# REPRESENTATION STABILITY AS A REGULARIZER FOR IMPROVED TEXT ANALYTICS TRANSFER LEARNING

**Matthew Riemer, Elham Khabiri, and Richard Goodwin**
IBM T.J. Watson Research Center
Yorktown Heights, NY, USA
{mdriemer, ekhabiri, rgoodwin}@us.ibm.com

## ABSTRACT

Although neural networks are well suited for sequential transfer learning tasks, the catastrophic forgetting problem hinders proper integration of prior knowledge. In this work, we propose a solution to this problem by using a multi-task objective based on the idea of distillation and a mechanism that directly penalizes forgetting at the shared representation layer during the knowledge integration phase of training. We demonstrate our approach on a Twitter domain sentiment analysis task with sequential knowledge transfer from four related tasks. We show that our technique outperforms networks fine-tuned to the target task. Additionally, we show both through empirical evidence and examples that it does not forget useful knowledge from the source task that is forgotten during standard fine-tuning. Surprisingly, we find that first distilling a human made rule based sentiment engine into a recurrent neural network and then integrating the knowledge with the target task data leads to a substantial gain in generalization performance. Our experiments demonstrate the power of multi-source transfer techniques in practical text analytics problems when paired with distillation. In particular, for the SemEval 2016 Task 4 Subtask A (Nakov et al., 2016) dataset we surpass the state of the art established during the competition with a comparatively simple model architecture that is not even competitive when trained on only the labeled task specific data.

## 1 INTRODUCTION

Sequential transfer learning methodologies leverage knowledge representations from a source task in order to improve performance for a target task. A significant challenge faced when transferring neural network representations across tasks is that of catastrophic forgetting (or catastrophic interference). This is where a neural network experiences the elimination of important old information when learning new information. The very popular strategy of fine-tuning a neural network involves first training a neural network on a source task and then using the model to simply initialize the weights of a target task network up to the highest allowable common representation layer. However it is highly susceptible to catastrophic forgetting, because in training for the target task it has no explicit incentive to retain what it learned from the source task. While one can argue that forgetting the source task should not matter if only the target task is of interest, our paper adds to the recent empirical evidence across problem domains (Li & Hoiem, 2016),(Rusu et al., 2016) that show additional network stability can lead to empirical benefits over the fine-tuning algorithm. It seems as though for many Deep Learning problems we can benefit from an algorithm that promotes more stability to tackle the well known stability-plasticity dilemma. One popular approach for addressing this problem is rehearsals (Murre, 1992), (Robins, 1995). Rehearsals refers to a neural network training strategy where old examples are relearned as new examples are learned. In the transfer setting it can be seen as related to multi-task learning (Caruana, 1997) where two tasks are trained at the same time, rather than sequentially, while sharing a common input encoder to a shared hidden representation. However, in rehearsals the representation is biased in favor of the source task representation through initialization. This technique is very sensible because while fine-tuning is susceptible to catastrophic forgetting, multi-task learning is not (Caruana, 1997).

One of the biggest issues with the standard rehearsals paradigm is that it requires a cached memory of training examples that have been seen in the past. This can be a massive requirement as the number of source tasks and training data sizes scale. One compelling technique for addressing this problem is the concept of pseudorehearsals (Robins, 1995), (Robins, 1996), where relearning is performed on an artificially constructed population of pseudoitems instead of the actual old examples. Unfortunately, current automatic techniques in the text analytics domain have not yet mastered producing linguistically plausible data. As such, the pseudorehearsals paradigm is likely to waste computational time that could be spent on learning realistic patterns that may occur during testing. In our work, we extend the Learning without Forgetting (LwF) paradigm of (Li & Hoiem, 2016) to the text analytics domain using Recurrent Neural Networks. In this approach, the target task data is used both for learning the target task and for rehearsing information learned from the source task by leveraging synthetic examples generated for the target task input by the model that only experienced training on the source task data. As argued by Li & Hoiem (2016), this setup strikes an important balance between classification performance, computational efficiency, and simplicity in deployment.

Regardless of whether they are applied to real source task examples, real target task examples, or synthetic examples, paradigms in the style of rehearsals all address the shortcomings of neural network forgetting by casting target task integration as a multi-task learning problem. However, this is not quite the purpose of the multi-task learning architecture, which was designed for joint learning of tasks from scratch at the same time. The key disconnect is that in multi-task learning, the transformation from the shared hidden layer to the outputs for each task are all learned and updated with the changing hidden representation. This would imply that, in the framework of rehearsals, it is possible for there to be significant changes during learning of the network's representation, and thus its abilities on the source task itself. While it would be desirable to claim we were allowing our source task network to become even better based on the target task than it was before, this motivation seems idealistic in practice. One reason this is idealistic is because multi-task learning generally only works well when tasks are sampled at different rates or alternatively given different priority in the neural network loss function (Caruana, 1997). As a result, it is most likely that auxilirary source tasks will receive less priority from the network for optimization than the target task. Additionally, we observe in our experiments, and it has been observed by others in (Rusu et al., 2015), that it is generally not possible to distill multiple complex tasks into a student network at full teacher performance for all tasks. This seems to imply the degradation of the source task performance during training is somewhat inevitable in a multi-task learning paradigm.

