The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Actionable Email Intent Modeling with Reparametrized RNNs*†

Chu-Cheng Lin‡
Johns Hopkins University
Baltimore, MD
clin103@jhu.edu

Dongyeop Kang
Carnegie Mellon University
Pittsburgh, PA
dongyeok@cs.cmu.edu

Michael Gamon
Microsoft Research
Redmond, WA
mgamon@microsoft.com

Patrick Pantel
Microsoft Research
Redmond, WA
ppantel@microsoft.com

Abstract

Emails in the workplace are often intentional calls to action for its recipients. We propose to annotate these emails for what action its recipient will take. We argue that our approach of action-based annotation is more scalable and theory-agnostic than traditional speech-act-based email intent annotation, while still carrying important semantic and pragmatic information. We show that our action-based annotation scheme achieves good inter-annotator agreement. We also show that we can leverage threaded messages from other domains, which exhibit comparable intents in their conversation, with domain adaptive RAINBOW(Recurrently AttentIve Neural Bag-Of-Words). On a collection of datasets consisting of IRC, Reddit, and email, our reparametrized RNNs outperform common multitask/multidomain approaches on several speech act related tasks. We also experiment with a minimally supervised scenario of email recipient action classification, and find the reparametrized RNNs learn a useful representation.

1 Introduction

Despite the emergence of many new communication tools in the workplace, email remains a major, if not the dominant, messaging platform in many corporate settings (Agema 2015). Helping people manage and act on their emails can make them more productive. Recently, Google's system that suggests email replies has gained wide adoption (Kannan et al. 2016). We can imagine many other classes of assistance scenarios that can improve worker productivity. For example, consider a system that is capable of predicting your next action when receiving an email. The system could then offer assistance to accomplish that action, for example in the form of a quick reply, adding a task to your to-do list, or helping you take action against another system. To build and train such systems, email data sets are essential, but unfortunately public email datasets such as Klimt and Yang; Oard et al. (2004; 2015) are much smaller than the proprietary data used by Google; and more importantly, they lack any direct information/annotation regarding the recipients' actions.

In this paper, we design an annotation scheme for such actions and have applied it to a corpus of publicly available emails. In order to overcome the data bottleneck for end-to-end training, we leverage other data and annotations that we hypothesize to contain structures similar to email and recipient actions. We apply multitask and multidomain learning, which use domain or task invariant knowledge to improve performance on a specific task/domain (Caruana 1997; Yang and Hospedales 2014). We show that these secondary domains and tasks in combination with multitask and multidomain learning can help our model discover invariant structures in conversations that improve a classifier on our primary data and task: email recipient action classification.

Previous work in the deep learning literature tackled multidomain/multitask learning by designing an encoder that encodes all data and the domain/task description into a *shared* representation space (Collobert and Weston 2008; Glorot, Bordes, and Bengio 2011; Ammar et al. 2016; Yang, Salakhutdinov, and Cohen 2017). The overall model architecture generally is unchanged from the single-domain single-task setting; but the learned representations are now reparametrized to take account of knowledge from additional data and task/domain knowledge. In this work, we propose an alternative approach of *model reparametrization*. We train multiple parameter-sharing models across different domains and tasks jointly, without maintaining a shared encoded representation in the network. We show that reparametrized LSTMs consistently achieve better likelihood and overall accuracy on test data than common domain adaption variants. We also show that the representation extracted from a network instantiated with the shared parameter weights performs well on a previously unseen task.

The contributions of this paper are:

First, we designed an annotation scheme for labeling actionable workplace emails, which as we argue in section 2.2,

*Madian Khabsa (madian@apple.com) and Ahmed Hassan Awadallah (hassanam@microsoft.com) are co-authors of this work. An oversight during submission omitted their names and that error could not be amended due to the AAAI policy regarding author list modification. For academic integrity reasons, please cite the corrected version here: https://rebrand.ly/rainbowmsr.

†This work was done when Madian Khabsa was at Microsoft Research.

‡This work was done during an internship at Microsoft Research.

is more amenable to an end-to-end training paradigm, and collected an annotated dataset. Second, we propose a family of reparametrized RNNs for both multitask and multidomain learning. Finally, we show that such models encode domain-invariant features and, in the absence of sufficient data for end-to-end learning, still provide useful features for scoping tasks in an unsupervised learning setting.

2 Data

2.1 The Avocado Dataset

In this study, all email messages we annotate and evaluate on are part of the Avocado dataset (Oard et al. 2015), which consists of emails and attachments taken from 279 accounts of a defunct information technology company referred to as "Avocado".[1] Email threads are reconstructed from the recipients' mailboxes. For the purpose of this paper, we only use complete (thread contains all replies) and linear (every follow-up is a reply to the previous email) threads.[2]

2.2 Recipient Actions

Workplace email is known to be highly task-oriented (Khoussainov and Kushmerick 2005; Corston-Oliver et al. 2004). As opposed to chit chat on the Internet, speaker intents and expected actions on the email are in general very precise. We aim to annotate the actions, which makes our approach differ in a subtle but important way from previous work such as (Cohen, Carvalho, and Mitchell 2004), which is mostly focused on annotating emails for *sender intents*, modeled after illocutionary acts in Speech Act theory (Searle 1976). We believe that annotating recipient actions has the following advantages over annotating sender intents: First, action based annotation is not tied to a particular speech act taxonomy. The design of such a taxonomy is highly dependent on the system's use cases (Traum 1999) and definitions of sender intent can be circular (Riezler 2014). Even within a single domain such as email, there have been several different sender intent taxonomies (Goldstein and Sabin 2006). A speech-act-agnostic scheme that focuses on the recipient's action generalizes better across scenarios. Our annotation scheme also has a lower risk of injected bias because the annotation relies on expected (or even observed) actions performed in response to an email, as opposed to relying on the annotator's intuition about the sender's intent. Lastly, while in this paper we rely on annotators for these action annotations, many of our annotated actions translate into very specific actions on the computer. Therefore we anticipate intelligent user interfaces could be used to capture and remind users of such email actions, as in Dredze et al. (2008).

