Dialogue Act Classification for Virtual Agents for Software Engineers during Debugging

Andrew Wood
Boston University
Boston, MA, USA
aewood@bu.edu

Zachary Eberhart
University of Notre Dame
Notre Dame, IN, USA
zeberhar@nd.edu

Collin McMillan
University of Notre Dame
Notre Dame, IN, USA
cmc@nd.edu

ABSTRACT

A “dialogue act” is a written or spoken action during a conversation. Dialogue acts are usually only a few words long, and are often categorized by researchers into a relatively small set of dialogue act types, such as eliciting information, expressing an opinion, or making a greeting. Research interest into automatic classification of dialogue acts has grown recently due to the proliferation of Virtual Agents (VA) e.g. Siri, Cortana, Alexa. But unfortunately, the gains made into VA development in one domain are generally not applicable to other domains, since the composition of dialogue acts differs in different conversations. In this paper, we target the problem of dialogue act classification for a VA for software engineers repairing bugs. A problem in the SE domain is that very little sample data exists – the only public dataset is a recently-released Wizard of Oz study with 30 conversations. Therefore, we present a transfer-learning technique to learn on a much larger dataset for general business conversations, and apply the knowledge to the SE dataset. In an experiment, we observe between 8% and 20% improvement over two key baselines.

KEYWORDS

dialogue act classification, transfer learning, intelligent agents, software engineering

ACM Reference Format:
Andrew Wood, Zachary Eberhart, and Collin McMillan. 2020. Dialogue Act Classification for Virtual Agents for Software Engineers during Debugging. In *IEEE/ACM 42nd International Conference on Software Engineering Workshops (ICSEW’20), May 23–29, 2020, Seoul, Republic of Korea.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3387940.3391487

1 INTRODUCTION

A **dialogue act** (DA) is a written or spoken action taken during a conversation. For example, an utterance “turn left at the next light” is an instruction, versus a request for information such as “when should I turn?” Dialogue acts are important components of discourse modeling, and have been studied for decades: first in sociology [5, 37] and later in computational linguistics [36]. DAs are “the minimal unit of linguistic communication” [37] and are key to both automated comprehension and generation of natural language dialogue.

Dialogue act classification refers to the automated labeling of utterances with the appropriate DA types. It is a well-studied problem that has largely followed the history of other areas of applied machine learning: early attempts involved manually-curated feature sets based on word usage [2], later enhanced by knowledge of relationships between DA types such as common conversation flows [6]. Today, state-of-the-art performance on large datasets is achieved by neural network-based architectures [13, 20, 22].

Interest in DA classification has ballooned due to the proliferation of automated virtual agents (VAs) such as Siri, Cortana, and Alexa. One of the first tasks a VA must perform is classification of a human conversation participant’s utterance into a dialogue act; it must know whether it is e.g. being given an command or being asked a question before it can craft a sensible reply. And for many VAs, large and context-appropriate datasets exist to train strong classifiers [39]. For example, a designer of a VA for appointment scheduling could draw on a dataset with many thousands of utterances [11].

But this straightforward approach is not available for many specialty applications where datasets are rare and expensive. As Gangadharaiah *et al.* [16] pointed out, “methods that achieve state of the art performance on synthetic datasets perform poorly in real world dialog tasks.” Likewise, Kang *et al.* [21] emphasize that methods trained on datasets in one domain tend not to generalize well to problems in other domains, since the number and composition of dialogue act types differ dramatically.

In this paper, we target the problem of dialogue act classification for virtual agents in the domain of *software engineering* (SE). Specifically, we envision building a VA to assist software engineers during debugging: the imagined situation is that a programmer receives a bug report, and has questions about the codebase in which the bug occurs. A VA is desirable in this situation because debugging is often assigned to junior programmers who are learning a new codebase [7] and may have more questions than senior colleagues can handle [35].

Wood *et al.* [44] have found that discourse structure is quite different in SE compared to the in other domains; unfortunately, datasets in this domain are expensive to create. The only relevant public dataset, released by Wood *et al.* [44], required over a year of effort and many thousands of dollars in recruitment and technology costs. Even so, it involves only 30 participants and a few thousand utterances – generally speaking, it is not enough to train a state-of-the-art system.

Therefore, we propose a transfer learning architecture to “learn what we can” from an available, large dataset and apply it to the

ICSEW’20, May 23–29, 2020, Seoul, Republic of Korea
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7963-2/20/05…$15.00
https://doi.org/10.1145/3387940.3391487

specialized SE VA problem we target. Our approach works by combining two encoders: a "global" encoder trained with a large corpus of generic conversations, and a "local" encoder trained with the limited SE domain-specific data.

We evaluate our approach against two baselines: 1) a recent applicable approach from related literature trained only on the limited available domain-specific dataset, and 2) the same algorithm trained with data from a large corpus of generic conversations. We select a subset of dialogue act types as a target set, since not all types are of equal value for a VA to recognize. Experimental results show that our approach outperforms the baselines by 8% and 20% on the target dialogue act types.

2 PROBLEM AND SCOPE

The long-term vision of this paper is an automated virtual agent for software engineers during debugging. A recent book by Rieser and Lemon [32] outlines an accepted process for designing natural language dialogue systems. The book's recommendation is to start with Wizard of Oz (WoZ) experiments to collect simulated data, then study the data to learn and predict the dialogue act types, followed by reinforcement learning to optimize response strategy, and language generation to produce comprehensible replies.

A recent paper by Wood *et al.* [44] released one of the only experimental WoZ datasets in the field of software engineering designed for building VAs. The paper reports a manual identification of the DA types, and a method for automatically predicting them. The automatic method is a simple bag-of-words SVM-based text classifier, which the paper states is only intended to establish a baseline.

