set of automated tasks we suggest include (1) next word prediction, (2) next character prediction, and (3) missing word completion. An example of a language model is shown in 3.1.

Note that automated tasks are not limited to NLP problems and can be extended to computer vision problems. In computer vision, we have images which are represented by multi-channel matrices of pixel values. For this type of data, one possible automated task is next row prediction given the previous rows of pixels.

For word generation, we trained a language model to predict the next word given the words from the previous K steps. For this task, we use a many-to-many model where each time-step has both an input and an output prediction. The input and target words are

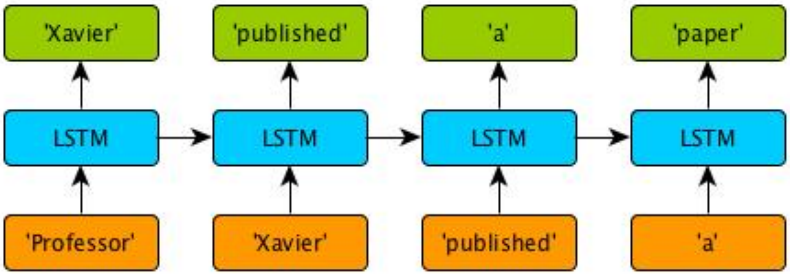

Figure 3.1: Language Model Task.

encoded as N dimensional word2vec embeddings which we generate from a word2vec model. Thus, the input and target at each time-step is of size N. For our particular experiments, we chose a word2vec model trained on Google News. The word2vec model trained on Google News encodes each word as a 300 dimensional vector.

For character generation, we trained a language model to predict the next character given the characters from the previous K steps. For this task, we also use a many-to-many model where each time-step has both an input and an output prediction. To reduce the amount of time required to train our models, we chose to use only 66 of the most common ascii characters in our input data. We preserve this heuristic throughout all our experiments. In particular, this simplified our experiment for the Twitter dataset in which there was immense variety in special characters (such as miscellaneous emojis). With this heuristic, characters are encoded in 66 dimensions and are presented as one-hot vectors.

For the missing word completion task, we removed a random non-stop-word from each document and replaced it with a plain text 'UNK' placeholder. The removed word is fed into the word2vec model trained on Google News [13] and the resulting 300-dimensional vector is used as the target. For this particular task, we use a many-to-one network where we take in the entire sequence and predict a single output (the UNK replaced word). We performed regression to minimize the mean squared error of

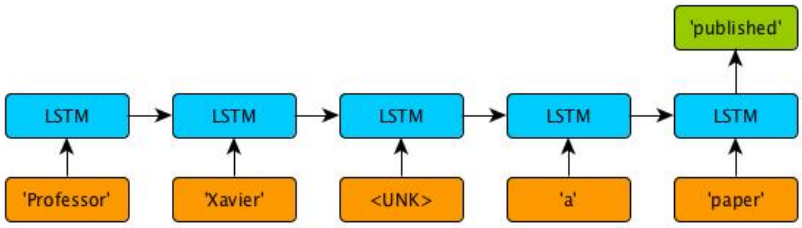

Figure 3.2: Missing Word Completion Task.

predicting the missing word vector given the text. Because word2vec vectors exist in some meaningful semantic vector space we generated predictions by finding the target word vector with the highest cosine similarity to the output vector. An example of a missing word completion model is shown in figure 3.2.

3.2 Models

In this section, we present two models: the multi-tasking RNN (MRNN), and the cascaded RNN (CRNN). These models can be created by simply adding a supervised branch to existing RNN architectures. Additionally, these models may be used for three or more tasks by adding multiple supervised branches (one for each task). Finally, we may combine the MRNN and CRNN architectures as a single multi-tasking model.

3.2.1 MRNN

The multi-tasking RNN, MRNN, is a simple MTL model that we use to train our primary and automated tasks in parallel. The MRNN's initial layers are shared and help the primary task learn more robust representations in service of the task. The shared layers are trained by both tasks. The later layers branch out to separate tasks and are trained by their respective tasks only. The training schedule we used is generated by round-robin scheduling. However, this schedule could be generated by sampling from a

Bernoulli distribution. A simple example of an MRNN is shown in figure 3.3.