Based on our findings in two pilot runs of email annotations among the authors, we propose the set of recipient actions listed in table 1, which fall in three broad categories:

Message sending We identify that in many cases, the recipient is most likely to send out another email, either as a reply to the sender or to someone else. As listed in table 1, Reply-Yesno, Reply-Ack, Reply-Other, Investigate, Send-New-Email are actions that send out a new email, either on the same thread or a new one.

Software interaction In our pilot study we find some of the most likely recipient actions to be interaction with office softwares such as Setup-Appointment and Approve-Request.

Share content On many occasions, the most likely actions are to share a document, either as an attachment or via other means. We have an umbrella action Share-Content to capture these actions.

2.3 Data Annotation

Action	Description
Reply-Yesno	Short yes/no reply to a question raised in the previous email
Reply-Ack	Simple acknowledgements such as 'got it', 'thank you.'
Reply-Other	Reply to the thread based on information that is available *without doing any additional investigation.*
Investigate	Look into some questions/problems to gather the necessary information and reply with that information.
Send-New-Email	Write a new email that is not a reply to the current thread.
Setup-Appointment	Set up appointments/cancel appointments.
Approve-Request	Approve requests (typically from subordinates) through an external system such as an expense report system etc.
Share-Content	Share content, as an attachment, a link in the email body, or a location on the network that is known to both the sender and recipients

Table 1: Set of possible recipient actions in our annotation scheme.

A subset of the preprocessed email threads described in section 2.1 are subsequently annotated. We ask each annotator to imagine that they are a recipient of threaded emails in a workplace environment. For each message, we ask the annotator to read through the previous messages in the thread, and annotate with the most likely action (in table 1) they may perform if they had been the addressee of that message. If the most probable action is not defined in our list, we ask the annotators to annotate with an Other action.

A total of 399 emails from 110 distinct threads have been annotated by two paid and trained independent annotators. Cohen's Kappa is 0.75 for the two annotators. The authors arbitrated the disagreements. We include the distribution across the actions in table 1.

[1] We considered other email corpora such as the Enron corpus (Klimt and Yang 2004). We decided to use the Avocado dataset because it is the largest and newest one publicly available.

[2] The summary statistics are in table 3.

Dataset	Message
IRC	`could somebody explain how i get the oss compatibility drivers to load automatically in ubuntu ?`
IRC	`you should try these ones , apt src deb __URL__ unstable/`
IRC	`Ah , cool . Thanks , I 'll try that .`
Reddit	`Does this really appeal to Sanders supporters ? Can one ( or more of you ) explain to me why ? Full disclosure : I do n't pay ATM fees .`

Table 2: Some example non-email messages that are likely to elicit actions related to those observed in email data. IRC chats are very task specific. They are mostly about obtaining technical help. Therefore, we observe many conversational turns that start with information requests, followed by delivery of that information. The Reddit dataset, on the other hand, is more diverse: the discussions in *r/politics* more or less pertain to comments on American public policies and politics. We rarely observe messages that require the recipient to take action; but there are requests and deliveries of information which can potentially help learn the underlying representation.

Dataset name (type)	# of threads	# of messages	Average thread length	Average message length
Avocado (Email)	50 890	121 917	2.4	73.0
r/politics (Reddit)	15 813	42 952	2.7	31.4
Ubuntu Dialog (IRC)	50 812	416 721	8.2	12.7

Table 3: Statistics of conversational data used in this paper. During preprocessing we truncate each message to 256 words, including BOS and EOS symbols; and each thread to 32 messages. The original Ubuntu dataset is much larger (with $\approx 500\,000$ threads). We truncated it to match the Avocado dataset size for faster training and evaluation of our model.

2.4 Additional Domains

The annotations we collect are comparable in size to other speech act based annotation datasets. However like other expert-annotated datasets, ours is not large enough for end-to-end training. Therefore, we aim to enrich our training with additional semantic and pragmatic information derived from other tasks and domains without annotation for expected action. We consider data from the following additional domains for multidomain learning:

IRC The Ubuntu Dialog Corpus is a curated collection of chat logs from Ubuntu's Internet Relay Chat technical support channels (Lowe et al. 2015).

Reddit Reddit is an internet discussion community consisting of several *subreddits*, each of which is more or less a discussion forum pertaining to a certain topic. We curate a dataset from the subreddit *r/politics* over two consecutive months. Each entry in our dataset consists of the post title, an optional post body, and an accompanying tree of comments. We collect linear threads by recursively sampling from the trees.

Messages from IRC and Reddit are less precise in terms of speaker intents; and our recipient action scheme is not directly applicable to them. However, previous studies on speech acts in Internet forums and chatrooms have shown that there are speech acts common to all these heterogeneous domains, e.g. information requests and deliveries. Some such examples are listed in table 2. (Arguello and Shaffer 2015; Moldovan, Rus, and Graesser 2011) We hypothesize that more data from these domains will help recognition of these speech acts, which in turn help recognize the resulting recipient actions.

In all experiments in section 4, we use half of the dataset as training data, a quarter as the validation data and the remaining quarter as test data.

Identifier	Dataset	Description
E-T	Email	end of an email thread
E-A	Email	this message has attachment(s)
I-T	IRC	turntaking
R-T	Reddit	end of a Reddit thread

Table 4: Description of additional prediction labels for multitask learning that we extracted from datasets introduced in section 2.

2.5 Metadata-Derived Prediction Tasks

The datasets introduced in sections 2.1 and 2.4 are largely unlabeled as far as recipient actions are concerned, except for the small subset of Avocado data that was manually annotated. However we can still extract useful information from their metadata, such as inferred end-of-thread markers or system-logged events that can help us formulate additional prediction tasks for a multitask learning setting (listed in table 4). We also use these multitask labels to evaluate our multitask/domain model in section 4.3.