We pick up where Wood *et al.* [44] left off, using the SE WoZ dataset. Following the process recommended by Rieser and Lemon, we have attempted to build an effective dialogue act classifier based on the published state-of-the-art. However, in pilot studies we found results similar to Gangadharaiah *et al.* [16] and Kang *et al.* [21]; the latest technology often does not achieve usable results in practice.

In our view, there are two reasons for the relatively low state-of-the-art performance: first, the size of the available data is limited in our application, compared to the large synthetic datasets used in many papers on dialogue act classification. The neural algorithms used in recent papers are notorious for requiring tens of thousands or more examples for training for good results in text classification. The WoZ dataset we use is not even one tenth as large.

Second, the discourse structure in the dataset we use is quite different, meaning that the DA classification accuracy levels reported for large synthetic datasets do not apply. The majority of papers on dialogue act classification report an overall accuracy for all classes. For example, Chen *et al.* [13] report a remarkable 91.7% accuracy, versus 90.9% for a baseline on a standard dataset. However, the performance can vary considerably for different dialogue act types. In our situation, we care a lot more about acts denoting information requests (given that we seek to answer questions) than we do about greetings or opinions. Also, it is important for us to distinguish differences such as a request for information and a request for an assessment. But dialogue act types associated with these are among the worst performers in the approach taken by Chen *et al.* [13]. Therefore, despite reporting high (90%+) overall accuracy, "off-the-shelf" approaches are not necessarily suitable for this domain.

Table 1: Selection of closely-related projects targeting Dialogue Act Classification.

Project	Year	W	M	H	N	P
Andernach [2]	1996	x	x			
Reithinger and Klesen [31]	1997	x		x		
Stolcke et al. [40]	2000	x	x	x		
Serafin et al. [38]	2003	x				
Grau et al. [19]	2004	x	x			
Ang et al. [4]	2005	x	x			
Surendran and Levow [41]	2006	x		x		
Geertzen et al. [17]	2007	x	x			
Zimmermann [45]	2009	x	x			
Boyer et al. [9]	2010			x		
Tavafi et al. [42]	2013	x		x		
Blunsom et al. [8]	2013	x	x		x	
Milajevs and Purver [28]	2014	x		x	x	
Khanpour et al. [22]	2016	x			x	
Ji et al. [20]	2016	x		x	x	
Lee and Dernoncourt [25]	2016	x			x	
Liu et al. [26]	2017	x		x	x	x
Kumar et al. [24]	2018	x		x	x	x
Chen et al. [13]	2018	x		x	x	x

3 BACKGROUND / RELATED WORK

This section summarizes the history of research on classification of dialogue acts and interactive natural language systems for software engineering.

3.1 Dialogue Act Classification

Table 1 summarizes the key related work on Dialogue Act Classification. Related work can be broadly categorized along the five dimensions in the table. Column *W* indicates whether the classifier uses words in the dialogue as features. *M* indicates whether the classifier uses manually-crafted features other than words (but which may be based on words, e.g. length). *H* indicates whether the history of previous dialogue acts in a conversation is used to predict the current act type. *N* indicates whether the approach is based on neural nets. *P* indicates whether the approach is post-hoc, that is whether it predicts all act types for a whole conversation, rather than one act at a time (i.e. ongoing conversations).

Observations include: 1) nearly all related work uses the words in a dialogue act as features for classification in one way or another; some projects rely on a bag-of-words representations, and others use n-grams or sequence-based representations (e.g. as provided by a recurrent neural network). 2) A clear shift from manually-crafted features to neural net-based approaches begins around the year 2013. This shift reflects the trend across many areas of NLP and AI research, as a recent survey by Chen *et al.* points out [12]. And, 3) several studies have found that history of previous dialogue act types in a conversation improves classification performance.

The neural net-based approaches can be further classified as either post-hoc or online. A post-hoc approach retroactively labels every utterance in an entire conversation, in contrast to an online approach which labels a "current" dialogue act given some conversation history. The approach taken by Kumar *et al.* [24] takes an entire dialogue as input, using recurrent "sentence level" layers that output to another recurrent "conversation level" layer that ultimately produces predictions for every utterance in a conversation. This structure is useful in post-hoc analysis, but does not fit the need for predictions in VAs, which must classify dialogue acts as soon as they are received.

3.2 Software Engineering Virtual Agents

Virtual Agents to assist software engineers during development have been envisioned for decades [43], but the recent proliferation of virtual agents for general tasks (e.g. Siri, Cortana, Alexa) has reignited research towards that vision [33]. Key related work includes WhyLine by Ko and Myers [23], which attempts to explain program behavior; TiQi by Pruski *et al.* [30], a dialogue system for database queries; and a natural language dialogue system by Escobar-Avila *et al.* [15] to accompany video tutorials. Most recently, Bradley *et al.* [10] designed a virtual agent that performs various tasks for programmers, and Wood *et al.* [44] simulated a virtual agent for bug repair in a WoZ experiment.

4 MODEL DESIGN

This section describes our model design. A key component of our model is the "dual encoder" design, in which one encoder is trained using a large **source** dataset of generic conversations, while another is trained on a small, specialized **target** dataset of conversations in the target domain. For our purposes, the target domain is software engineering debugging virtual agents, for which we use the small dataset provided by Wood *et al.* [44], and the source dataset is the AMI business conversation corpus provided by McCowan *et al.* [27].

4.1 Overview

Figure 1 shows an overview of our model. At a high level, it resembles many attentional encoder/decoder NMT systems, though many practical details differ. Of note, it makes use of two encoders: a **global** encoder and a **local** encoder. We train these encoders separately to provide the model with both an extensive knowledge base from a broad domain and the specialized knowledge available in the target domain.

