Henrik Haugland Syverinsen

Supervised Pre-training for Dialogue Act Classification in Task-oriented Dialogue

Master's thesis in Computer Science
Supervisor: Krisztian Balog
June 2021

Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Henrik Haugland Syverinsen

Supervised Pre-training for Dialogue Act Classification in Task-oriented Dialogue

Master's thesis in Computer Science
Supervisor: Krisztian Balog
June 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Abstract

In recent years dialogue systems, where a user can converse with an agent in natural language, have become ubiquitous through smartphones, smart speakers, and customer service. Dialogue act classification, the task of detecting the function of an utterance, is an important part of the natural language understanding module in task-oriented dialogue systems, where the system helps a user perform a task. Executing the task of dialogue act classification well is critical for the performance of the task-oriented dialogue system.

State-of-the-art approaches often fine-tune language models that have been pre-trained in a semi-supervised fashion with large amounts of unlabeled data. This pre-training gives a general language understanding, but usually leaves task-specific features to be learned during fine-tuning, which may be difficult if there is little data available. We propose performing task-specific supervised pre-training with several unified datasets to learn task-specific features before performing dataset-specific fine-tuning. We unify the label sets of task-oriented dialogue act datasets to a universal schema, implement pre-training and fine-tuning architectures, and make experimental comparisons. We find that models pre-trained in a supervised fashion often lead to better performance than the same non pre-trained models. The largest improvements are found when limited data is available for fine-tuning, indicating that this is a viable approach for dialogue act classification when there is limited data available.

Sammendrag

I de siste årene har dialogsystemer, hvor en bruker kan lede en samtale med en agent i naturlig språk, blitt allestedsnærværende gjennom smarttelefoner, smarte høytalere, og kundeservice. *Dialogue act* klassifisering, som omhandler å gjenkjenne funksjonen en ytring formidler, er en viktig del av modulen som prøver å forstå naturlig språk i oppgaveorienterte dialogsystemer, hvor systemet hjelper en bruker å utføre en oppgave. God utførelse av *dialogue act* klassifisering er kritisk for ytelsen til det oppgaveorienterte dialogsystemet.

Moderne metoder finjusterer ofte språkmodeller som er pre-trent på en semi-veiledet måte med store mengder data som ikke har tilhørende utgangsverdier. Denne pre-treningen oppnår en generell språkforståelse, men lar ofte oppgavespesifikke trekk være for å bli lært under finjustering, som kan være vanskelig med begrenset tilgang på data. Vi foreslår å utføre oppgavespesifikk veiledet pre-trening med flere forente datasett for å lære oppgavespesifikke trekk før vi gjør datasettspesifikk finjustering. Vi forener settene av utgangsverdier til oppgaveorienterte *dialogue act* datasett slik at de følger et universalt skjema. Videre implementerer vi pre-trening- og finjusterings-arkitekturer og gjør eksperimentelle sammenligninger. Resultater viser at modeller pre-trent på en veiledet måte ofte leder til bedre ytelse enn det ikke pre-trente modeller kan oppnå. De største forbedringene i ytelse finner vi når mengden treningsdata for finjustering er begrenset, som indikerer at metoden vår er effektiv for *dialogue act* klassifisering når mengden treningsdata tilgjengelig er begrenset.

Preface

The present master's thesis is written during the spring of 2021 at the Norwegian University of Science and Technology (NTNU). The thesis is a finalisation of a 5-year master's degree in Computer Science at the Department of Computer Science, Faculty of Information Technology and Electrical Engineering at NTNU. Supervision has been provided by Professor Krisztian Balog.

Contents

Chapter 1

Introduction

This chapter gives an introduction to the thesis through its motivation, the research questions that will be answered, its main contributions, and an outline of the rest of the thesis.

1.1 Motivation

In recent years deep learning methods have become popular in training dialogue systems and their components. Deep learning methods require large amounts of labeled training data to be effective. However, existing labeled dialogue datasets are often small in size since collection and human annotation is expensive and time-consuming. Previous work has experimented and succeeded with self-supervised pre-training on abundant unlabeled data in attempts to alleviate the data scarcity problem [8, 17, 24, 12, 39].