3 Modeling Threaded Messages

3.1 Notations

We model threaded messages as a two-layer hierarchy: at the lower layer we have a *message* $\mathbf{m}$ consisting of a list of words: $\mathbf{m} = [w_{1\ldots|\mathbf{m}|}]$. And in turn, a *thread* $\mathbf{x}$ is a list of messages: $\mathbf{x} = [\mathbf{m}_{1\ldots|\mathbf{x}|}] \in \mathcal{X}$. We assume each message thread to come from a specific domain; and therefore define a many-to-one

mapping $f(\mathbf{x}) = d$ where $d \in \mathcal{D}$ is the set of all domains. We also define the tasks $t \in \mathcal{T}$ to have a many-to-one mapping $g(t) = d, d \in \mathcal{D}$. For prediction we define the *predictor* of task t as $h_t(\mathbf{x}) : \mathcal{X} \rightarrow \mathcal{Y}$, which predicts sequential tags $\mathbf{y} = [y_1 \dots y_{|\mathbf{x}|}] \in \mathcal{Y}$ from a thread $\mathbf{x}$ on (a valid) task t. We also define the real-valued *task loss* of task t on thread $\mathbf{x}$ to be $\ell_t(\mathbf{y}, \hat{\mathbf{y}}) : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$, where $\hat{\mathbf{y}} \in \mathcal{Y}$ is the ground truth.

3.2 Definition of Multitask/domain Loss

In this paper, we define the *multitask loss* l_{MT} as the sum of task losses of tasks $\mathcal{T}_d$ under the same domain d for a single (output, ground truth) pair $(\mathbf{y}, \hat{\mathbf{y}})$:

$$l_{\text{MT}}(\mathcal{T}_d, \mathbf{y}, \hat{\mathbf{y}}) = \sum_{t \in \mathcal{T}_d} \ell_t(\mathbf{y}, \hat{\mathbf{y}}),$$

and the aggregate loss

$$L_{\text{MT}}(\mathcal{T}_d, \{\mathbf{y}^{(d)}_{1\dots K_d}, \hat{\mathbf{y}}^{(d)}_{1\dots K_d}\}) = \sum_{k=1}^{K_d} l_{\text{MT}}(\mathcal{T}_d, \mathbf{y}^{(d)}_k, \hat{\mathbf{y}}^{(d)}_k)$$

is the sum over K_d examples $\mathbf{y}^{(d)}_1 \dots \mathbf{y}^{(d)}_{K_d}$.

We also define the *multidomain loss* L_{MD} to be the sum of aggregate losses over $\mathcal{D}$:

$$L_{\text{MD}}(\{\{\mathbf{y}^{(d)}_{1\dots K_d}, \hat{\mathbf{y}}^{(d)}_{1\dots K_d}\} \mid d \in \mathcal{D}\}) = \sum_{d \in D} L_{\text{MT}}(\mathcal{T}_d, \{\mathbf{y}^{(d)}_{1\dots K_d}, \hat{\mathbf{y}}^{(d)}_{1\dots K_d}\}) \quad (1)$$

3.3 The Recurrent AttentIve Neural Bag-Of-Words model (Rainbow)

We start with the **R**ecurrent **A**ttent**I**ve **N**eural **B**ag-**O**f-**W**ord model (Rainbow) as the baseline model of threaded messages. From a high level view, Rainbow is a hierarchical neural network with two encoder layers: the lower level encoder is a neural bag-of-words encoder that encodes each message $\mathbf{m}$ into its message embeddings $b(\mathbf{m})$. And in turn, the upper level encoder transforms the independently encoded message embeddings $[b(\mathbf{m}_1) \dots b(\mathbf{m}_{|\mathbf{x}|})]$ into thread embeddings via a learned recurrent neural network $\mathbf{e}_1 \dots \mathbf{e}_{|\mathbf{x}|} = f_{\text{RNN}}(b(\mathbf{m}_1) \dots b(\mathbf{m}_{|\mathbf{x}|}))$.[3] Rainbow has three main components: message encoder, thread encoder, and predictor.

Message encoder. We implement the message encoder $b(\mathbf{m})$ as a bag of words model over the words in $\mathbf{m}$. Motivated by the unigram features in previous work on email intent modeling, we also add an attentive pooling layer (Rush, Chopra, and Weston 2015) to pick up important keywords. The averaged embeddings then undergo a nonlinear transformation:

$$\text{b}(\mathbf{m}) = q\left(\sum_{w \in \mathbf{m}} \frac{\exp(a(\mathbf{emb}(w)))}{\sum_{w' \in \mathbf{m}} \exp(a(\mathbf{emb}(w')))} \mathbf{emb}(w)\right), \quad (2)$$

where $q : \mathbb{R}^n \rightarrow \mathbb{R}^h$ is a learned feedforward network, $\mathbf{emb} : \mathbb{N} \rightarrow \mathbb{R}^n$ is the word embeddings of w and $a : \mathbb{R}^n \rightarrow \mathbb{R}$ is the (learned) attentive network that judges how much each word w contributes towards the final representation $b(\mathbf{m})$.[4]

Thread encoder and predictor. The message embeddings are passed onto the thread-level LSTM to produce a *thread embeddings* vector:

$$[\mathbf{e}_1 \dots \mathbf{e}_{|\mathbf{x}|}] = r(b(\mathbf{m}_1) \dots b(\mathbf{m}_{|\mathbf{x}|}))$$