Specifically, for automated MTL, we construct the simplest version of the MRNN such that the primary task and automated task(s) share an LSTM layer. Although the shared layer is supervised by both the primary and automated task(s) and learns internal representations for both tasks, the errors for each branch are back-propagated separately. This means that when we are training on the primary task, the weights in the automated branch are not updated (and vice versa).

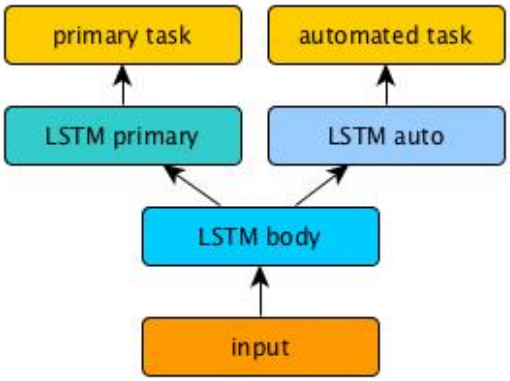

Figure 3.3: MRNN

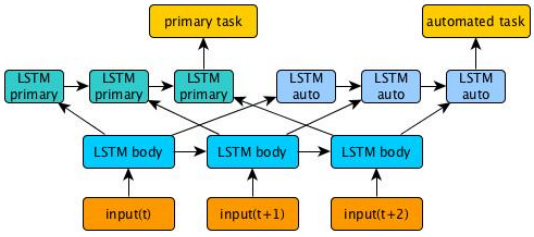

Figure 3.4: MRNN unrolled

3.2.2 CRNN

Inspired by previous work on hierarchical models [23], we present the CRNN model. The CRNN assumes that the primary task has a hierarchical relationship with the

automated task. If no such automated task can be found, the CRNN can be supervised by the same task at different layers. This hierarchical relationship allows a higher-level task to use features learned from the lower-level task. We predict that the CRNN will outperform the MRNN when there exists a clear hierarchy between the primary and automated tasks. A simple example of a CRNN is shown in figure 3.5.

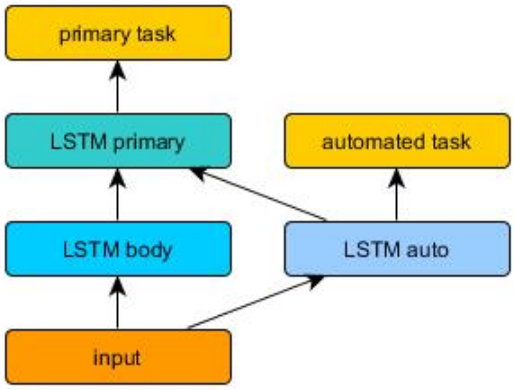

Figure 3.5: CRNN

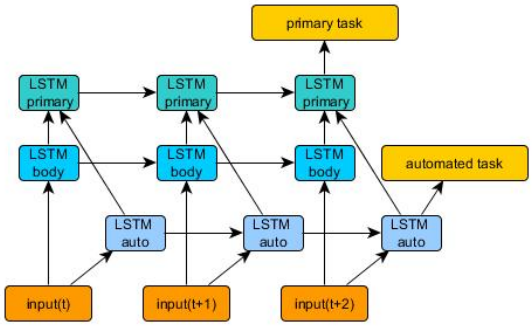

Figure 3.6: CRNN unrolled

Specifically, we designed the CRNN to use the representations learned from an automated task as a concatenative input for the primary task. The set of equations for the concatenative LSTM layer are identical to previously developed concatenative networks [7, 14]. Equation 3.1 explicitly demonstrates the concatenative properties of the LSTM body layer. We omit the equations for the other layers.

Equation 3.1 follows the convention where a_t is the output of LSTM auto (shown in 3.5) at time-step t. C_t represents the cell state at time-step t, h_t represents the output of the primary LSTM (shown in figure 3.5) at time-step t, x_t represents the input to the primary LSTM at time-step t, W represents the various gate parameters of the LSTM shared across time-steps. Essentially, our model concatenates the output of the previous time step with the input at the current time step and the hidden layer representation learned through the automated task: $[h_{t-1}, x_t, a_t]$.