During training, we first provide the global encoder (area 1) with every word sequence in the large source dataset. We train that encoder based on the label for each dialogue act (area 2), and we save all of the weights learned by that encoder.

We then train the local encoder (area 3) on the target dataset. To do so, we first reload the weights learned by the global encoder and set that encoder as not trainable. We then send the each word sequence in the target dataset through both encoders. The global encoder is not trainable at this point, and creates a representation based on the source dataset alone. The local encoder's initial state is set equal to the final state of the global encoder (area 4).

Next, we attend each state in the local encoder to states in the global encoder (area 5). The intent is to identify states where the global and local encoders are similar (i.e. where global knowledge is relevant to the local situation) while attenuating other states (i.e. less relevant global knowledge).

We create a context vector by concatenating the attended global states to the local states (area 6). We then use one fully-connected layer per concatenated state (area 7) to determine how to combine the

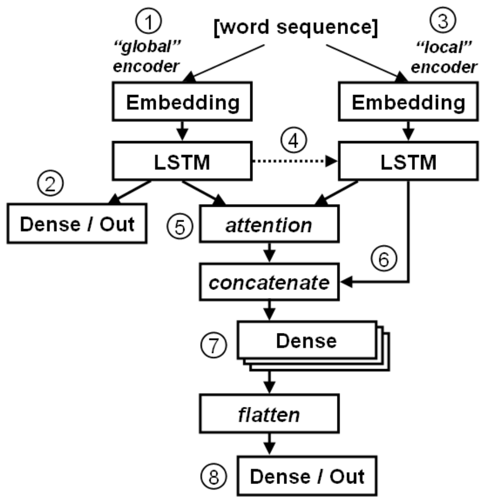

Figure 1: Overview of our model. See text in Section 4.1 for discussion of labeled areas.

global and local state (i.e. which should be used for classification). Finally, we flatten the output of these layers into a single vector and prepare for the final classification (area 8).

During inference, we provide a word sequence to both encoders, discard the output at area 2, and use only the output at area 8 as the final prediction.

4.2 Model Details

Given a corpus of utterances $C = \{\mathbf{u}_j\}_{j=0}^{k}$ composed of k utterances, where each utterance is a sequence of words $\mathbf{u}_j = \{w_{jn}\}_{n=0}^{l}$ that has a maximum utterance size l, we can construct a vocabulary V_C for corpus C such that $|V_C| = m$ and contains the top m most frequent words in C. We define our DA classification problem $\mathbf{f}$ as predicting a single DA type p_i out of d DA types given an utterance $\mathbf{u}_i$. We construct the solution as a decision function that computes the probability distribution over all potential DA types given an utterance $\mathbf{u}_j$:

$$\mathbf{f} = \{Pr p_i | \mathbf{u}_j\}_{i=0}^{d} \tag{1}$$

Specifically, our base models compute this decision function using an embedding function $\mathbf{e}$ to map each word into a continuous vector space $\mathbf{e} : w_i \in V_C \rightarrow \mathbb{R}^{1 \times n}$, a Long-Term-Short-Memory (LSTM) Recurrent Neural Network (RNN), and an output softmax layer. Formally, we can divide this decision function into two discrete parts: the **encoder** containing the embedding function and RNN encoder with q encoding dimensions, and the **predictor** containing the softmax layer. Given learnable encoder parameters $\mathbf{\Phi}_{enc}$ and learnable predictor parameters $\mathbf{\Phi}_{pred}$, the base decision function can be expressed as (for an utterance $\mathbf{u}_j$):

$$\mathbf{h}_{enc,t}, \mathbf{c}_{enc,t} = f_{enc}\mathbf{h}_{enc,t-1}, \mathbf{c}_{enc,t-1}, \mathbf{e}\mathbf{u}_{jt}, \mathbf{\Phi}_{enc} \tag{2}$$

$$\mathbf{f} = softmax\mathbf{h}_{enc,|\mathbf{u}_j|}, \mathbf{\Phi}_{pred} \tag{3}$$

where $softmax \mathbf{v}_k = \frac{e^{\mathbf{y}_k}}{_j e^{\mathbf{y}_j}}$ returns a probability distribution, and $\mathbf{h}_{enc,t}$ and $\mathbf{c}_{enc,t}$ are the LSTM encoding and context of the sequence over time t.

We use two corpora: a large source corpus C_1 and a smaller target corpus C_2, which have separate vocabularies V_{C_1} and V_{C_2} that share some common words $V_{overlap}$. The source corpus is significantly larger than the target corpus, or $|C_2| \ll |C_1|$. We considered four decision functions, each building on the previous one, designed to minimize cross entropy loss over the target corpus.

Our first decision function is trained on a single corpus using only the architecture described above, and it can be directly expressed by equations (2) and (3). We use this simple decision function as our baseline method in Section 6.

Our second decision function uses the same architecture. However, it trains on both corpora: it is trained on the target corpus immediately after training on the source corpus. This can be expressed as conditioning the function for the target corpus $\mathbf{f}_2$ with the function learned for the source corpus $\mathbf{f}_1$. Since these architectures share all learnable parameters, we can express $\mathbf{f}_2$ as:

$$\mathbf{h}_{2,enc,t}, \mathbf{c}_{2,enc,t} = f_{2,enc} \mathbf{h}_{2,enc,t-1}, \mathbf{c}_{2,enc,t-1}, \mathbf{e}\mathbf{u}_{jt}, \mathbf{\Phi}_{2,enc} | \mathbf{\Phi}_{1,enc} \tag{4}$$

$$\mathbf{f}_2 = softmax \mathbf{h}_{2,enc,|\mathbf{u}_j|}, \mathbf{\Phi}_{2,pred} | \mathbf{\Phi}_{1,pred} \tag{5}$$