Thread embeddings are then passed to the predictor layer. In this paper, the predictions are distributions over possible labels. We therefore define the predictor h_t to be a 2-layer feed forward network p that maps thread embeddings to distributions over V_t, the label set of task t: $h_t(\mathbf{e}_1 \dots \mathbf{e}_{|\mathbf{x}|}) = [p(\cdot \mid \mathbf{e}_1) \dots p(\cdot \mid \mathbf{e}_{|\mathbf{x}|})]$. The accompanying loss is naturally defined as the cross entropy between the predictions $p(\mathbf{e}_1) \dots p(\mathbf{e}_{|\mathbf{x}|})$ and the empirical distribution $\hat{\mathbf{y}} = \hat{y}_{1\dots|\mathbf{x}|}$:

$$\ell_t(\hat{\mathbf{y}}, \mathbf{e}_{1\dots|\mathbf{x}|}) = \frac{\sum_{i=1}^{|\mathbf{x}|} -\hat{y}_i \log p(\hat{y}_i \mid \mathbf{e}_i)}{|\mathbf{x}|}. \quad (3)$$

3.4 Multi-Task RNN Reparametrization

Rainbow is an extension of Deep Averaging Networks (Iyyer et al. 2015) to threaded message modeling. It works well for tagging threaded messages for the messages' properties, such as conversation-turn marking in online chats and end-of-thread detection in emails. However, in its current form, the model is trained to work on exactly one task. It also does not capture the shared dynamics of these different domains *jointly* when given out-of-domain data. In this section we describe a family of reparametrized recurrent neural networks that easily accommodates multi-domain multi-task learning settings.

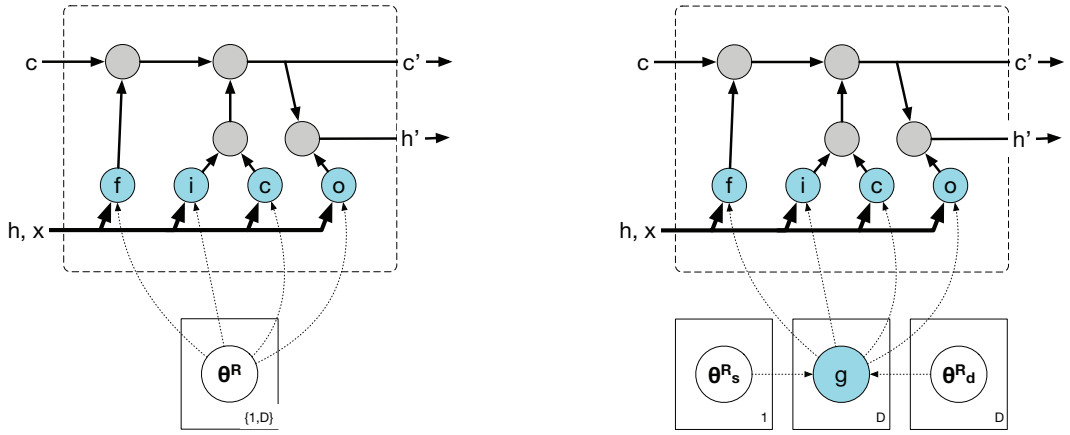
In general, recurrent neural networks take a sequence of input data $\mathbf{x}$ and recurrently apply a nonlinear function, to get a sequence of transformed representation $\mathbf{h}$. Here we denote such transformation with the function f_{RNN} parametrized by the RNN parameters $\boldsymbol{\theta}^R$ as $\mathbf{h} = f_{\text{RNN}}(\mathbf{x}; \boldsymbol{\theta}^R)$. For an LSTM model, $\boldsymbol{\theta}^R$ can be formulated as the concatenated vector of input, output, forget and cell gate parameters $[\mathbf{W}_i, \mathbf{W}_o, \mathbf{W}_f, \mathbf{W}_c]$. And in general, the goal of training an RNN is to find the optimal real-valued vector $\hat{\boldsymbol{\theta}}^R$ such that $\hat{\boldsymbol{\theta}}^R = \arg\min_{\boldsymbol{\theta}^R} L(f_{\text{RNN}}(\mathbf{x}; \boldsymbol{\theta}^R))$, for a given loss function L.

In the context of multidomain learning, we parametrize eq. (1) in a similar fashion:

$$L_{\text{MD}}(\{\{\mathbf{y}^{(d)}_{1\dots K_d}, \hat{\mathbf{y}}^{(d)}_{1\dots K_d}\} \mid d \in \mathcal{D}\}) =$$
$$L_{\text{MD}}(\{\{h(\mathbf{x}^{(d)}_1) \dots h(\mathbf{x}^{(d)}_{K_d}), \hat{\mathbf{y}}^{(d)}_{1\dots K_d}\} \mid d \in \mathcal{D}\}) =$$
$$\sum_{t, \mathbf{x}, \hat{\mathbf{y}}} \ell_t(h(\mathbf{x}), \hat{\mathbf{y}}; \boldsymbol{\theta}^R_t).$$

[3] There is a slight abuse of annotation since f_{RNN} actually differs for $\mathbf{x}$ of different lengths.

[4] There may be concerns about the unordered nature of the neural bag-of-words (NBOW) model. However it has been shown that with a deep enough network, an NBOW model is competitive against syntax-aware RNN models such as Tree LSTMs(Tai, Socher, and Manning 2015). In preliminary experiments we did not find the difference between an NBOW and an RNN to be substantial. But the NBOW architecture trains much faster.



(a) LSTM cell (b) Parameter-sharing LSTM cell

Figure 1: A comparison between partial computation graphs of a single (vanilla) LSTM cell, and our proposed parameter-sharing variants described in section 3.4. White circles are learned parameters. Dotted connections indicate parametrization. Parametrized and non-parametrized functions are indicated with blue and gray circles respectively. To model sequences from multiple domains, the conventional LSTM (depicted in fig. 1a) either shares everything with a set of parameters (the TIED setup; $i = 1$) or do not share parameters at all (the DISJOINT setup; $i = D$). In contrast, our parameter-sharing variant in fig. 1b models domain-invariant parameters with $\boldsymbol{\theta}_s$ and domain-specific parameters with $\{\boldsymbol{\theta}_d\}$.