$$
\begin{aligned}
f_t &= \sigma(W_f \cdot [h_{t-1}, x_t, a_t] + b_f) \\
i_t &= \sigma(W_i \cdot [h_{t-1}, x_t, a_t] + b_t) \\
\tilde{C}_t &= tanh(W_c \cdot [h_{t-1}, x_t, a_t] + b_C) \\
C_t &= f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \\
o_t &= \sigma(W_o \cdot [h_{t-1}, x_t, a_t] + b_o) \\
h_t &= o_t \odot tanh(C_t)
\end{aligned}
\tag{3.1}
$$

3.3 Multi-tasking Learning Rate Modulation

For the MTL models, we need to tune the learning rate hyper-parameter of the automated task. Instead of tuning the hyper-parameters separately, we have used an alternative method for tuning the learning rates where lr_{actual} is the only learning rate hyper-parameter. Equation 3.2 shows the values of the primary learning rate, lr_{prim} and the learning rate for the automated task lr_{auto} as functions of the current epoch.

$$
\begin{aligned}
lr_{prim}(epoch) &= epoch * (\frac{lr_{actual}}{totalEpochs}) \\
lr_{auto}(epoch) &= lr_{actual} - lr_{prim}(epoch)
\end{aligned}
\tag{3.2}
$$

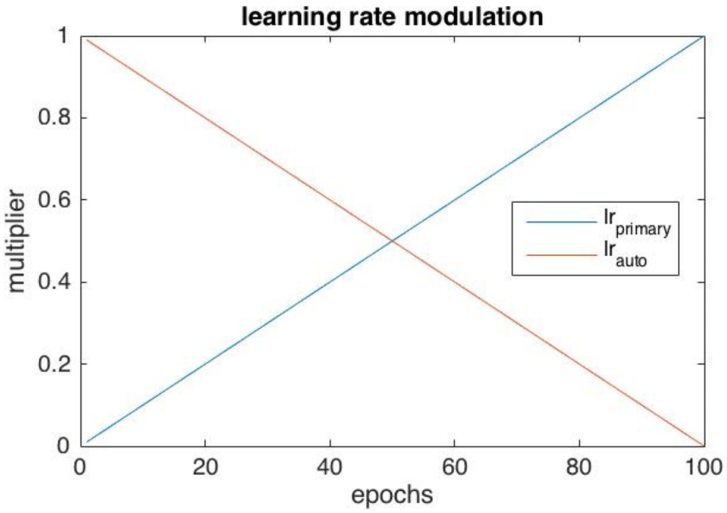

Figure 3.7: Linear Modulation.

We also explored a different form of multi-tasking learning rate modulation as shown in figure 3.8. In figure 3.8, the learning rates can take on only discrete values of 0, 0.5, and 1. This allows us to decrease the computational complexity of our automated multi-tasking models by turning off the branch with a learning rate of 0.

Either of these learning rate modulations effectively simulate network pre-training on the automated task in the earlier epochs (where the learning rate for the primary task is small and the learning rate for the automated task is large), learn shared representations in the intermediate epochs through multi-task learning (where the learning rates for both the primary and secondary tasks are comparable), and train more exclusively on the primary task during the later epochs (where the learning rate for the automated task is small and the learning rate for the primary task is large). We found that this method of training outperforms the best constant learning rates chosen from a hyper-parameter grid search.

Using linear learning rate modulation, we realized various benefits (aside from an obvious decrease in the hyper-parameter search space). First, we saw an overall increase the speed at which our networks converged to their respective optima. On average, the network experiences more than double the speed of convergence with linear learning

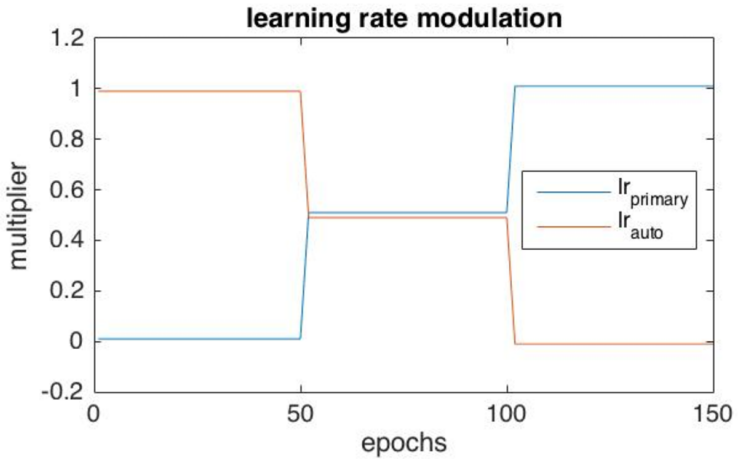

Figure 3.8: Step Modulation.

rate modulation over using the optimal constant learning rate for both the tasks. As we mentioned before, using the step learning rate modulation allows additional speedups by reducing the overall complexity of the model itself. Additionally, we witnessed a marked improvement in network performance in our sentiment analysis, topic prediction, and hashtag recommendation experiments using linear learning rate modulation.