We address this issue with our proposed forgetting cost technique. We demonstrate that it, in fact, can be valuable to keep the hidden to output transformation of the source tasks fixed during knowledge integration with the target task. This way, we impose a stronger regularization on the hidden representation during target task integration by not allowing it to change aspects that were important to the source task's performance without direct penalization in the neural network's loss function. We demonstrate empirically both that freezing the source task specific weights leads to less deterioration in the accuracy on the source task after integration, and that it achieves better generalization performance in our setting. The forgetting cost is practical and easy to implement in training any kind of neural network. In our experiments, we explore application of the forgetting cost in a recurrent neural network to the three way Twitter sentiment analysis task of SemEval 2016 Task 4 Subtask A and find it to achieve consistently superior performance to reasonable baseline transfer learning approaches in four examples of knowledge transfer for this task.

We also demonstrate how powerful distillation can be in the domain of text analytics when paired with the idea of the forgetting cost. Significantly, we show that a high quality gazetteer based logical rule engine can be distilled using unlabeled data into a neural network and used to significantly improve performance of the neural network on the target task. This is achieved with a novel extension of the LwF paradigm by Li & Hoiem (2016) to the scenario of a source task with the same output space as the target task. This can be a very promising direction for improving the ability of humans to directly convey knowledge to deep learning algorithms. Indeed, a human defined rule can contain far more information than a single training example, as that rule can be projected on to many unlabeled examples that the neural network can learn from. This is the reason human teachers generally begin teaching human students tasks by going over core rules at the onset of learning. Moreover, we showcase that multiple expert networks trained on the target task with prior knowledge from different source tasks can be effectively combined in an ensemble and then distilled into a single GRU model (Cho et al., 2014), (Chung et al., 2014). Leveraging this combination of distillation

and knowledge transfer techniques allows us to achieve state of the art accuracy on the SemEval task with a model that performs 11% worse than the best prior techniques when trained only on the labeled data.

## 2 RELATED WORK

Since the work of (Bucilu et al., 2006) and (Hinton et al., 2015) showed that an ensemble of neural network classifier can be distilled into a single model, knowledge distillation from a teacher network to a student network has become a growing topic of neural network research. In (Ba & Caruana, 2014) it was shown that a deep teacher neural network can be learned by a shallow student network. This idea was extended in (Romero et al., 2014), where it was demonstrated that a deep and narrow neural network can learn a representation that surpasses its teacher. The use of distillation as a means of sharing biases from multiple tasks was explored in (Lopez-Paz et al., 2016), where the teacher network is trained with the output of the other tasks as input. It is not obvious how to extend a recurrent neural network to best use this kind of capability over a sequence. The idea of distilling from multiple source task teachers into a student network was highlighted in the reinforcement learning setting in (Rusu et al., 2015). Additionally, the concept of using distillation for knowledge transfer was also explored in (Chen et al., 2015), where function preserving transformations from smaller to bigger neural network architectures were outlined. This technique could also provide value in some instances for our approach where wider or deeper neural networks are needed for the task being transferred to than was needed for the original task. Distillation over target task data was first proposed as a means of elevating catastrophic forgetting in sequential knowledge transfer as applied to image classification in (Li & Hoiem, 2016). We extend this approach for its first application to our knowledge for text analytics problems, with a recurrent neural network architecture, and in the setting where the source task and target task have the same output. The chief distinction of our proposed forgetting cost is that source task specific parameters are held fixed during integration with the target task as opposed to the joint training of all parameters used by Li & Hoiem (2016). Our experiments empirically support the intuition that freezing these parameters leads to greater retention of source task performance after target task integration and better generalization to the target task.

An ensemble over multiple diverse models trained for the same sentiment analysis task was also considered in (Mesnil et al., 2014) for the IMDB binary movie reviews sentiment dataset (Maas et al., 2011). We tried this ensemble model in our work and found that it gave very limited improvement. Our ensemble technique learns a more powerful weighted average based on the soft targets of each task and a multi-step greedy binary fusion approach that works better for the Twitter sentiment analysis task in our experiments. Knowledge transfer from multiple tasks was considered to estimate the age of Twitter users based on the content of their tweets in (Riemer et al., 2015). We experimented with the hidden layer sharing approach outlined in that work and found that even when using just a single softmax combining layer, it would overfit on our limited training and validation data. Progressive neural networks (Rusu et al., 2016) is a recently proposed method very similar in motivation to our forgetting cost as it is directly trying to solve the catastrophic forgetting problem. The idea is that learned weight matrices relate the fixed representations learned on the source task to the construction of representations for the target task. In our experiments, the progressive neural networks approach consistently fails to even match the results achieved with fine-tuning. We hypothesize that although using fixed representations to aid learning addresses catastrophic forgetting, it suffers from the curse of dimensionality. As such, when training data is relatively small given the complexity of the task, it is prone to overfitting as it effectively increases the input dimension size through shared fixed representations.

The combination of logic rules and neural networks has been explored in a variety of different architectures and settings. These neural-symbolic systems (Garcez et al., 2012) include early examples such as KBANN (Towell et al., 1990) that construct network architectures from given rules to perform reasoning. (Hu et al., 2016) very recently also looked at the problem of distilling logical rules into a neural network text analytics classifier. However, our approach is much more generic as it can be applied to integrate knowledge from any kind of pre-made classifier and treats the rule engine as a black box. In (Hu et al., 2016) they consider the individual rules and leverage an iterative convex optimization algorithm alongside the neural network to regularize the subspace of the network. In our work we demonstrate that, by guarding against catastrophic forgetting, it is possible to efficiently leverage rules for transfer by utilizing a generic sequential knowledge transfer framework. We do

not need to make any modification to the architecture of the neural network during testing and do not need iterative convex optimization during training.