Here we are faced with two modeling choices (depicted in fig. 1a): we can either model every task t DISJOINTly or with TIED parameters. The DISJOINT approach learns a separate set of parameters $\boldsymbol{\theta}^R_t$ per task t. Therefore, performance of a task is little affected by data from other domain/tasks, except for the regularizing effect through the word embeddings.

On the other hand the TIED approach ties parameters of all domains to a single $\boldsymbol{\theta}^R$, which has been a popular choice for multitask/domain modeling — it has been found that the RNN often learns to encode a good shared representation when trained jointly for different tasks (Collobert et al. 2011; Yang, Salakhutdinov, and Cohen 2016). The network also seems to generalize over different domains, too (Ragni et al. 2016; Peng and Dredze 2016). However it hinges on the assumption that either all domains are similar, or the network is capable enough to capture the dynamics of data from all domains at the same time.

In this paper we propose an alternative approach. Instead of having a single set of parameters $\hat{\boldsymbol{\theta}}^R$ for all domains, we propose to reparametrize $\boldsymbol{\theta}^R$ as a function ϕ of **s**hared components $\boldsymbol{\theta}^{R_s}$ and **d**omain specific components $\boldsymbol{\theta}^{R_d}$. Namely:

$$\boldsymbol{\theta}^R = \phi(\boldsymbol{\theta}^{R_s}, \boldsymbol{\theta}^{R_d}), \tag{4}$$

and our goal becomes minimizing the loss w.r.t both $\boldsymbol{\theta}^{R_s}, \boldsymbol{\theta}^{R_d}$:

$$\hat{\boldsymbol{\theta}}^{R_s}, \hat{\boldsymbol{\theta}}^{R_d} = \arg\min_{\boldsymbol{\theta}^{R_s}, \boldsymbol{\theta}^{R_d}} \sum \ell_t(\mathbf{x}, \hat{\mathbf{y}}; \boldsymbol{\theta}^{R_s}, \boldsymbol{\theta}^{R_d}). \tag{5}$$

A comparison between the vanilla RNN and our proposed modification can be found in fig. 1. This reparametrization allows us to share parameters among networks trained on data of different domains with the shared component $\boldsymbol{\theta}_s$, while allowing the network to work differently on data from each domain with the domain specific parameters $\boldsymbol{\theta}_d$.

The design of the function ϕ requires striking a balance between model flexibility and generalizability. In this paper we consider the following variants of ϕ:

Additive (ADD) First we consider ϕ to be a linear interpolation of a shared base $\boldsymbol{\theta}^{R_s}$ and a network specific component $\boldsymbol{\theta}^{R_d}$:

$$\boldsymbol{\theta}^R = \phi_{\text{ADD}}(\boldsymbol{\theta}^{R_s}, \boldsymbol{\theta}^{R_d}; u_d) = \boldsymbol{\theta}^{R_s} + \exp(u_d)\boldsymbol{\theta}^{R_d}, \tag{6}$$

where $u_d \in \mathbb{R}$. In this formulation ADD we learn a shared $\boldsymbol{\theta}^{R_s}$, and additive domain-specific parameters $\{\boldsymbol{\theta}^{R_d} \mid d \in \mathcal{D}\}$ for each domain. We also learn u_d for each domain $d \in \mathcal{D}$, which controls how much effect $\boldsymbol{\theta}^{R_d}$ has on the final parameters.

Both DISJOINT and TIED can be seen as degenerate cases of ADD: we recover DISJOINT when the shared component is a zero vector: $\boldsymbol{\theta}^{R_s} = \mathbf{0}$ And with $u_d \to -\infty$ we have $\boldsymbol{\theta}^R = \boldsymbol{\theta}^{R_s}$, namely TIED.

Additive + Multiplicative (ADDMUL) ADD has no nonlinear interaction between $\boldsymbol{\theta}^{R_s}$ and $\boldsymbol{\theta}^{R_d}$: they have independent effects on the composite $\boldsymbol{\theta}^R$. In ADDMUL we have two components in $\boldsymbol{\theta}^{R_d} = [\boldsymbol{\theta}^{R_{da}}, \boldsymbol{\theta}^{R_{dm}}]$: the additive component $\boldsymbol{\theta}^{R_{da}}$ and the multiplicative component $\boldsymbol{\theta}^{R_{dm}}$ which introduces nonlinearity without significantly increasing the parameter count:

$$\begin{aligned}\boldsymbol{\theta}^R &= \phi_{\text{ADDMUL}}(\boldsymbol{\theta}^{R_s}, \boldsymbol{\theta}^{R_d}; u_d, v_d) \\ &= \boldsymbol{\theta}^{R_s} + \exp(u_d)\boldsymbol{\theta}^{R_{da}} + \exp(v_d)\boldsymbol{\theta}^{R_{dm}} \otimes \boldsymbol{\theta}_s,\end{aligned} \tag{7}$$

where $\otimes$ is the Hadamard product and $u_d, v_d \in \mathbb{R}$ are learned parameters as in the ADD formulation.

Affine (AFFINE) In this formulation $\boldsymbol{\theta}^{R_d}$ are seen as *task embeddings*. We apply a learned affine transformation $\mathbf{W}$ to

the task embeddings and add up the shared component $\boldsymbol{\theta}^{R_s}$:

$$\boldsymbol{\theta}^{R} = \textsc{Affine}(\boldsymbol{\theta}^{R_s}, \boldsymbol{\theta}^{R_d}; \mathbf{W}) = \boldsymbol{\theta}^{R_s} + \mathbf{W}\boldsymbol{\theta}^{R_d}, \quad (8)$$

where $\mathbf{W}$ is a learned parameter.

3.5 Optimization

We optimize for the multidomain loss as defined in eq. (1) with gradient descent methods. To update parameters, we sample one thread from each domain $\{\mathbf{m}_d \mid d \in \mathcal{D}\}$ and optimize the network parameters with the ADAM optimizer.(Kingma and Ba 2014)

4 Experiments

4.1 Evaluation Metrics

In this section we evaluate Rainbow and its multi-task/multidomain variants on the datasets we introduced in section 2. We also apply our extracted thread embeddings on a real-world task setting of email action classification with impoverished resources.