Dialogue act classification (DAC) is the task of detecting the function of an utterance in a dialogue interaction (e.g., requesting information or welcoming a user). For the DAC task in task-oriented dialogue there exists several labeled datasets. The datasets are labeled with different annotation schemas that may have many labels in common. Previous pre-training approaches, like BERT [8], typically pre-train on unlabeled data before they fine-tune on a single labeled target dataset. We want to take advantage of datasets with overlapping annotation schemas by unifying them, and by performing further task-specific supervised pre-training with multiple datasets, followed by dataset-specific fine-tuning. With this approach, we hope to learn task-specific representations beneficial for DAC from multiple datasets, and transfer this to a target dataset. Architectures will also be developed to facilitate this transfer learning. This approach will hopefully make training with little labeled data more efficient. We also want to investigate how the pre-trained model transfers to datasets with annotation schemas of varying similarity to the unified schema used during pre-training.

This thesis builds on the work of Paul et al. [27] where they train a universal dialogue act tagger (U-DAT) on two human-machine datasets after unifying their annotation schemas. U-DAT predicts the next dialogue act of the system, given a user utterance and dialogue history. We want to take this idea further by unifying a larger corpus for pre-training and aim to apply it on classification of user utterances. Classification will be done on both the unified schema and on the original schemas of the target datasets.

1.2 Research Questions

With the motivation from the previous section we define the following research questions:

- **RQ1** How does the performance of a further pre-trained BERT model compare to a BERT model that's only fine-tuned?
  - **RQ1.1** How does further pre-training affect performance when training and evaluating on the universal and original dialogue act schema?
  - **RQ1.2** How does further pre-training affect performance when there is limited training data available?
  - **RQ1.3** How do different architectures for fine-tuning a further pre-trained model to datasets with varying annotation schemas affect performance?
  - **RQ1.4** How does further pre-training affect performance on datasets with different dialogue act schemas?

These research questions will be answered by first aligning multiple task-oriented datasets labeled with dialogue acts to the universal schema proposed by Paul et al.

[27]. The resulting corpus, named TODA, is then used to perform further pre-training in a supervised fashion on a BERT model. We name the further pre-trained model TODA-BERT. Results of fine-tuning TODA-BERT will be compared to results of fine-tuning of a basic BERT model.

- **RQ2** How does the performance of a pre-trained dialogue context inclusive model compare to a model that's only fine-tuned?
  - **RQ2.1** How does further pre-training affect performance when training and evaluating on the universal and original dialogue act schema?
  - **RQ2.2** How does further pre-training affect performance when there is limited training data available?

These research questions will be answered by using the TODA corpus to perform supervised pre-training on a U-DAT model. We name the pre-trained model TODA-UDAT. Results of fine-tuning TODA-UDAT will be compared to fine-tuning of a non pre-trained U-DAT model.

1.3 Main Contributions

The following points are the main contributions this thesis makes to the research area of pre-training and dialogue act classification:

- Alignment of several datasets to a universal dialogue act schema.
- Pre-training and fine-tuning architectures.
- Implementation of architectures, as well as a pre-trained model, made publicly available[1]
- Experimental comparison and insights.

1.4 Outline

Chapter 2 introduces theory necessary to understand the classification problem, algorithms, and evaluation metrics used in the thesis.

[1]`https://github.com/hsyver/TODA-BERT`

Chapter 3 presents related work.

Chapter 4 describes the alignment of datasets to a universal schema and architectures used in experiments.

Chapter 5 presents and discusses the results of the experiments.

Chapter 6 concludes the thesis and presents further work.

Chapter 2

Preliminaries

This chapter presents dialogue systems, the classification problem, algorithms, and evaluation metrics used in this thesis to give a better understanding of the following chapters.

2.1 Dialogue Systems

A dialogue system involves a human user and a system agent which converse in natural language. The literature divides dialogue systems into three categories: task-oriented, question-answering, and social chatbots [7].

A task-oriented dialogue system helps a user solve a task. Tasks may range from booking movie tickets to setting reminders or getting the weather forecast. Task-oriented dialogue systems usually steer the dialogue to follow a structure which ensures that the constraints of the user are satisfied (i.e., number of tickets to book, theatre location, and time slot). These constraints are gathered by the system in as few turns as possible.

Question-answering dialogue systems try to answer the questions of the user. The structure of dialogues with question-answering dialogue systems is centered around questions, answers, and follow-up questions for clarification.

Social chatbots attempt to replicate social interaction and make chitchat conversation with the user without any underlying task to solve.

Dialogue systems typically include the four following modules: (1) a natural language understanding (NLU) module which extracts intent and other information from a user utterance, (2) a dialogue state tracker that keeps information of the conversation, (3) a decision making module that decides the action of the system based on the current state of the conversation, and (4) a natural language generation (NLG) module which takes the action of the decision making module and converts it to natural language to be returned to the user [11].