Chapter 3, in part, has been submitted for publication of the material as it may appear in Deep Automated Multi-Task Learning, Liang, Davis; Shu, Yan; Garrison Cottrell, EMNLP 2017. The thesis author was the primary investigator and author of this paper.

Chapter 4

Experiments and Data

We ran experiments to measure the performance of our models in 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 with an equivalent number of parameters along the primary task stream.

We ran each experiment for Rotten Tomatoes 10 times, AG News 3 times, and Twitter 3 times. We separately tuned the hyper-parameters for each model with the validation sets and took the average results across the runs. Note that from our experiments, we noticed that there is not a significant difference in time per epoch between the MTL and LSTM models (both took approximately the same amount of time to run each individual epoch during training).

We found that missing word completion is especially detrimental to our MTL models. We believe that removing a word from each document in the dataset, which consists almost exclusively of short sequences, discards a large portion of the useful information. We hypothesize that missing word completion is more useful for datasets with longer documents where discarding individual words does not have a major effect

Table 4.1: Dataset statistics. (*character count)

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

Will Trump slash public funding for scientific research? 🤔🤔 #tech #science #potus
will trump slash public funding for scientific research?

wire: Dayton coach Archie Miller denies contact with #NCState 'or anyone else' #nba
wire: dayton coach archie miller denies contact with 'or anyone else'

You can't make time; so those who waste the least, achieve the most. - Tim Fargo #quote
you can't make time; so those who waste the least, achieve the most. - tim fargo

Figure 4.1: Unprocessed and Processed Tweets.

on each document.

4.1 Data

The Rotten Tomato Movie Review (RTMR) [21] dataset consists of 5331 positive and 5331 negative review snippets. The snippets have, on average, 20 words each. 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 [6].

The AG News [27] dataset consists of 120,000 training and 7,600 testing documents. The articles are, on average, 34 words long. The task is to classify the documents into one of four topics. The given topics are Sports News, World News, Business News, and Science/Technology. Following [25], we took the same 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 then kept the tweets with the 71 most popular English hashtags, and removed the hashtags from the tweets. Our heuristic for selecting the hashtags to use in our dataset is covered in depth in the Appendix. We split the remaining data into a training set with 80% of the data, a validation set with 10% of the data, and a test set with the remaining 10% of the data. Although Twitter's Developer Policy prevents us from releasing the dataset to protect user anonymity, we have made the entire data collection and processing pipeline available. Several examples of unprocessed and processed tweets are shown in figure 4.1.

Chapter 4, in part, has been submitted for publication of the material as it may appear in Deep Automated Multi-Task Learning, Liang, Davis; Shu, Yan; Garrison Cottrell, EMNLP 2017. The thesis author was the primary investigator and author of this paper.

Chapter 5

Experimental Results

In the following experiments, we use 512 LSTM cells for all models trained on the Rotten Tomato dataset and 128 LSTM cells for all models trained on 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. The output layer is a softmax layer with two units for RTMR, a softmax layer with four units for AG News, and a tanh layer with 300 units for Twitter.

5.1 Rotten Tomatoes

5.1.1 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 normalized between -1 and 1, and hence is a 300 unit tanh layer.

For the LSTM we use a learning rate of 0.0001. We use our linearly modulated learning rates where lr_{actual} is the only learning rate hyper-parameter to tune. lr_{actual} is optimized on the validation set. Finally, we use an additional exponential learning rate decay factor for lr_{actual} of 0.99999 on top of the linearly modulated learning rate.

The MTL and LSTM models both use word-level word2vec representations trained on Google News [13]. The word2vec representations are 300 dimensional vectors with both positive and negative values. The primary sentiment analysis task is trained using Adam optimization on cross-entropy loss while the automated word generation task is trained using mean-squared error [12]. Specifically, we follow general conventions in choosing cross-entropy loss for the classification task and we choose mean-squared error for the regression task.