# 3 FORGETTING COST REGULARIZATION

## 3.1 SEQUENTIAL KNOWLEDGE TRANSFER PROBLEM STATEMENT

In the sequential knowledge transfer problem setting explored in this paper, training is first conducted solely on the source task examples $S$, including $K_S$ training examples $(x_{Si}, y_{Si}) \in S$ where $x_{Si}$ is the input representation and $y_{Si}$ is the output representation. After training is complete on $S$, we would like to now use prior knowledge obtained in the model trained on $S$ to improve generalization on a new target task with examples $T$, which includes $K_T$ training examples $(x_{Ti}, y_{Ti}) \in T$. Here we assume that the input representations $x_{Si}$ and $x_{Ti}$ are semantically aligned in the same representation space. As such, if there is useful knowledge in $S$ that applies in some direct or indirect way to the target task that is not present in $T$, we would expect a good knowledge integration approach to generalize better to the target task than it is possible to using the training data in $T$ alone. Strong performance for the sequential knowledge transfer problem is a first step towards the greater goal of a mechanism for effective lifelong learning (Thrun, 1996).

## 3.2 FORGETTING COST FOR TUNING A TARGET TASK MODEL

The most straightforward application of our proposed forgetting cost paradigm is for the case of integrating a neural network that has been trained on source task data $S$, which has outputs in the same representation space as the outputs for the target task data $T$. In this case, the forgetting cost amounts to the addition of a regularization term in the objective function during the integration phase when we train using $T$. This promotes the neural network to be able to recreate the soft labels of the initialized model found after training on $S$ before integration is started with $T$. More formally:

$$Loss = L(y, \hat{y}) + \alpha_f L(y_{init}, \hat{y}) \tag{1}$$

where $L$ is some loss function (we use mean squared error in our experiments) and $y_{init}$ is the soft label generated for the target task input $x_{Ti}$ based on the model after training just on $S$. The model trained just on $S$ is also used to initialize the weights of the target task model before integration with $T$ as we do in the standard fine-tuning paradigm. $\alpha_f$ is a hyperparameter that can be utilized to control the extent of allowed forgetting. Of course, a very similar way to express this idea would be to mix synthetic training examples $T'$ with the same input as $T$ and output generated by the model trained just on $S$ with the true target task training examples $T$. In this case, the mixing rate of the teacher generated training examples is analogous to our forgetting parameter $\alpha_f$ determining the prioritization. These techniques perform quite similarly in our experiments, but we actually find that the formulation in equations 1 and 3 perform slightly better on the test set. For example, this formulation is superior by 0.4% accuracy in tuning a distilled representation of a logical rule engine. We conjecture that learning tasks in the same gradient step when they are related to the same input data results in slightly less noisy gradients.

## 3.3 FORGETTING COST FOR KNOWLEDGE TRANSFER FROM A RELATED TASK

The assumption in section 3.2 that the output of the source task data $S$ should be in the same representation space as the output for the target task data $T$ is quite a big one. It rules out the vast majority of knowledge sources that we can potentially leverage. As such, we propose an extension that does not make this restriction for application in sequential knowledge transfer of tasks that are not directly semantically aligned. We update our model to include another predicted output separate from $\hat{y}$:

$$\hat{y}_{init} = f_{init}(W_{fixed} h_{shared} + b_{fixed}) \tag{2}$$

where $\hat{y}_{init}$ is a predicted output attempting to recreate the soft labels of the original model trained just on $S$. $f_{init}$ is the non-linearity used in the final layer of the source task model. Weight matrix $W_{fixed}$ and bias $b_{fixed}$ are taken from the final layer of the source task model and are not updated

during integration with the target task data $T$. As a result, the loss function is updated from section 3.2:

$$Loss = L(y, \hat{y}) + \alpha_f L(y_{init}, \hat{y}_{init}) \tag{3}$$

where the hidden state is shared between both terms in the objective function. Up to the shared hidden layer, we initialize the model for the target task with the weights learned just using $S$. Random matrices and bias vectors are now used to initialize the prediction of $\hat{y}$ based on the shared hidden representation. This can be seen as a weak form of restricting the model parameters that can be useful for regularization. The hidden representation is in effect constrained so that it is promoted not to change in key areas that have a large effect on the output vector of the source task model. On the other hand, there is little regularization for parameters that have little effect on the output vector for the source task model.

## 4 Recurrent Neural Network Model

In recent years, recurrent neural network models have become a tool of choice for many NLP tasks. In particular, the LSTM variant (Hochreiter & Schmidhuber, 1997) has become popular as it alleviates the vanishing gradients problem (Bengio et al., 1994) known to stop recurrent neural networks from learning long term dependencies over the input sequence. In our experiments we use the simpler GRU network (Cho et al., 2014), (Chung et al., 2014) that generally achieves the same accuracy despite a less complex architecture. Each time step $t$ is associated with an input $x_t$ and a hidden state $h_t$. The mechanics of the GRU are defined with the following equations:

$$z_t = \sigma(W_{xz} x_t + W_{hz} h_{t-1}) \tag{4}$$

$$r_t = \sigma(W_{xr} x_t + W_{hr} h_{t-1}) \tag{5}$$

$$\tilde{h}_t = tanh(W_{xh} x_t + r_t \circ W_{hh} h_{t-1}) \tag{6}$$

$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t \tag{7}$$

where $\circ$ denotes an element-wise product. $W_{xz}$, $W_{xr}$, and $W_{xh}$ represent learned matrices that project from the input size to the hidden size. $W_{hz}$, $W_{hr}$, and $W_{hh}$ represent learned matrices that project from the hidden size to the hidden size. In our work we evaluate the GRU in the categorical prediction setting. For each document, the hidden state after the last word $h_L$ is used for the prediction $\hat{y}$ of the label $y$. As such, we treat $h_L$ as the shared hidden representation $h_{shared}$ from section 3.3 for our experiments.