Our third decision function introduces the "dual-encoder" architecture. It is defined over the target corpus $\mathbf{f}_2$ conditioned upon the encoding function for the source corpus $\mathbf{h}_{1,enc,t}$. It uses two separate LSTM RNNs that do not share trainable parameters. In fact, the global encoder cannot be trained when the local encoder is trained on the target corpus. The initial state of the local encoder is the output state of the global encoder that encoded the same input sequence:

$$\mathbf{h}_{1,enc,t}, \mathbf{c}_{1,enc,t} = f_{1,enc} \mathbf{h}_{1,enc,t-1}, \mathbf{c}_{1,enc,t-1}, \mathbf{e}_1 \mathbf{u}_{jt}, \mathbf{\Phi}_{1,enc} \tag{6}$$

$$\mathbf{h}_{2,enc,t}, \mathbf{c}_{2,enc,t} = f_{2,enc} \mathbf{h}_{2,enc,t-1}, \mathbf{c}_{2,enc,t-1}, \mathbf{e}_2 \mathbf{u}_{jt}, \mathbf{\Phi}_{2,enc} | \mathbf{h}_{1,enc,|\mathbf{u}_j|} \tag{7}$$

$$\mathbf{f}_2 = softmax \mathbf{h}_{2,enc,|\mathbf{u}_j|}, \mathbf{\Phi}_{2,pred} \tag{8}$$

The decision function first encodes the utterance $\mathbf{u}_j$ using the global encoder $f_{1,enc}$, and then encodes $\mathbf{u}_j$ again using the local encoder conditioned on the global encoder.

The final decision function uses the complete architecture shown in Figure 1. The local and global encoder relationship is the same as described in the previous decision function; however, an additional attention mechanism is introduced between the encoder and predictor sections. This model collects all global and local encoder states for a sequence, sends them through the attention mechanism, and uses the flattened output of the attention mechanism as the input to the prediction softmax layer:

$$\mathbf{h}_{1,enc,t}, \mathbf{c}_{1,enc,t} = f_{1,enc} \mathbf{h}_{1,enc,t-1}, \mathbf{c}_{1,enc,t-1}, \mathbf{e}_1 \mathbf{u}_{jt}, \mathbf{\Phi}_{1,enc} \tag{9}$$

$$\mathbf{h}_{2,enc,t}, \mathbf{c}_{2,enc,t} = f_{2,enc} \mathbf{h}_{2,enc,t-1}, \mathbf{c}_{2,enc,t-1}, \mathbf{e}_2 \mathbf{u}_{jt}, \mathbf{\Phi}_{2,enc} | \mathbf{h}_{1,enc,|\mathbf{u}_j|} \tag{10}$$

$$\mathbf{w}_{attn} = \{softmax \mathbf{h}_{1,enc,t} \odot \mathbf{h}_{2,enc,t}\}_{t=0}^{|\mathbf{u}_j|} \tag{11}$$

$$\mathbf{c}_{attn} = \{\mathbf{w}_{attn,t} \odot \mathbf{h}_{1,enc,t}\}_{t=0}^{|\mathbf{u}_j|} \tag{12}$$

$$\mathbf{a}_{attn} = \mathbf{c}_{attn,1,1}, \ldots, \mathbf{c}_{attn,q,1}, \ldots, \mathbf{c}_{attn,1,q}, \ldots, \mathbf{c}_{attn,q,q} \tag{13}$$

$$\mathbf{f}_2 = softmax \mathbf{a}_{attn}, \mathbf{\Phi}_{2,pred} \tag{14}$$

We direct readers to the `dualencattendlstmwe.py` file in our online appendix (Section 8) for an implementation in Keras [14].

5 DATA PREPARATION

We used two datasets for this paper: 1) the AMI business meeting corpus [27], which served as the source dataset and 2) a corpus of Wizard of Oz conversations for software debugging [44], which served as the target dataset. We chose to use the debugging corpus because it is one of the few available datasets in the target domain. There were several suitable candidates for the source dataset, such as the TRAINS corpus [1], the Switchboard corpus [18], and the MapTask corpus [3]. We ultimately chose to use the AMI corpus because of its relatively large size, broad range of topics, thorough documentation, and prior use in the SE domain [34].

The AMI corpus was transcribed from over 100 hours of meeting recordings and contains over 100,000 utterances. The AMI corpus was both *segmented* and *annotated* by its authors. The authors provide a thorough guide, but in brief, the process involves human evaluators reading transcripts of the conversations and selecting a continuous sequence of words by one speaker (an utterance), and then labeling that sequence with one dialogue act type. The authors developed a set of 14 dialogue act type labels, which they used to label all utterances in the corpus. The dialogue act types are "generic" in that they are applicable to a wide variety of conversations.

The debugging corpus comprises 30 WoZ dialogues between human developers and simulated virtual agents, and it contains a total of 2243 written dialogue turns. Wood *et al.* annotated each message with a dialogue act types specific to the SE domain (e.g. "API Question").

The segmentation and annotation processes used to generate the source and target datasets must be identical for our transfer learning approach. To that end, we hired two human annotators to independently segment and annotate all 30 debugging dialogues using the AMI dialogue act types. Both annotators had at least two years of programming experience and followed the published AMI annotation guidelines. Because the annotators independently segmented the dialogues, they did not always end up annotating the same utterances (e.g., one annotator may have segmented a message into two utterances, while the other considered the entire message to be one utterance). On the utterances that they did both annotate, their annotation agreement (as measured by Cohen's kappa statistic) was 0.42, indicating moderate agreement. Additionally, because there were two annotators, we could not automatically resolve disagreements in segmentation or annotation (e.g., by taking a majority vote). Therefore, we include both annotators' segments and labels in the target dataset.