NLU is often broken down into three tasks: identification of domain, intent or dialogue act classification, and extraction of slots. In an example utterance like "I'm looking for a moderately priced restaurant in the east part of town.", the domain would be *restaurant*, the detected dialogue act would be *inform*, and slot-value pairs would be *pricerange=moderate* and *area=east*. The two first tasks are classification problems, while slot filling is a sequence labelling task, where each word of an utterance is classified.

2.2 Classification

Classification is the task of identifying which class or classes a sample belongs to. An example classification task can be to detect whether an email is spam or not spam. An algorithm that implements classification is called a *classifier*. The classifier aims to learn a function f given training data x with accompanying classes y such that $y = f(x)$. The performance of the classifier is measured by classifying unseen data.

2.2.1 Dialogue Act Classification

Dialogue act classification is the task of detecting the dialogue acts which describe the function of an utterance in a dialogue interaction. The function of an utterance can for example be to inform about preferences, request information, or request alternatives. A dialogue D of N utterances, u_i, is denoted as $D = u_1, u_2, ..., u_N$. Let A be a pre-defined set of M dialogue acts, i.e., $A = a_1, a_2, ..., a_M$. Given a user utterance u_i and possibly the dialogue history, DAC aims to detect the set of dialogue acts $A_i \subset A$ that belong to u_i.

2.3 Algorithms

This section presents learning algorithms used in this thesis.

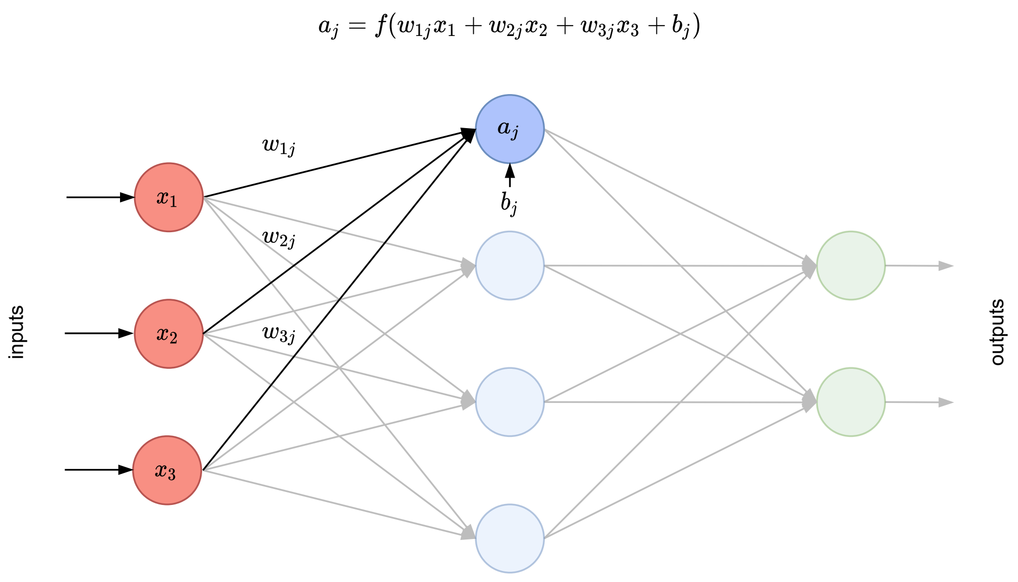


Figure 2.1: Feed forward neural network with an example computation. The function f is the activation function.

2.3.1 Feed Forward Neural Networks

Feed forward neural networks are learning algorithms made up of layers of connected nodes. An input layer takes one value for each node in the layer and an output layer gives one value for each node in the layer. Between the input and output layer, there can be any number of hidden layers. Nodes are connected between layers and values are fed forward through the network. Values are updated as they pass through the connections of the network. A connection between two nodes i and j has a weight w_{ij} which the value is multiplied with. Every node in a hidden or output layer has a bias value which is added to the incoming value. When multiple connections lead to a node, the incoming values are summed before the bias is added. Before the value is passed to the next layer, an activation function is applied. An example of an activation function is the ReLU, which limits values to positive values. The neural network learns as weights and biases are updated when the difference between a prediction and target, expressed by a loss function, is backpropagated. Figure 2.1 shows an example of a feed forward neural network with three input nodes, one hidden layer, and an output layer with two nodes.