$$\hat{y} = f(W_{yh} h_L + b_y) \tag{8}$$

The prediction goes through one other non-linear function $f$ after the final hidden state is derived. In our experiments we use the softmax function, but others are useful in different settings. A model that builds on top of GRUs with an external memory storage paradigm (Kumar et al., 2015) currently holds the state of the art on movie review sentiment analysis. However, we focus just on the straightforward single layer GRU model in our experiments so that we can more easily disentangle factors of influence on performance. Our GRU model was fed a sequence of fixed 300 dimensional Glove vectors (Pennington et al., 2014), representing words based on analysis of 840 billion words from a common crawl of the internet, as the input $x_t$ for all tasks. It has been shown in a number of papers that tuning the word embeddings during training could increase performance, and it is possible our approach could have performed better had we done so.

## 5 Sequential Knowledge Transfer Experiments

### 5.1 Experiment Details

Our neural network models were implemented in Theano (Theano Development Team, 2016) and trained with Stochastic Gradient Descent. As we did not use an advanced optimization method and

noticed run to run variation in performance, for all of our transfer learning models we trained 10 parallel versions and chose the one with the highest validation accuracy. The SemEval 2016 Task 4 Subtask A training set consists of 10,000 total training examples, but we were only able to receive 8,906 because of tweet removals when we used the downloading script. For the target task data across our experiments, 7,600 examples of the SemEval training set examples were used for training and the rest for validation. The GRU model achieves only 53.6% accuracy on the SemEval testing data when just training with the target task data and random initialization. In order to improve, we consider knowledge transfer from GRUs trained for the following source tasks to the SemEval target task data:

**Distilling Logical Rules:** Knowledge distillation can be performed using teacher models that are very different in structure than their neural network based student models. We demonstrate with this task that a compilation of logical linguistic rules can be used as an effective teacher for a GRU by having the GRU attempt to create the output of the rule engine generated over unlabeled in domain data. Specifically, our gazetteer based logical rule engine separates sentences and phrases in the text. It then applies dictionaries of positive and negative sentiment words and phrases to the corresponding text. For each positive or negative phrase found, it checks to see if negation or double negation are applied, and modifies the polarity of the sentiment accordingly. The result for any piece of text is a count of positive and negative sentiment occurrences. For this task, we simply count the total number of positive and negative indicators to give an overall positive, negative or neutral score. We provide addition details on how we mapped rules to soft targets for the student network to recreate in Appendix A. We utilized a GRU model with 50 hidden units and 50,000 unlabeled examples for our source task model. We distill off the soft labels as in (Hinton et al., 2015), but set our temperature fixed at 1.0. It is possible that our performance could have improved by tuning this parameter. Additional details about the selection of the network and data size are included in Appendix B. The logical rule model itself achieves 57.8% accuracy on the SemEval testing data and the rules distilled into a GRU as explained in section 4 achieves 58.9% accuracy before any integration with the SemEval target task data. We leverage this task for comparison of knowledge transfer techniques when the source task and target task share an output space as discussed in section 3.2.

**Binary Movie Reviews:** For knowledge transfer from related tasks as discussed in section 3.3 we first consider the Stanford Sentiment Treebank (Socher et al., 2013), which is a popular sentiment dataset based on the movie review domain. We consider one source task to be the binary (positive, and negative) sentence level sentiment subtask which contains 6,920 training examples, 872 validation examples, and 1,821 testing examples. Our GRU model with 40 hidden units achieves 85.5% accuracy on this task.

**Five Class Movie Reviews:** We also consider another source task leveraging the Stanford Sentiment Treebank data from the fine grained (very positive, positive, neutral, negative, and very negative) sentence level sentiment substask which contains 8,544 training examples, 1,101 validation examples, and 2,210 testing examples. We use a GRU model with 200 hidden units to accommodate for the increased task complexity and achieve 45.9% accuracy. This fine grained model can actually be assessed directly on the SemEval task by projecting from five classes to three classes, but it only achieves 44.2% accuracy with no tuning on the target task data. Our performance on these two movie review source tasks is quite similar to what was reported in (Tai et al., 2015) when using a similar setup, but with LSTMs for both subtasks.

**Emoticon Heuristic:** Finally, we consider a semi-supervised task based on emoticon prediction motivated by the successful work in (Go et al., 2009), leveraging it in the twitter sentiment domain and its use as a vital component of the SemEval competition winning system (Bethard et al., 2016). We find unlabelled tweets that contain smileys, frowns, or laughing emoticons. We remove emoticons from the tweet before prediction and compile a dataset of 250,000 training examples, 50,000 validation examples, and 100,000 testing examples for each of the three classes. This is multiple orders of magnitude smaller than the 90 million tweets used in (Bethard et al., 2016) to allow for quick experimentation. Our GRU model with 50 hidden units achieves 63.4% accuracy on the emoticon prediction test set.