Probabilistic models are usually evaluated on the log-likelihood of the test data $S = \{(\mathbf{x}_1, \hat{\mathbf{y}}_1) \ldots (\mathbf{x}_{|S|}, \hat{\mathbf{y}}_{|S|})\}$: $\sum_{(\mathbf{x},\hat{\mathbf{y}})\in S} \log p(\hat{\mathbf{y}} \mid \mathbf{x})$. However, in our multidomain setting we have multiple datasets that differ in size and average sequence length. Therefore we evaluate our models on mean average cross entropy (MACE):

$$\text{MACE}(S) = \frac{-\sum_{(\mathbf{x},\hat{\mathbf{y}})\in S}(1/|\mathbf{y}|)\cdot\left(\sum_{i=1}^{|\hat{\mathbf{y}}|}\log p_t(\hat{y}_i \mid \mathbf{e})\right)}{|S|}, \quad (9)$$

where $\mathbf{e}$ are the thread embeddings of $\mathbf{x}$, and $p_t(\cdot \mid \mathbf{e})$ follow the definition in section 3.3. MACE normalizes by both sequence length $|\mathbf{y}|$ and dataset length $|S|$: a model that ignores the resource-poor tasks or short sequences tends to perform poorly under this metric. MACE can therefore be seen as per-task (log) perplexity: a larger MACE value means the model performs worse on the dataset; and the oracle would obtain a MACE value of 0. The average of MACE scores also has the natural interpretation of the geometric mean of log likelihoods over different tasks/domains. In addition to MACE, we also evaluate on accuracy in table 6.

All experiments in section 4 are trained on train splits. For experiments in sections 4.2 and 4.3 we evaluated on metadata-derived features in table 4. After each epoch of training, the model is evaluated on the validation split to check if the performance has stopped increasing. The training procedure terminates when no new performance gains are observed for two consecutive epochs.

4.2 Effectiveness of Rainbow: Ablation Studies

We evaluate Rainbow by comparing it, in the single task setup, against two simpler variant architectures: one is taking away the **r**ecurrent thread encoder (**-R**), the other is replacing the **a**ttentive pooling layer with an unweighted mean (**-A**). We evaluate the four configurations on the four labels listed in table 4 and report the averaged MACE numbers in table 5. We find that both attentive pooling and the recurrent network help; but the latter has a much more pronounced effect. Rainbow without the two additions (**-R**, **-A**) is reduced to the vanilla Deep Average Network model, a neural baseline that has been shown to be competitive against other neural and non-neural models.

Configuration	+R	-R
+A	0.0796	0.1163
-A	0.0800	0.1174

Table 5: MACE values of the Rainbowablation tests (lower is better). **+/-R** and **+/-A** indicates the presence/absence of the thread encoder and the attentive pooling layer, respectively.

4.3 Multidomain/task Experiments

We compare our reparametrized models against the following feature-reparametrizing approaches:

MaLOPa For each task t, we concatenate the word embeddings $\mathbf{emb}(w)$ with task embeddings $\mathbf{k}_t$: $[\mathbf{emb}(w); \mathbf{k}_t]$. $\mathbf{k}_t$ are trained along with the network, and hopefully contains task-relevant information. This idea originated from the MaLOPa (MAny Language One PArser) parser (Ammar et al. 2016).

Fenda In this setting, each task has its own predictor and two message encoders, one shared and the other specific to itself. The two encoder outputs are concatenated, linearly transformed, and fed into the predictor. This is an adaptation of the Fenda(Frustratingly Easy Neural Domain Adaption) model in (Kim, Stratos, and Sarikaya 2016), which in turn is a neural extension of the classic paper by Daume III (2007).

We also compare them against the two baselines:

Disjoint Each task has its own predictor, thread encoder, and message encoder.

Tied Each task has its own predictor. All tasks share the same thread encoder and message encoder. As we noted in section 3.4 it has been empirically found that the model is capable of learning a shared representation across tasks and domains.(Glorot, Bordes, and Bengio 2011)

We evaluate our proposed models, feature-reparametrizing models, and the non-domain-adaptive baselines on tasks listed in table 4 in these following multidomain/multitask transfer settings: **(E)**, **(E+I)**, **(E+R)**, **(I+R)**, **(E+I+R)**, where **E**=Email, **I**=IRC, **R**=Reddit. Note that since only the emails have two meta features E-A and E-T, we have **(E)** as our only multitask transfer setting. The results are in table 6. Difference between results from all models is small. We inspected the model outputs and found they all suffer severely from the label bias problem — all four tasks have very unbalanced label distributions; and the network learns to strongly favor the more frequent label. The label bias problem can potentially be addressed by using a globally normalized model which we leave as future work. Despite the small margins, we can see that both model- and feature-reparametrizing models outperform the baselines in terms of likelihood. Moreover, our reparametrized models consistently achieve higher likelihood

Task	E-T		E-A		I-T		R-T		Average	
	MACE	Acc	MACE	Acc	MACE	Acc	MACE	Acc	MACE	Acc
Aggregated Results										
ADD	**0.0919**	**81.0**	0.0509	93.4	**0.0741**	22.3	**0.1039**	**68.0**	**0.3208**	66.2
ADDMUL	0.0927	**81.0**	0.0510	93.3	0.0742	22.4	0.1050	67.6	0.3229	66.1
AFFINE	0.0930	81.0	**0.0502**	93.4	0.0741	22.7	0.1055	66.2	0.3228	65.8
DISJOINT	0.0933	80.9	0.0507	**93.4**	0.0742	22.4	0.1078	65.4	0.3260	65.5
TIED	0.0937	80.7	0.0518	93.0	0.0744	22.7	0.1048	67.1	0.3246	65.9
MALOPA	0.0939	81.0	0.0514	93.3	0.0744	**22.8**	0.1044	67.9	0.3241	**66.2**
FENDA	0.0919	80.8	0.0516	93.1	0.0741	22.8	0.1054	67.7	0.3231	66.1

Table 6: Aggregated Multidomain/multitask results of tasks in table 4: **bold** indicates best average results over all models.

than baselines on test data in all transfer settings. In addition, ADD and ADDMUL perform comparably well against strong domain-adaptive models in terms of accuracy.