We observe that the composition of dialogue acts in the two datasets is quite different, as depicted in Figure 2. Some DA types are in roughly the same proportion, such as Inform and Assess, but others, such as Elicit-Inform, are far more common in the SE domain-specific dataset.

This observation is important because some dialogue act types are much more important for a VA to recognize than others (see Section 2 above). In particular, we identify the following five dialogue act types are the most significant ones for a SE/debugging VA to classify correctly:

- **Elicit-Inform**
  - *Example: "Is there an event listener on the frame class?"*
- **Elicit-Offer-Or-Suggestion**
  - *Example: "How should I start?"*
- **Comment-About-Understanding**
  - *Example: "I am confused"*
- **Elicit-Assessment**
  - *Example: "Is this incorrect?"*
- **Elicit-Comment-About-Understanding**
  - *Example: "do you get me?"*

These are the most important for our use case for two reasons: First, these cover the questions asked by programmers of a virtual agent. We do not care if a system can classify text generated by the agent itself, because the agent will already know what its own dialogue act type is. Second, mistakes made in classifying commentary as e.g. Be-Positive are easily recoverable by the VA in a real conversation, while mistakes in classifying questions can degrade trust in the system to answer those questions properly. Third, these dialogue act types closely overlap with the key dialogue act types uncovered by Wood *et al.* related to the questions programmers ask (such as Elicit-Inform and API Question).

6 EXPERIMENT

This section describes the experiment we use to evaluate our model, as well as relevant metrics and baseline methods.

6.1 Research Questions

Our research objective is to assess the performance of our model in comparison to competitive baselines. To that end, we ask the following Research Questions (RQs):

RQ$_1$ What is the baseline performance training and testing with AMI data?

RQ$_2$ What is the baseline performance training with AMI data and testing with the SE debugging dataset?

RQ$_3$ What is the baseline performance training and testing with the SE debugging dataset?

RQ$_4$ What is our model's performance training with AMI and SE data, and testing with the SE dataset?

The rationale behind RQ$_1$ is to establish a "reasonable expectation" for performance given state-of-the-art tools and a large dataset. A key goal of our model is to transfer knowledge learned on a large generic dataset to a small, domain-specific dataset, and one simple transfer approach is to just train on the large dataset. Another straightforward approach is to train only on the limited domain-specific data we have available. Our model will need to improve over these approaches to justify the added complexity of our model, so we ask RQ$_2$ and RQ$_3$ to establish baseline performance and RQ$_4$ to compare our model to that performance.

6.2 Methodology

Our methodology to answer RQ$_1$ is to randomly split the conversations from AMI into training/validation/test sets (80%, 10%, 10%), in order to prevent dialogue acts from the same conversations from appearing in both the training and test sets. We then train each baseline using the training set and report results from the test.

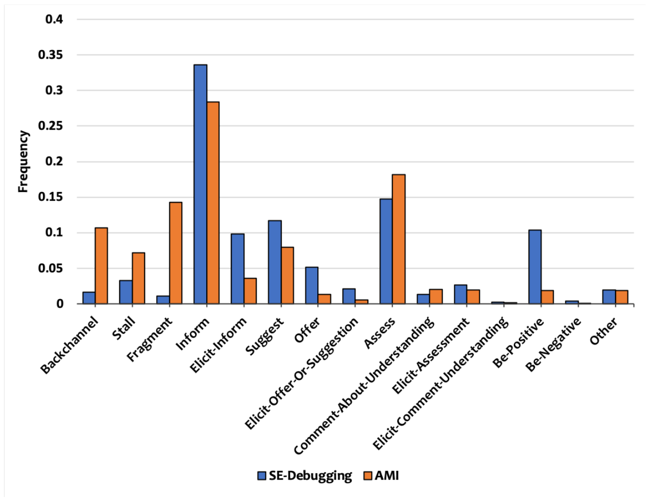

Figure 2: The composition of dialogue acts types using the AMI DA labels. Note that in our experiment we collapse the three smallest classes into a 15th "other" class due to their tiny size in the SE-Debugging data.

For RQ$_2$ and RQ$_4$, we combine the training and test sets of AMI and use both together to train the baseline model (for RQ$_2$) and the global encoder portion of our model (for RQ$_4$). This combination maximizes the amount of available data without compromising evaluation results (as RQ$_1$ is the only one that tests on the AMI data). We still hold the validation set aside for model selection.

For the debugging dataset, we were interested in evaluating the approaches on very small dataset sizes, so we created ten random subsets containing between 3 and 30 conversations (ranging from 10 to 100% of the dataset). For each subset, we created ten random training/val/test splits (70%, 10%, 20%), resulting in a total of 100 datasets based on the 30 conversations. We enforced a minimum of one conversation in validation and testing, so in some cases fewer than 70% of the conversations were in the training set (e.g. for the subset with three conversations, the training/val/test sets consisted of one conversation each).

Finally, for RQ$_2$, we evaluate the baseline by training on the AMI dataset and testing on each of the 100 test sets from the random splits. For RQ$_3$, we evaluate the baseline using the training and test sets from the 100 splits. And for RQ$_4$, we evaluate our model training on the AMI dataset and the training sets from the 100 splits, and testing on the test sets from the 100 splits. We report the average of the ten random splits for at each of the ten sizes of conversation (e.g., average results for all ten splits of 21 conversations).

6.3 Metrics

We use three standard metrics: precision, recall, and F1-score. We weight based on class size using the default `sklearn` [29] settings.