### 5.2 SEQUENTIAL KNOWLEDGE TRANSFER ALGORITHMS

We consider multiple sequential knowledge transfer algorithms for experimental comparison. Each uses only the source task data for learning the source task and only the target task data for integrating with the target task. This way integration is fast and simple, because it does not incorporate storage and replay of examples from the potentially very large source task as argued in (Li & Hoiem, 2016).

**Fine-Tuning:** The representation is simply initialized with the representation found after training on the source task and then trained as usual on the target task. This approach was pioneered in (Hinton & Salakhutdinov, 2006), in application to unsupervised source tasks and applied to transfer learning in (Bengio et al., 2012), and (Mesnil et al.). The learning rate is tuned by a grid search based on the validation set performance.

**Progressive Networks:** We also compare with our implementation of a progressive neural network (Rusu et al., 2016), where the representation learned for the source task is held fixed and integrated with a target task specific model via lateral connections trained using the target task data. The learning rate is also tuned based on a grid search using the validation set.

**Learning without Forgetting (LwF):** In the LwF paradigm, joint training is performed after parameter initialization. This is achieved by treating the target task data and the output generated by the source task model based on the target task input data as two jointly learned tasks as in (Caruana, 1997). As opposed to our proposed forgetting cost, the source task specific parameters are not held fixed while training on the target task data. The learning rate and mixing rate between the tasks are tuned by a grid search based on validation set performance. We first consider a version of the LwF model that leverages a random initialization of the target task specific parameters and initialization of all parameters learned on the source task with the learned values. We also consider another formulation that we call Greedy LwF. This is actually more closely aligned with the original paper (Li & Hoiem, 2016). All source task parameters are first held fixed, and the target task specific parameters are learned alone before joint training with all of the parameters unfrozen as a second step. For the case of source tasks with output in the space of the target task output, there are no source task specific parameters, so the forgetting cost can be viewed as a viable interpretation of the LwF paradigm appropriate in that setting.

**Forgetting Cost:** Finally, we compare each baseline model with our proposed forgetting cost described in section 3. The learning rate as well as $\alpha_f$ from equations 1 and 3 were tuned by a grid search based on the validation set performance.

### 5.3 TARGET TASK RESULTS

We empirically evaluate the generalization performance of the forgetting cost for sequential knowledge transfer from four different source tasks in Table 1 and Table 2. The source task considered in Table 1 is distilling a logical rule model, leveraging the technique outlined in equation 1. In Table 2 we leverage the forgetting cost for related task knowledge transfer as outlined in equation 3.

Our experimental results on the SemEval data validate our intuition that the forgetting cost should lead to stronger regularization and better generalization performance. One thing to note about our progressive neural networks implementation is that it effectively has only one hidden layer, because we hold our embeddings fixed during model training and the same embeddings are shared among the models used for all of the tasks. It is possible that having multiple layers of lateral connections is important to achieving good performance. However, this setting was not applicable in our experiments. Our results for sequential knowledge transfer on the SemEval benchmark are quite encouraging as the forgetting cost outperforms baselines significantly in all cases.

We additionally have validated the intuition that equation 1 should perform stronger regularization than equation 3 when equation 1 is applicable. In fact, for our distilled logical rule model tuning experiments, we found that equation 1 performs 3% better on the test set. In an attempt to understand more about what caused this performance difference, we monitored testing set performance at each epoch and noticed that equation 3 is actually prone to overfitting away from a good solution on the test set. However, it often finds a pretty good one comparable to equation 1 early in training. When equation 1 could be applied, it seems to be a useful regularization to constrain both the hidden layer and the output layer to align with the model learned on the source task. In equation 3, the

| Model Description | Accuracy on SemEval Test Set |
|---|---|
| Forgetting Cost Transfer | **64.4%** |
| Fine-tuning Transfer | 58.5% |
| Progressive Networks Transfer | 56.9% |
| Distilled Logical Rule Model | 58.9% |
| Logical Rule Model | 57.8% |
| GRU Trained on Only SemEval Data | 53.6% |

Table 1: Evaluation of target task tuning methodologies for a distilled rule model to the task of SemEval 2016 Task 4 Subtask A.

| Source Task | Fine-Tuning | Progressive Networks | LwF | Greedy LwF | Forgetting Cost |
|---|---|---|---|---|---|
| Binary Movie Reviews | 57.3% | 54.5% | 58.1% | 58.8% | **59.7%** |
| Five Class Movie Reviews | 57.4% | 54.6% | 57.1% | 56.6% | **58.2%** |
| Emoticon Heuristic | 55.8% | 53.2% | 57.7% | 56.7% | **58.6%** |

Table 2: Evaluation of knowledge transfer from three source tasks to the task of SemEval 2016 Task 4 Subtask A.

hidden to output transformation learned for the target task can in contrast learn to deviate from the transformation learned for the source task.

### 5.4 Source Task Performance After Target Task Integration

In Table 3 we explore the retention of empirical performance on the source task for knowledge transfer algorithms after integration with the target task is complete. Apparently in these cases, allowing relearning of the source task model during integration with the target task data is indeed destructive to source task performance. LwF outperforms Fine-Tuning significantly in knowledge retention for movie reviews, but interestingly does not for the emoticon heuristic. The effect of the greedy target task initialization strategy also appears inconsistent. It seems it is possible that this greedy initialization could improve our proposed forgetting cost paradigm in some cases as well. However, a rigorous analysis of the tradeoffs for this initialization approach is beyond the scope of this paper.

As the source task representation is literally stored fixed as part of the target task representation in progressive neural networks, it is not clear how to assess any effective forgetting of the source task during target task integration. As a result, we omit them from our source task forgetting experiments.