4.4 Recipient Action with Minimal Supervision

Setting	F_1
ADD	32.7*
ADDMUL	32.6
AFFINE	31.7
DISJOINT	27.9
TIED	30.7
MALOPA	30.7
FENDA	31.4
DOC2VEC	26.7

Table 7: Results of section 4.4. ADD is significantly outperforming the best baseline FENDA($p = 0.0443$) and ADDMUL borderline significant ($p = 0.0816$) against FENDA , the best-performing domain-adaptive baseline model under paired t-test. The difference between ADDand ADDMUL against other baseline models are also significant under paired t-test. Hyperparameters are regularization strength C and transfer setting.

	F_1			
Setting	E	E+I+R	E+R	E+I
ADD	24.3	30.2	26.0	32.7
ADDMUL	22.9	30.3	27.8	33.1
AFFINE	30.8	28.4	26.6	33.1
DISJOINT	27.6	29.3	26.2	25.8
TIED	27.2	31.2	25.2	30.9

Table 8: Breakdown on different transfer settings.

We now turn to a task-based evaluation where we use our extracted thread embeddings on the task of predicting an email recipient's next action. In particular, we focus on scenarios where we do not have a sizable amount of annotated data to train a neural network in an end-to-end fashion, and when we simply did not anticipate the task when we trained the model. This setting evaluates the network's ability to generalize over multiple tasks and learn a good representation.

To be more specific, the setup is as follows: we use trained models from section 4.3 to encode thread embeddings from action-annotated emails $\{\mathbf{m}_a\}$ of section 2. Subsequently we use these thread embeddings to train L_2-regularized logistic regression classifiers for the action labels. We compare them against classifiers trained with features extracted from the baselines TIED, DISJOINT, MALOPA, and FENDA. We also compare it against doc2vec embeddings trained on the whole Avocado corpus (listed in table 7 as DOC2VEC).

Given the small size of annotated data, we decide to evaluate the models with nested cross validation (CV). In the outer layer, we randomly split the annotated emails into (train+dev)-test splits,[5] in a thread-wise fashion. In the inner layer, we use 7-fold CV on the (train+dev) split to find the best hyperparameters. The best hyperparameters are then used to train a classifier, which is subsequently evaluated on the test split of the outer layer CV. We report the average F_1 in table 7. DISJOINT performs poorly on this task since there is no baked-in constraint for it to learn a shared representation. All shared-representation baselines (TIED, FENDA, MALOPA) performed better than both DISJOINT and DOC2VEC. Still, our reparametrized models compare favorably against the feature-reparametrizing baselines.

We do another cross validation evaluation, over different transfer settings in table 8. It seems that while both Reddit **(E+R)** and the IRC **(E+I)** datasets do better than email only **(E)**, the IRC dataset is much more helpful than Reddit. This resonates with our initial findings in section 2.4 that the IRC dataset is more similar to emails. We note that all the F_1 scores are low. Nonetheless we find it encouraging that out-of-domain data is able to help learn a better representation in this extremely resource-scarce setting.

5 Related Work

There has been a lot of work on multidomain/task learning with shared representation as we described in section 1. Our work is also closely related to work on email speech act modeling and recognition (Cohen, Carvalho, and Mitchell 2004; Lampert et al. 2008; Jeong, Lin, and Lee 2009; De Felice

[5] 120 splits with a ratio of (0.67, 0.33)

and Deane 2012). The idea of model reparametrization for domain adaption is abundant in the literature of hierarchical Bayesian modeling, such as Finkel and Manning; Eisenstein, Ahmed, and Xing (2009; 2011).

Within the deep learning literature, our work is also related to work on DNN reparametrization for multitask learning, such as Spieckermann, Udluft, and Runkler; Yang and Hospedales (2014; 2016). Our work shows the reparametrization approach also works for domain adaptation. Finally we would like to point out that Ha, Dai, and Le (2016) introduces an alternative and much more sophisticated reparametrization of RNNs. An interesting future direction of our work is to follow this work by reparametrizing networks as hypernetworks that take a task embedding as an input. In that case, using the terminology introduced in this paper, we will be feature-reparametrizing the hypernetwork; which in turn model-reparametrizes an RNN.

6 Conclusion

In this paper, we have introduced an email recipient action annotation scheme, and a dataset annotated according to this scheme. By annotating the recipient action rather than the sender's intent, our taxonomy is agnostic to specific speech act theories, and arguably more suitable for training systems that suggest such actions. We have curated an annotated dataset, which achieved good inter-annotator agreement levels. We have also introduced a hierarchical threaded message model Rainbow to model such emails. To cope the problem of data scarcity, we have introduced RNN reparametrization as an approach to domain adaptation, and applied it onto the problem of email recipient action modeling. It is competitive against common feature-reparametrized neural models when trained in an end-to-end fashion. We also show that while it is not explicitly designed to encode a shared representation across tasks and domains, it learns to generalize in a minimally supervised scenario. There are many possible future directions of our work. For example, with appropriate software, we can obtain more annotation automatically, and possibly learn the taxonomy along. Also our reparametrization framework is quite extensible. For instance, user-specific parameters for each user can be learned for personalized models, as in Li et al. (2016).

References

Agema, L. 2015. Death by e-mail overload.