6.4 Baselines

The main baseline we use is based on related work by Khanpour *et al.* [22]: a word embedding and LSTM with the word sequence of the dialogue act as input. We provide an implementation of this baseline in file lstmwe.py in our online appendix (Section 8). One configuration of the baseline is to train with the AMI corpus and test with the SE debugging corpus (RQ$_2$), which represents a standard attempt at transfer learning. Another configuration is to train and test on only the SE debugging corpus (RQ$_3$). We keep parameters of the model e.g. word embedding dimensions and LSTM output size identical to the global encoder in our approach (Section 4), to minimize the number of experimental conditions.

As discussed in Section 3.1, there are many published approaches for DA classification. Several of the newest are not directly comparable because they are "post hoc" in that they are intended to classify an entire conversation at once, rather than each dialogue act as it occurs like a virtual agent would need to do. A consensus of the others is that a combination of word embedding space and recurrent or convolutional layers leads to good performance, leading to our choice of Khanpour's approach as a strong baseline to evaluate transfer learning strategies for our specific problem area.

In pilot studies, we verified a conclusion by Milajevs and Purver [28] that incorporating history data (i.e. the prior DA types in a conversation) does not improve performance when combined with text data, except in post hoc approaches. Therefore, we do not use history data in our baseline.

6.5 Threats to Validity

As with any study, our experiment carries threats to validity; many of these threats stem from the data preparation and processing. For instance, the dialogues in the chosen corpora are not necessarily representative of all "debugging" and "generic" dialogues, meaning that different target or source datasets may lead to different results. The "moderate" agreement between the annotators is another potential threat, and it implies that this is classification task is difficult even for humans. Furthermore, by including both annotators' segments and labels in the target dataset, we wind up with situations in which two different labels are associated with identical segments. This may artificially lower model performance during evaluation, as the models would not predict different labels for the same segment. Additionally, models trained on messy datasets (e.g. with incorrect or contradictory labels) tend to perform worse than those trained on clean datasets. However, the different models are all trained and tested on the same datasets and splits, meaning that we can still directly compare model performance.

Other threats include errors in extracting and parsing the word sequences, random factors such as a random dataset split in which the testing set turns out to be "easier" than average, and experimental parameters that we chose such as the LSTM output size. While we have taken steps to minimize these threats such as averaging results of ten splits instead of relying on one, it is possible that large changes in any of these threats could lead to different conclusions.

7 EXPERIMENTAL RESULTS

In this section we answer our Research Questions, providing our rationale and supporting data.

7.1 RQ$_1$: Baseline Expectations

The baseline performance when trained and tested on the AMI corpus is a 72.59% precision, 54.88% recall, and 58.90% F1 score, with

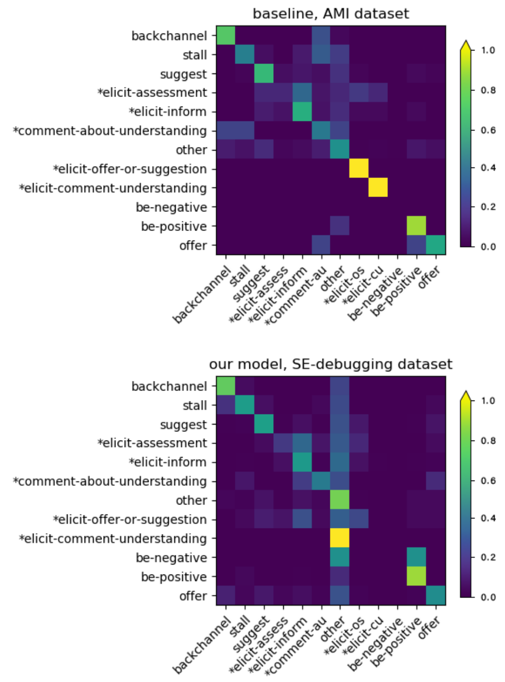

Figure 3: Confusion heatmap for RQ$_1$ (top) and RQ$_4$ (bottom). The row denotes the true label, and the column denotes the predicted label. Note that we have collapsed three tiny classes to "other" (see Figure 2).

most errors related to the "other" category, as shown in Figure 3. We view this as a ceiling on performance expectations for the SE dataset. Also, we note that these results were very similar for all dialogue act types and the five target types, unlike in the SE-debugging dataset.

7.2 RQ$_2$: Baseline Transfer Learning

In terms of all dialogue act types, the baseline transfer learning approach obtains much lower performance than the other approaches for all dataset sizes six conversations and above (Figure 4). However, the baseline transfer learning approach performs only slightly worse than the others when considering only the five target dialogue act types (Figure 5), meaning that its error rate is higher when classifying DA types of lower value to us, such as backchannel and stall.

7.3 RQ$_3$: Baseline Direct Training

The F1 score of the baseline direct training approach is similar to our approach for all dialogue act types, but does not exceed the baseline transfer learning for the five target types. Precision on the five target types (Figure 5c) begins to exceed our model for the two largest dataset sizes, which may imply diminishing returns for transfer learning at greater dataset sizes.

7.4 RQ$_4$: Our Model's Performance

The strongest area of performance for our model is in recall on the five target dialogue act types (Figure 5b), when recall exceeds the baselines for all dataset sizes six conversations or greater. As shown

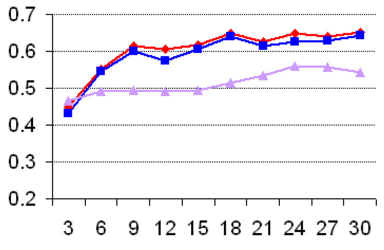

Figure 4: F1 scores for all dialogue act types.