2.3.2 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM), introduced by Hochreiter and Schmidhuber [18], is a recurrent neural network (RNN) architecture for processing sequences of data. RNNs contain cycles where activations from previous time steps are used as input for the current time step. The LSTM architecture was designed to address the problems of conventional RNNs: vanishing or exploding gradients when updating weights through backpropagation and limited ability to model long range dependencies.

A standard LSTM architecture includes an input layer, a recurrent LSTM layer and an output layer. At the core of the LSTM layer lies the cell state. The LSTM updates the cell state through gates made up of a sigmoid function and a multiplication operation which optionally lets information through. The LSTM network maps the input sequence $(x_1, ..., x_T)$ to an output sequence $(y_1, ..., y_T)$ by calculating activations in the LSTM layer from time step $t = 1$ to T according to the following equations:

$$\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\
g_t &= tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot tanh(c_t)
\end{aligned}$$

where h is the hidden state, c is the cell state, x is the input, and i, f, g, o are the input, forget, cell, and output gates, respectively. The W terms represent weight matrices and b terms represent bias vectors. σ is the sigmoid function, and $\odot$ is the element-wise product of vectors.

A bidirectional LSTM is an LSTM that operates on the input sequence in both directions when making a decision for the current input to learn dependencies in both directions of the sequence.

2.3.3 Transformers

The transformer architecture, proposed by Vaswani et al. [36], is a state-of-the-art architecture for processing sequences. Transformers differ from sequence processing

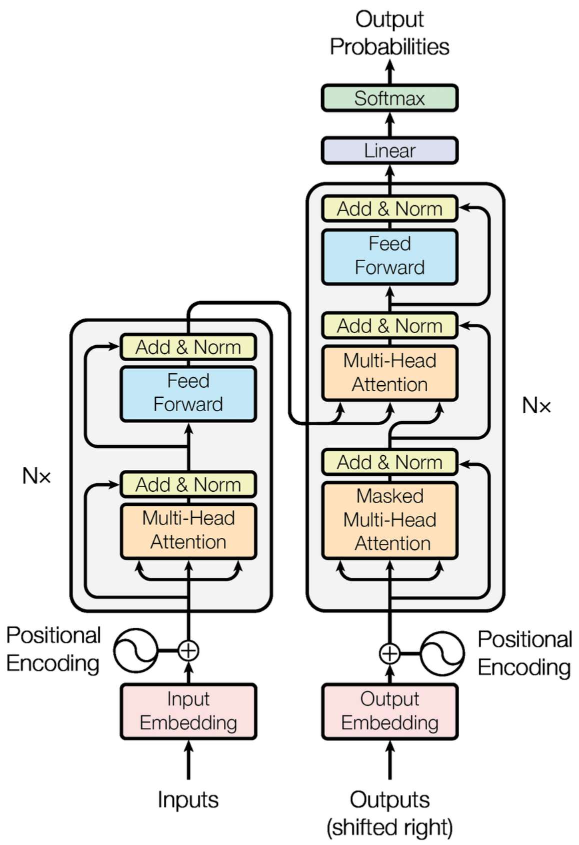


Figure 2.2: Original Transformer model architecture [36].

architectures like RNNs in that entire sequences are perceived at once, relying on an attention mechanism, rather than perceiving one element of the sequence at a time. Perception of the entire sequence at once allows for more parallelization in training, which is a fundamental constraint of RNNs [36].

The original Transformer, proposed by Vaswani et al., employs an encoder-decoder structure where an input sequence is mapped by the encoder. The decoder uses the mapped representation of the input to generate an output sequence one symbol at a time. The architecture of the Transformer is illustrated in Figure 2.2.

Each symbol of a sequence is transformed to an embedding $(x_1, ..., x_n)$. This embedding, along with a positional embedding, forms the input to the encoder stack. Six identical encoder layers make up the encoder stack.

At the core of an encoder layer lies an attention mechanism. In the first step of the attention mechanism, three matrices are calculated: the query matrix (Q), the key matrix (K), and the value matrix (V). The embeddings $(x_1, ..., x_n)$ are packed to the matrix X which gives Q, K, and V when multiplied with the trainable weight matrices W^Q, W^K, and W^V. The attention mechanism then computes Z according to Eq. (2.1), where d_k is the dimension of queries and keys. Z represents which symbols are relevant to each symbol in the sequence. Eight attention mechanisms in parallel forms the multi-head attention. The reason multiple attention mechanisms are used is such that multiple relationships between symbols can be learned. Outputs of the attention mechanisms are concatenated and multiplied with the trainable weight matrix W^O. The result is then passed through a feed forward neural network.

$$Z = \text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2.1}$$

Residual connections and layer normalization is applied around the multi-head attention and the feed forward neural network. The residual connection simply adds the input of the multi-head attention to the component's output. The output of the feed forward neural network, passed through normalization, is the output of a single encoder layer. This becomes the input of the next encoder layer in the encoder stack.