Ammar, W.; Mulcaire, G.; Ballesteros, M.; Dyer, C.; and Smith, N. 2016. Many languages, one parser. *Transactions of the Association for Computational Linguistics* 4:431–444.

Arguello, J., and Shaffer, K. 2015. Predicting speech acts in mooc forum posts. In *ICWSM*, 2–11.

Caruana, R. 1997. Multitask learning. *Machine Learning* 28(1):41–75.

Cohen, W.; Carvalho, V.; and Mitchell, T. 2004. Learning to classify email into "speech acts". In *EMNLP*.

Collobert, R., and Weston, J. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, 160–167. ACM.

Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; and Kuksa, P. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.* 12:2493–2537.

Corston-Oliver, S. H.; Ringger, E.; Gamon, M.; and Campbell, R. 2004. Task-focused summarization of email. In *ACL*.

Daume III, H. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, 256–263. Prague, Czech Republic: Association for Computational Linguistics.

De Felice, R., and Deane, P. 2012. Identifying speech acts in e-mails: Toward automated scoring of the toeic e-mail task. *ETS Research Report Series* 2012(2):i–62.

Dredze, M.; Brooks, T.; Carroll, J.; Magarick, J.; Blitzer, J.; and Pereira, F. 2008. Intelligent email: Reply and attachment prediction. In *IUI*.

Eisenstein, J.; Ahmed, A.; and Xing, E. P. 2011. Sparse additive generative models of text. In Getoor, L., and Scheffer, T., eds., *ICML*, 1041–1048. Omnipress.

Finkel, J. R., and Manning, C. D. 2009. Hierarchical bayesian domain adaptation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, 602–610. Stroudsburg, PA, USA: Association for Computational Linguistics.

Glorot, X.; Bordes, A.; and Bengio, Y. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, 513–520.

Goldstein, J., and Sabin, R. E. 2006. Using speech acts to categorize email and identify email genres. In *System Sciences, 2006. HICSS'06. Proceedings of the 39th Annual Hawaii International Conference on*, volume 3, 50b–50b. IEEE.

Ha, D.; Dai, A. M.; and Le, Q. V. 2016. Hypernetworks. *CoRR* abs/1609.09106.

Iyyer, M.; Manjunatha, V.; Boyd-Graber, J. L.; and Daumé, H. 2015. Deep unordered composition rivals syntactic methods for text classification. In *ACL*.

Jeong, M.; Lin, C.-Y.; and Lee, G. G. 2009. Semi-supervised speech act recognition in emails and forums. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3*, EMNLP '09, 1250–1259. Stroudsburg, PA, USA: Association for Computational Linguistics.

Kannan, A.; Kurach, K.; Ravi, S.; Kaufmann, T.; Tomkins, A.; Miklos, B.; Corrado, G.; Lukács, L.; Ganea, M.; Young, P.; et al. 2016. Smart reply: Automated response suggestion for email. *arXiv preprint arXiv:1606.04870*.

Khoussainov, R., and Kushmerick, N. 2005. Email task management: An iterative relational learning approach. In *CEAS*.

Kim, Y.-B.; Stratos, K.; and Sarikaya, R. 2016. Frustratingly

easy neural domain adaptation. ACL âĂŞ Association for Computational Linguistics,.

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

Klimt, B., and Yang, Y. 2004. *The Enron Corpus: A New Dataset for Email Classification Research*. Berlin, Heidelberg: Springer Berlin Heidelberg. 217–226.

Lampert, A.; Dale, R.; Paris, C.; et al. 2008. The nature of requests and commitments in email messages. In *Proceedings of the AAAI Workshop on Enhanced Messaging*, 42–47.

Li, J.; Galley, M.; Brockett, C.; Spithourakis, G. P.; Gao, J.; and Dolan, W. B. 2016. A persona-based neural conversation model. *CoRR* abs/1603.06155.

Lowe, R.; Pow, N.; Serban, I.; and Pineau, J. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *arXiv preprint arXiv:1506.08909*.

Moldovan, C.; Rus, V.; and Graesser, A. C. 2011. Automated speech act classification for online chat. *MAICS* 710:23–29.

Oard, D.; Webber, W.; Kirsch, D.; and Golitsynskiy, S. 2015. Avocado research email collection. DVD.

Peng, N., and Dredze, M. 2016. Multi-task multi-domain representation learning for sequence tagging. *arXiv preprint arXiv:1608.02689*.

Ragni, A.; Dakin, E.; Chen, X.; Gales, M. J.; and Knill, K. M. 2016. Multi-language neural network language models. In *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, volume 8, 3042–3046.

Riezler, S. 2014. On the problem of theoretical terms in empirical computational linguistics. *Computational Linguistics* 40(1):235–245.

Rush, A. M.; Chopra, S.; and Weston, J. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Searle, J. R. 1976. A classification of illocutionary acts. *Language in society* 5(01):1–23.

Spieckermann, S.; Udluft, S.; and Runkler, T. 2014. Data-efficient temporal regression with multi-task recurrent neural networks.

Tai, K. S.; Socher, R.; and Manning, C. D. 2015. Improved semantic representations from tree-structured long short-term memory networks. *CoRR* abs/1503.00075.

Traum, D. R. 1999. Speech acts for dialogue agents. In *Foundations of rational agency*. Springer. 169–201.

Yang, Y., and Hospedales, T. M. 2014. A Unified Perspective on Multi-Domain and Multi-Task Learning. *ArXiv e-prints*.

Yang, Y., and Hospedales, T. M. 2016. Deep multi-task representation learning: A tensor factorisation approach. *CoRR* abs/1605.06391.

Yang, Z.; Salakhutdinov, R.; and Cohen, W. W. 2016. Multi-task cross-lingual sequence tagging from scratch. *CoRR* abs/1603.06270.

Yang, Z.; Salakhutdinov, R.; and Cohen, W. W. 2017. Transfer Learning for Sequence Tagging with Hierarchical Recurrent Networks. *ArXiv e-prints*.