### 5.5 Inspection of Learned Representations

Now that we have established the empirical benefits of our proposed forgetting cost, we will demonstrate what it achieves qualitatively through examples. In Table 4 we include a sample of examples that are predicted correctly by transferring the knowledge source with the forgetting cost paradigm and not with fine-tuning based integration. The effect is, perhaps, easiest to understand for the rule based and movie review based transfer scenarios. For the rule based transfer setting you can literally map insights that are not forgotten to their respective logical rule in the model, as is the case in these examples. Moreover, we can see movie domain specific terminology such as "May the force be with" is seemingly forgotten with standard fine-tuning, but not when the forgetting cost regularization is applied.

| Source Task | Fine-Tuning | LwF | Greedy LwF | Forgetting Cost | Source Only |
|---|---|---|---|---|---|
| Binary Movie Reviews | 80.7% | 81.3% | 81.5% | **83.3%** | 85.5% |
| Five Class Movie Reviews | 41.6% | 42.8% | 43.1% | **43.3%** | 45.9% |
| Emoticon Heuristic | 59.4% | 59.1% | 58.9% | **60.3%** | 63.4% |

Table 3: Evaluation of accuracy on the source task after integration with the target task data of SemEval 2016 Task 4 Subtask A. The accuracy after only source task training prior to integration with the target task is included for reference as a baseline.

| Source | Tweet | Label | Fine-Tuning | Forgetting Cost |
|---|---|---|---|---|
| Logical Rules | John Kasich should feel proud of his performance at the #GOPDebate Thursday night. He looked more presidential than the rest of the field. | Positive | Neutral | Positive |
| Logical Rules | @BrunoMars I'm so tired of you dressing like you ain't got no money. You went from wearing Gucci loafers to 6th grade boy Sketchers. | Negative | Neutral | Negative |
| Logical Rules | @DavidVonderhaar loving the beta Vahn, even playing it on PC with a PS4 controller without aim assist, can't wait for November 6 | Positive | Neutral | Positive |
| Movie Reviews | Selena Gomez presented Amy Schumer with an award and a heap of praise at the Hollywood Film Awards on November 1. | Positive | Negative | Positive |
| Movie Reviews | mailjet: It's Fri...we mean Star Wars Day. May the force be with all of your emails! https://t.co/FbDdjiJVUT | Positive | Neutral | Positive |
| Movie Reviews | Straight Outta Compton's success hopefully convinces New Line Cinema to give Ice Cube the right budget for the last Friday movie. | Positive | Neutral | Positive |
| Emoticons | That ball Kris Bryant just hit is the 2nd farthest ball I've ever seen hit. He is officially ridiculous. | Positive | Neutral | Positive |
| Emoticons | This fandom's a mess omg, I wouldn't be surprise if tomorrow there's a trend who says Niall's going to marry his cousin #WeKnowTheTruth | Negative | Positive | Negative |
| Emoticons | Christians snapchat story makes me want to kill myself..like I feel like a depressed 8th grader going through that emo phase | Negative | Neutral | Negative |

Table 4: Some transfer learning examples from each knowledge source to SemEval 2016 where the GRU model successfully predicts sentiment when using the forgetting cost paradigm, but not with fine-tuning based integration.

Considering that we have shown a neural network can distill and improve a representation learned by a logical rule engine, how the final representation differs from the logic of the original engine is of practical interest. We thus compare the agreement of our fine-tuned rule based GRU with the original rule model on the SemEval testing set. We find that the transferred model achieves 78.7% agreement with the rule model when the rule model is right. This clearly indicates that our final model is not deterministic based on the rule engine, and has a probability of adding errors even when the original rule model works well. However, our model actually has 44.7% accuracy on the examples the rule model got wrong. Our approach yields significant gains in comparison to the original rule classifiers, improving from 57.8% to 64.4% test set accuracy before even incorporating in auxiliary knowledge sources.

# 6 Integrating Transfer Learning from Multiple Tasks with Ensemble Distillation

## 6.1 Ensemble Methodology

In our experiments we tried to find a balance between an ensemble model that is powerful enough to have an adaptive weighted average decision function and not so powerful that it overfits on our limited training and validation data. Our model is quite similar in architecture to the gating network component of a hierarchical mixture of experts model (Jacobs et al., 1991), (Jordan & Jacobs, 1994). We tried our model over all four representations at once and found that it overfits. Our experiments showed it is more effective to adopt a greedy ensembling strategy where all models are combined with the best performing model on the validation set at each phase until only two models are left. Finally, these two models are combined with the same mechanism. (Riemer et al., 2016) suggests that a many element gating network can be improved with a sparsity constraint, but this did not work as well as the greedy strategy for our model and experiments.