5.1.2 Results

The results are shown in Table 5.1. All of our networks beat the SA-LSTM [6], that does not use outside data for pre-training. However, the adversarial [20] and SA-LSTM models [6], 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 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. All our MTL and classic LSTM models

share the same number of parameters along the primary stream.

5.2 AG News

5.2.1 Training Details

For the AG News experiment, the primary task is topic prediction and the automated task is next-word prediction. 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. Finally, we use an exponential learning rate decay factor for lr_{actual} of 0.99999.

5.2.2 Results

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. All our MTL and classic LSTM models share the same number of parameters along the primary stream.

5.3 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 that have at

Table 5.1: Experimental results. (*trained on external unlabeled dataset)

Dataset	Model	Accuracy
RTMR	SA-LSTM [6]	79.7%
RTMR	SA-LSTM [6]*	83.3%
RTMR	Adversarial [20]*	83.4%
RTMR	LSTM	80.2%
RTMR	**CRNN**	80.1%
RTMR	**MRNN**	**81.2%**
AGNews	Residual [25]	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%**

least one English hashtag. For each 5,964 tweets, each will have a hashtag label that corresponds to the first English hashtag. We performed regression on the word2vec representation of the hashtag given the tweet text. 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.

5.3.1 Training Details

For the Twitter hashtag recommendation 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

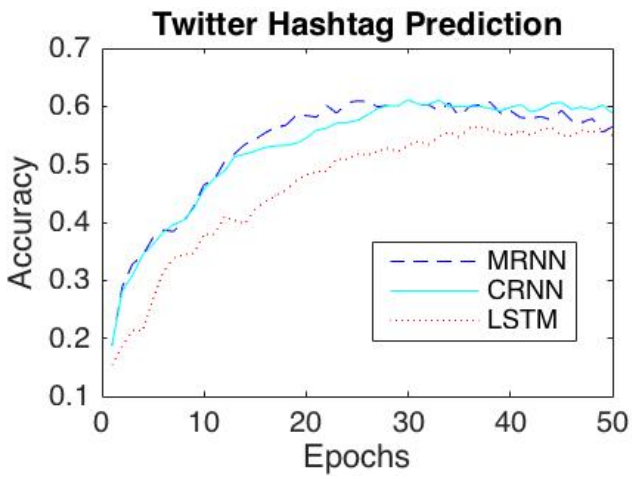

Figure 5.1: Hashtag prediction in Twitter.

corresponding to the ascii characters that we kept during preprocessing. The ascii values kept are 32 through 63 and 96 through 127, inclusive. 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 simulate a simple environment in which 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 4 closest cosine distance words contained the target hashtag.

5.3.2 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. In particular, we have shown in figure 5.1 that both the MRNN and CRNN models converge faster than the LSTM model; both MTL models reach 50% accuracy by epoch 12 whereas the LSTM

model requires about 25 epochs to achieve similar results with the same constant learning rate. This result persists across every reasonable constant learning rate that we tried. Thus, we have shown that at least for the Twitter dataset, our MTL models consistently train better and faster than the corresponding baseline LSTM model. All our MTL and classic LSTM models share the same number of parameters along the primary stream.

Chapter 5, in part, has been submitted for publication of the material as it may appear in Deep Automated Multi-Task Learning, Liang, Davis; Shu, Yan; Garrison Cottrell, EMNLP 2017. The thesis author was the primary investigator and author of this paper.

Chapter 6

Conclusion

In this paper, we showed that automated multi-task learning can improve the performance of a sequential deep learning model. Specifically, the MTL models using automated tasks consistently outperform the LSTM in sentiment analysis, topic prediction, 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. Additionally, the automated MTL models perform well with little data. Because automated MTL can be integrated into an existing network by adding a new branch to a pre-existing computational graph, we can substitute bidirectional LSTMs, GRUs, and vanilla RNNs for LSTMs in our MTL models. We will experiment on these variations in the future.

Chapter 6, in part, has been submitted for publication of the material as it may appear in Deep Automated Multi-Task Learning, Liang, Davis; Shu, Yan; Garrison Cottrell, EMNLP 2017. The thesis author was the primary investigator and author of this paper.