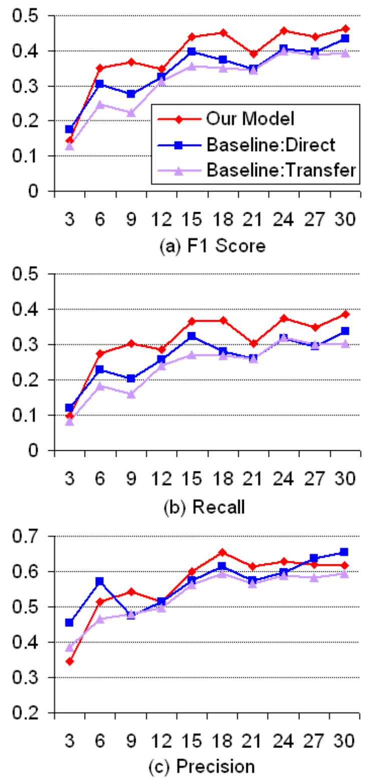

Figure 5: Performance metrics for the five most-important dialogue act types discussed in Section 5. X-axis is the number of SE debugging conversations.

in Figure 3, the approach does struggle with the type elicit-comment-about-understanding, probably due to the low incidence of that type. However, a majority of the mistakes for the target DA types are categorized as one of the other four target types, as opposed to the lesser-important ones e.g. backchannel.

An important caveat is that the F1 scores for all approaches are about 20% lower for the five target dialogue act types than for all types ($\sim$40% versus $\sim$60%). Nearly all papers on dialogue act classification report accuracy over all dialogue act types, but some types are more important than others, and some types are much "easier" to detect than others.

8 CONCLUSION AND REPRODUCIBILITY

We have presented a technique for transfer learning in dialogue act classification. For domain-specific applications such as building virtual agents for program debugging, the available data may be very small compared to the datasets available for open-topic conversations. However, a VA needs to be able to adequately identify the dialogue act types of user utterances in these specialized conversations in order to generate suitable responses. We find that our proposed model design can transfer knowledge from a large dialogue dataset to a smaller domain-specific one, and exceed baseline performance on classifying high value dialogue act types.

To promote reproducibility and assist future work, we release all data and source code in an online appendix:

https://tinyurl.com/y83v6v39

ACKNOWLEDGMENTS

This work is supported in part by NSF CCF-1452959 and CCF-1717607. Any opinions, findings, and conclusions expressed herein are the authors and do not necessarily reflect those of the sponsors.

REFERENCES

[1] James F Allen, Lenhart K Schubert, George Ferguson, Peter Heeman, Chung Hee Hwang, Tsuneaki Kato, Marc Light, Nathaniel Martin, Bradford Miller, Massimo Poesio, et al. 1995. The TRAINS project: A case study in building a conversational planning agent. *Journal of Experimental & Theoretical Artificial Intelligence* 7, 1 (1995), 7–48.
[2] Toine Andernach. [n.d.]. A Machine Learning Approach to the Classification of Dialogue Utterances. In *NeMLaP '96*.
[3] Anne H Anderson, Miles Bader, Ellen Gurman Bard, Elizabeth Boyle, Gwyneth Doherty, Simon Garrod, Stephen Isard, Jacqueline Kowtko, Jan McAllister, Jim Miller, et al. 1991. The HCRC map task corpus. *Language and speech* 34, 4 (1991), 351–366.
[4] Jeremy Ang, Yang Liu, and Elizabeth Shriberg. [n.d.]. Automatic dialog act segmentation and classification in multiparty meetings. In *ICASSP'05*.
[5] Kent Bach and Robert Harnish. 1979. Linguistic communication and speech acts. (1979).
[6] Srinivas Bangalore, Giuseppe Di Fabbrizio, and Amanda Stent. 2008. Learning the structure of task-driven human–human dialogs. *IEEE Transactions on Audio, Speech, and Language Processing* 16, 7 (2008), 1249–1259.
[7] Andrew Begel and Beth Simon. 2008. Struggles of new college graduates in their first software development job. In *ACM SIGCSE Bulletin*, Vol. 40. ACM, 226–230.
[8] Phil Blunsom, Nal Kalchbrenner, and Nal Kalchbrenner. [n.d.]. Recurrent convolutional neural networks for discourse compositionality. In *CVSC'13*.
[9] Kristy Elizabeth Boyer, Eun Young Ha, Robert Phillips, Michael D Wallis, Mladen A Vouk, and James C Lester. [n.d.]. Dialogue act modeling in a complex task-oriented domain. In *SIGDIAL'10*.
[10] Nicholas Bradley, Thomas Fritz, and Reid Holmes. [n.d.]. Context-Aware Conversational Developer Assistants. In *ICSE'18*.
[11] Susanne Burger, Karl Weilhammer, Florian Schiel, and Hans G Tillmann. 2000. Verbmobil data collection and annotation. In *Verbmobil: Foundations of speech-to-speech translation*. Springer, 537–549.
[12] Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. 2017. A survey on dialogue systems: Recent advances and new frontiers. *ACM SIGKDD Explorations Newsletter* 19, 2 (2017), 25–35.
[13] Zheqian Chen, Rongqin Yang, Zhou Zhao, Deng Cai, and Xiaofei He. [n.d.]. Dialogue Act Recognition via CRF-Attentive Structured Network. In *SIGIR'18*. ACM, 225–234.
[14] François Chollet et al. 2015. Keras. https://keras.io.
[15] Javier Escobar-Avila, Esteban Parra, and Sonia Haiduc. [n.d.]. Text Retrieval-based Tagging of Software Engineering Video Tutorials. In *ICSE-C '17*.
[16] Rashmi Gangadharaiah, Balakrishnan Narayanaswamy, and Charles Elkan. [n.d.]. What we need to learn if we want to do and not just talk. In *NAACL-HLT'18*.
[17] Jeroen Geertzen, Volha Petukhova, and Harry Bunt. [n.d.]. A multidimensional approach to utterance segmentation and dialogue act classification. In *SIGDIAL'07 Workshop on Discourse and Dialogue*.
[18] John J Godfrey, Edward C Holliman, and Jane McDaniel. [n.d.]. SWITCHBOARD: Telephone speech corpus for research and development. In *ICASSP'92*.