More formally, for any two models $A$ and $B$ combined in an ensemble, we train the following mechanism using Stochastic Gradient Descent:

| Model Description | Accuracy on SemEval Test Set |
|---|---|
| Distilled GRU Trained on Full Ensemble | **66.0%** |
| Full Ensemble | 65.9% |
| Ensemble with Logical Rules and Both Movie Review Tasks | 65.7% |
| Ensemble with Logical Rules and Binary Movie Reviews | 65.4% |
| Ensemble with Logical Rules and Five Class Movie Reviews | 65.1% |
| Ensemble with Logical Rules and Emoticon Prediction | 65.0% |
| Ensemble with Both Movie Review Tasks | 62.1% |
| GRU Trained on Only SemEval Data | 53.6% |
| SwissCheese (Bethard et al., 2016) | 64.6% |
| NTNUSentEval (Jahren et al., 2016) | 64.3% |
| UniPI (Attardi & Sartiano, 2016) | 63.9% |
| CUFE (Nabil et al., 2016) | 63.7% |
| INSIGHT-1 (Ruder et al., 2016) | 63.5% |

Table 5: Empirical three way sentiment classification results on the SemEval 2016 Task 4 Subtask A test set.

$$m_A = \sigma(W_A \hat{y}_A + b_A) \tag{9}$$

$$m_B = \sigma(W_B \hat{y}_B + b_B) \tag{10}$$

$$a_A = \frac{m_A}{m_A + m_B} \tag{11}$$

$$a_B = \frac{m_B}{m_A + m_B} \tag{12}$$

$$\hat{y}_{ensemble} = a_A \hat{y}_A + a_B \hat{y}_B \tag{13}$$

where $\hat{y}_{ensemble}$ is the prediction vector of the combined ensemble. $\hat{y}_A$ and $\hat{y}_B$ are the output vectors of the individual models.

### 6.2 Ensemble Results

Our ensemble model was trained on what was set aside as the validation data during the initial training with early stopping. In the first phase of combining, the model transferred from the logical rule source task was combined with each model. In the second phase, the model based on transfer from the binary movie review sentiment model was combined with each model. In the third phase, the two remaining models were combined. The results of our ensemble in Table 5 suggest that it is possible to further improve the performance of a single sequential transfer model by intelligently combining its predictions with models that have other perspectives. This is because they are modeled using different source tasks for prior knowledge. Impressively, our final distilled model surpasses results from all prior models on the SemEval 2016 benchmark using the same final architecture of a 50 hidden unit GRU model that is clearly not even competitive when trained simply on the task specific labeled data. The prior best model SwissCheese (Bethard et al., 2016) consists of random forests ensemble built utilizing multiple convolutional neural network models and distant supervision. In fact, we achieve superior results despite using over an order of magnitude less total data for training our model.

We would also like to underscore that our total improvement of 1.5% as a result of creating an ensemble with our best transferred model from the logical rule source task can be viewed as quite disappointing, despite achieving state of the art results. In fact, in the theoretical limit of having a decision model that switches to the best already learned model at each point, our four transferred representations would achieve 85.1% accuracy together. For the combination of the movie review based models and logical rule based model we can get to 81.4% accuracy. Moreover, we can get 76.5% accuracy with just the logical rule based transfer model and the emoticon prediction based transfer model. Unfortunately, we achieve nowhere near these theoretical results despite representations that are apparently quite diverse. This seems indicative that there are significant gains yet to be uncovered in integrating these representations.

## 7 CONCLUSION

We consider a new methodology called the forgetting cost for preventing the catastrophic forgetting problem of neural network sequential transfer learning. The forgetting cost is practical and easy to implement. We have demonstrated for the challenging task of Twitter sentiment analysis that it can uncover significant gains in generalization performance and that it seems to not forget knowledge traditionally forgotten from the source task during fine-tuning. Our strong empirical results still motivate multiple avenues with high potential for continued exploration in text analytics. Using logical rules to improve neural network models is a promising direction for humans to efficiently contribute to increased model performance. Additionally, the large diversity of representations learned from multiple classifiers with the same target task but different source tasks seems to indicate there is potential to see even much greater gains when integrating multiple sources of knowledge transfer.

## REFERENCES

Giuseppe Attardi and Daniele Sartiano. Unipi at semeval-2016 task 4: Convolutional neural networks for sen-timent classification. *Proceedings of SemEval*, pp. 220–224, 2016.

Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pp. 2654–2662, 2014.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166, 1994.

Yoshua Bengio et al. Deep learning of representations for unsupervised and transfer learning. *ICML Unsupervised and Transfer Learning*, 27:17–36, 2012.

Steven Bethard, Daniel M. Cer, Marine Carpuat, David Jurgens, Preslav Nakov, and Torsten Zesch (eds.). *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*, 2016. The Association for Computer Linguistics. ISBN 978-1-941643-95-2. URL http://aclweb.org/anthology/S/S16/.

Cristian Bucilu, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 535–541. ACM, 2006.

Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997. doi: 10.1023/A:1007379606734. URL http://dx.doi.org/10.1023/A:1007379606734.

Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*, 2015.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Artur S d'Avila Garcez, Krysia Broda, and Dov M Gabbay. Neural-symbolic learning systems: foundations and applications, 2012.

Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. 2009.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. Harnessing deep neural networks with logic rules. *arXiv preprint arXiv:1603.06318*, 2016.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

Brage Ekroll Jahren, Valerij Fredriksen, Björn Gambäck, and Lars Bungum. Ntnusenteval at semeval-2016 task 4: Combining general classifiers for fast twitter sentiment analysis. *Proceedings of SemEval*, pp. 103–108, 2016.

Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. *arXiv preprint arXiv:1506.07285*, 2015.

Zhizhong Li and Derek Hoiem. Learning without forgetting. In *European Conference on Computer Vision*, pp. 614–629. Springer, 2016.

David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. Unifying distillation and privileged information. *stat*, 1050:26, 2016.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pp. 142–150. Association for Computational Linguistics, 2011.

Grégoire Mesnil, Yann Dauphin, Xavier Glorot, Salah Rifai, Yoshua Bengio, Ian J Goodfellow, Erick Lavoie, Xavier Muller, Guillaume Desjardins, David Warde-Farley, et al. Unsupervised and transfer learning challenge: a deep learning approach.