Chapter 7

Appendix

The accompanying table shows the hashtags that were chosen to be used in the Twitter experiment. The hashtags were chosen based on a metric that maximized the frequency of the hashtags while constraining the number of tweets associated with each hashtag to remain equivalent (for balanced categories). These hashtags all appear in the word2vec dictionary associated with Google News [19].

Our specific procedure for selecting the final corpus of tweets with the most popular hashtags is as follows. From the original corpus of tweets, we first lowercase all the alphabetical characters. We removed all tags, re-tweet headers, and URLs. Then, we removed all of the characters that were not within the ascii range (32 to 63 and 96 to 127). Then, we removed tweets with less than 60 characters. We removed all the tweets that did not have an English hashtag. From the left-over tweets, we applied a heuristic that maximized the number of usable tweets while balancing the dataset. Specifically, we chose to use tweets with hashtags with K or more occurrences. While minimizing K, we also attempted to maximize the number of categories. We solved both the minimization and maximization problem with a linear search on K. As mentioned previously, with this applied heuristic we ended up with just short of 6,000 tweets from an original corpus of

music	Scorpio	MAGA	Music	build	np
travel	browns	RT	Startups	FL	stream
beauty	Buzz	food	Sales	success	Weather
Cards	stocks	business	Hospitality	endorphins	IT
Steelers	Healthcare	health	CA	fitness	Nursing
tech	coins	win	News	GRAMMY	design
babies	auction	Movies	NBA	s	style
Sagittarius	Libra	Football	Makeup	fashion	holidays
love	quotes	TX	USA	news	Russia
deco	Trump	Retail	NFL	Follow	deals
Giveaway	marketing	privacy	free	SoundCloud	women
Capricorn	Tech	Shoes	giveaway	entrepreneur	job

Figure 7.1: Common Hashtags Used in Twitter Experiment.

over 300,000 tweets.

Bibliography

[1] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[2] T. Bansal, D. Belanger, and A. McCallum. Ask the gru: Multi-task learning for deep text recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 107–114. ACM, 2016.

[3] Y. Cheng, W. Xu, Z. He, W. He, H. Wu, M. Sun, and Y. Liu. Semi-supervised learning for neural machine translation. *arXiv preprint arXiv:1606.04596*, 2016.

[4] J. Chung, C. Gulcehre, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[5] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 160–167. ACM, 2008.

[6] A. M. Dai and Q. V. Le. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, pages 3079–3087, 2015.

[7] S. Ghosh, O. Vinyals, B. Strope, S. Roy, T. Dean, and L. Heck. Contextual lstm (clstm) models for large scale nlp tasks. *arXiv preprint arXiv:1602.06291*, 2016.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1412.03385*, 2015.

[9] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[10] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.:9, 1735-1780*, 1997.

[11] N. Kalchbrenner, I. Danihelka, and A. Graves. Grid long short-term memory. *arXiv:1507.01526*, 2015.

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

[13] Q. V. Le and T. Mikolov. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196, 2014.

[14] Z. Lipton, S. Vikram, and J. McAuley. Generative concatenative nets jointly learn to write and classify reviews. *arXiv preprint arXiv:1511.03683*, 2015.

[15] P. Liu, X. Qiu, and X. Huang. Deep multi-task learning with shared memory. *arXiv preprint arXiv:1609.07222*, 2016.

[16] P. Liu, X. Qiu, and X. Huang. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*, 2016.

[17] X. Liu, J. Gao, X. He, L. Deng, K. Duh, and Y. Wang. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *HLT-NAACL*, pages 912–921, 2015.

[18] M. Luong, V. Le, Quoc, I. Sutskever, O. Vinyals, and L. Kaiser. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*, 2015.

[19] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. *NIPS*, 2013.

[20] T. Miyato, A. Dai, and I. Goodfellow. Adversarial training methods for semi-supervised text classification. *arXiv:1605.07725*, 2016.

[21] B. Pang and L. Lee. 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, 2005.

[22] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *Signal Processing, IEE Transactions on, 45(11), 2673-2681*, 1997.

[23] A. Søgaard and Y. Goldberg. 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, 2016.

[24] I. Sutskever, O. Vinyals, and Q. Le. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, 2014.

[25] Y. Wang and F. Tian. 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, 2016.

[26] Y. Wu, M. Schuster, Z. Chen, Q. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klinger, A. Shah, J. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[27] X. Zhang, J. Zhao, and Y. LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.