[19] Sergio Grau, Emilio Sanchis, Maria Jose Castro, and David Vilar. [n.d.]. Dialogue act classification using a Bayesian approach. In *SPECOM'04*.

[20] Yangfeng Ji, Gholamreza Haffari, and Jacob Eisenstein. [n.d.]. A latent variable recurrent neural network for discourse relation language models. *NAACL-HLT'16* ([n. d.]).

[21] Yiping Kang, Yunqi Zhang, Jonathan K Kummerfeld, Lingjia Tang, and Jason Mars. [n.d.]. Data Collection for Dialogue System: A Startup Perspective. In *NAACL-HLT'18*.

[22] Hamed Khanpour, Nishitha Guntakandla, and Rodney Nielsen. [n.d.]. Dialogue act classification in domain-independent conversations using a deep recurrent neural network. In *COLING'16*.

[23] Andrew J. Ko and Brad A. Myers. 2010. Extracting and Answering Why and Why Not Questions About Java Program Output. *ACM Trans. Softw. Eng. Methodol.* 20, 2, Article 4 (2010), 36 pages. https://doi.org/10.1145/1824760.1824761

[24] Harshit Kumar, Arvind Agarwal, Riddhiman Dasgupta, Sachindra Joshi, and Arun Kumar. [n.d.]. Dialogue Act Sequence Labeling using Hierarchical encoder with CRF. ([n. d.]).

[25] Ji Young Lee and Franck Dernoncourt. [n.d.]. Sequential short-text classification with recurrent and convolutional neural networks. *NAACL-HLT'16* ([n. d.]).

[26] Yang Liu, Kun Han, Zhao Tan, and Yun Lei. [n.d.]. Using Context Information for Dialog Act Classification in DNN Framework. In *EMNLP'17*.

[27] Iain McCowan, Jean Carletta, W Kraaij, S Ashby, S Bourban, M Flynn, M Guillemot, T Hain, J Kadlec, V Karaiskos, et al. 2005. The AMI meeting corpus. In *Proceedings of the 5th International Conference on Methods and Techniques in Behavioral Research*, Vol. 88.

[28] Dmitrijs Milajevs and Matthew Purver. [n.d.]. Investigating the contribution of distributional semantic information for dialogue act classification. In *CVSC'14*.

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[30] Piotr Pruski, Sugandha Lohar, William Goss, Alexander Rasin, and Jane Cleland-Huang. 2015. TiQi: answering unstructured natural language trace queries. *Requirements Engineering* 20, 3 (01 Sep 2015), 215–232. https://doi.org/10.1007/s00766-015-0224-4

[31] Norbert Reithinger and Martin Klesen. [n.d.]. Dialogue act classification using language models. In *EUROSPEECH'97*.

[32] Verena Rieser and Oliver Lemon. 2011. *Reinforcement learning for adaptive dialogue systems: a data-driven methodology for dialogue management and natural language generation*. Springer Science & Business Media.

[33] Martin P Robillard, Andrian Marcus, Christoph Treude, Gabriele Bavota, Oscar Chaparro, Neil Ernst, Marco Aurélio Gerosa, Michael Godfrey, Michele Lanza, Mario Linares-Vásquez, et al. [n.d.]. On-Demand Developer Documentation. In *ICSME'17*.

[34] Paige Rodeghero, Siyuan Jiang, Ameer Armaly, and Collin McMillan. [n.d.]. Detecting user story information in developer-client conversations to generate extractive summaries. In *ICSE'17*.

[35] Tobias Roehm, Rebecca Tiarks, Rainer Koschke, and Walid Maalej. [n.d.]. How do professional developers comprehend software?. In *ICSE'12*.

[36] Ruhi Sarikaya, Paul A Crook, Alex Marin, Minwoo Jeong, Jean-Philippe Robichaud, Asli Celikyilmaz, Young-Bum Kim, Alexandre Rochette, Omar Zia Khan, Xiaohu Liu, et al. [n.d.]. An overview of end-to-end language understanding and dialog management for personal digital assistants. In *SLT'16*.

[37] John Searle. 1965. *What is a speech act?* Cambridge University Press.

[38] Riccardo Serafin, Barbara Di Eugenio, and Michael Glass. [n.d.]. Latent Semantic Analysis for dialogue act classification. In *NAACL-HLT'03*.

[39] Iulian Vlad Serban, Ryan Lowe, Peter Henderson, Laurent Charlin, and Joelle Pineau. 2018. A Survey of Available Corpora For Building Data-Driven Dialogue Systems. *Dialogue & Discourse* 9, 1 (2018), 1–49.

[40] Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics* 26, 3 (2000), 339–373.

[41] Dinoj Surendran and Gina-Anne Levow. [n.d.]. Dialog act tagging with support vector machines and hidden Markov models. In *ICSLP'06*.

[42] Maryam Tavafi, Yashar Mehdad, Shafiq Joty, Giuseppe Carenini, and Raymond Ng. [n.d.]. Dialogue act recognition in synchronous and asynchronous conversations. In *SIGDIAL'13*.

[43] Enn Tyugu. 1988. *Knowledge-Based Programming (Turing Institute Press Knowledge Engineering Tutorial Series)*. Addison-Wesley Longman Publishing Co., Inc.

[44] Andrew Wood, Paige Rodeghero, Ameer Armaly, and Collin McMillan. [n.d.]. Detecting Speech Act Types in Developer Question/Answer Conversations During Bug Repair. *FSE'18* ([n. d.]).

[45] Matthias Zimmermann. [n.d.]. Joint segmentation and classification of dialog acts using conditional random fields. In *ISCA'09*.