Grégoire Mesnil, Tomas Mikolov, Marc'Aurelio Ranzato, and Yoshua Bengio. Ensemble of generative and discriminative techniques for sentiment analysis of movie reviews. *arXiv preprint arXiv:1412.5335*, 2014.

Jacob MJ Murre. Learning and categorization in modular neural networks. 1992.

Mahmoud Nabil, Mohamed Aly, and Amir F Atiya. Cufe at semeval-2016 task 4: A gated recurrent model for sentiment classification. *Proceedings of SemEval*, pp. 52–57, 2016.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Veselin Stoyanovand, and Fabrizio Sebastiani. Semeval-2016 task 4: Sentiment analysis in twitter. *In Proc. of the 10th International Workshop on Semantic Evaluation (SemEval)*, 2016.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pp. 1532–1543, 2014.

Matthew Riemer, Sophia Krasikov, and Harini Srinivasan. A deep learning and knowledge transfer based architecture for social media user characteristic determination. *SocialNLP 2015 at NAACL*, pp. 39, 2015.

Matthew Riemer, Aditya Vempaty, Flavio Calmon, Fenno Heath, Richard Hull, and Elham Khabiri. Correcting forecasts with multifactor neural attention. In *Proceedings of The 33rd International Conference on Machine Learning*, 2016.

Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2): 123–146, 1995.

Anthony Robins. Consolidation in neural networks and in the sleeping brain. *Connection Science*, 8(2):259–276, 1996.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

Sebastian Ruder, Parsa Ghaffari, and John G Breslin. Insight-1 at semeval-2016 task 5: Deep learning for multilingual aspect-based sentiment analysis. *arXiv preprint arXiv:1609.02748*, 2016.

Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, pp. 1642. Citeseer, 2013.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.

Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016. URL `http://arxiv.org/abs/1605.02688`.

Sebastian Thrun. Is learning the n-th thing any easier than learning the first? *Advances in neural information processing systems*, pp. 640–646, 1996.

Geoffrey G Towell, Jude W Shavlik, and Michiel O Noordewier. Refinement of approximate domain theories by knowledge-based neural networks. In *In Proceedings of the Eighth National Conference on Artificial Intelligence*. Citeseer, 1990.

## A Mapping Sentiment Rules to Soft Targets

The gazetteer based logical rule engine separates sentences and phrases in the text. It then applies dictionaries of positive and negative sentiment words and phrases to the corresponding text. For each positive or negative phrase found, it checks to see if negation or double negation are applied, and modifies the polarity of the sentiment accordingly. The result for any piece of text is a count of positive and negative sentiment occurrences. For this task, we simply count the total number of positive and negative indicators to give an overall positive, negative or neutral score. To be concrete, we have a simple procedure for mapping positive and negative word counts to soft labels that could be used for distillation. If there are no positive or negative words, the output vector is a one hot vector corresponding to a neutral label. If there are an unequal number of positive and negative sentiment words, the neutral label is zero and the raw counts are sent to the softmax function to create a soft label over the positive and negative word occurrences. Finally, if there are an equal amount of positive and negative words, we consider the added total sentiment words plus one in the neutral label as well as the number of positive words and negative words before sending these totals through a softmax function.

## B Size Selection for the Rule Distillation Task

In Table 6 we detail the performance of distilling a logical rule engine into a GRU based recurrent neural network by imposing soft labels over unlabeled tweets. The fact that we keep our word representations fixed with general purpose unsupervised data makes it difficult for the GRU to distill the entire model without a large number of examples. Additionally, as there were a large number of examples in our distillation experiments, we did not experience high run to run variation and only trained a single GRU model for each distillation experiment (as opposed to picking the best validation error of 10 parallel training routines as in our transfer experiments). Our distilled GRU is

| Hidden Units | Examples | Alignment with Teacher | Accuracy on SemEval Test Set |
|---|---|---|---|
| 25 | 50,000 | 88.3% | 59.1% |
| 25 | 300,000 | 91.9% | 58.6% |
| 50 | 50,000 | 88.6% | 58.9% |
| 50 | 300,000 | 93.0% | 58.5% |
| 75 | 50,000 | 88.7% | 58.9% |
| 75 | 300,000 | 93.6% | 58.3% |
| 100 | 50,000 | 88.6% | 58.7% |
| 100 | 300,000 | 93.8% | 58.1% |
| 125 | 50,000 | 88.5% | 58.7% |
| 125 | 300,000 | 93.7% | 58.3% |
| 150 | 50,000 | 88.5% | 59.0% |
| 150 | 300,000 | 94.0% | 58.5% |

Table 6: Logical rule engine distillation performance and SemEval 2016 Task 4 Subtask A accuracy as a function of the number of hidden units in the GRU and the number of training examples. The 50 hidden unit and 50,000 training example model performs the best on the SemEval training set.

better on the testing set than the original classifier, likely because this input representation prevents the model from overfitting to the idiosyncrasies of the rule engine. This actually underscores an important point for the distillation of abstract knowledge. If the target task is known during distillation, it may be beneficial to stop short of totally distilling the original knowledge as it may hurt down stream performance past a certain point. We impose a simple policy where the best hidden unit and training example combination is selected based on performance on the training data of the target task. As a result, we use the model with 50 hidden units based on 50,000 training examples in our experiments integrating with other knowledge. This model is a pretty good one to choose, and achieves high transfer performance relative to models that overfit on the teacher network.