The decoder stack also contains six identical layers. Unlike the encoder layers, the decoder layers have two multi-head attention components per layer. The second multi-head attention gets the matrices K and V from the output of the encoder stack, while the matrix Q comes from the first multi-head attention in the layer.

The decoder generates the output sequence one symbol at a time. The input to the decoder is the output of the entire transformer from the previous time step. The output of the decoder is passed through a feed forward layer which gives an output vector in the size of the vocabulary. A softmax layer is applied to give probabilities for each symbol in the vocabulary.

2.3.4 BERT

BERT (Bidirectional Encoder Representations from Transformers) [8] is a method for pre-training language representations. A language understanding model is pre-trained from a large unlabeled text corpus through self-supervised learning. BERT uses English Wikipedia and BooksCorpus for pre-training. The pre-trained model can be applied on various natural language processing (NLP) tasks through fine-tuning, and Devlin et al. [8] achieve state of the art performance on 11 NLP tasks with this approach.

The architecture of a BERT model is nearly identical to the encoder stack of the Transformer proposed by Vaswani et al. The biggest difference is that BERT architectures have more encoder layers in the encoder stack and more attention mechanisms in parallel in the multi-head attention.

A BERT model is trained on two self-supervised tasks: *masked language modeling* (MLM) and *next sentence prediction* (NSP). MLM masks words in a sentence which are predicted to train the model. NSP draws sentence pairs from the corpora that have a 50% chance of being adjacent and the model is tasked with predicting whether the second sentence is the following sentence of the first or not.

Pre-training a BERT model is computationally expensive, which is why Devlin et al. have released BERT models of various sizes, $\text{BERT}_{\text{BASE}}$ being the most frequently used. The models can be adapted to various NLP tasks through inexpensive fine-tuning.

2.4 Evaluation Metrics

This section describes how models for multi-label dialogue act classification are evaluated in this thesis.

2.4.1 F1-score

The F1-score (2.2) is a function of precision (2.3) and recall (2.4) giving a score between 0 and 1.

$$\text{F1-score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.2}$$

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{2.3}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{2.4}$$

In the multi-label case, the F1-score can be computed in different ways based on the type of averaging. The micro-F1 score is computed by simply counting true positive, false positive, and false negative classifications globally. The macro-F1 score is calculated by taking the unweighted average of F1-scores for each class.

Chapter 3

Related Work

3.1 Natural Language Understanding Tools

This section describes some of the available platforms and toolkits for development and research in dialogue systems. We focus on functionality relevant for dialogue act classification or intent classification in the description of the tools.

3.1.1 Platforms

Several companies provide natural language understanding platforms to make it easier for developers to create conversational agents or incorporate natural language understanding in their apps. Some of the most popular platforms include:

- Wit.ai[1]
- Google's Dialogflow[2]
- Microsoft's LUIS[3]
- Amazon Lex[4]
- IBM's Watson Assistant[5]

[1]`https://wit.ai`
[2]`https://cloud.google.com/dialogflow`
[3]`https://luis.ai`
[4]`https://aws.amazon.com/lex`
[5]`https://ibm.com/cloud/watson-assistant`

- RASA[6]

An overview of main features related to dialogue act and intent classification in the platforms can be found in Table 3.1.

The platforms are centered around intents and entities. Developers can train the NLU module by defining intents and entities where they supply a small number of annotated example utterances. The cloud-based platforms (all except for RASA, which is open-source) have the advantage of being able to leverage data supplied by other users to expand upon the example utterances for training. In LUIS and RASA it's possible to define patterns or regular expressions to detect intents for improved performance. Most of the platforms provide pre-built intents. The pre-built intents are often domain-specific (e.g., *ChangeReservation*), but some of the platforms also provide general, dialogue act-like, pre-built intents like *Confirm*, *StartOver* or *Repeat* in LUIS.

Wit.ai and LUIS only perform intent classification and recognize entities. In the other platforms, responses and dialogue flows can be designed to create a complete conversational agent. RASA also has an experimental end-to-end training feature which only needs *stories* to train a conversational agent. This feature can also be combined with the traditional intent and entity detection.

Xingkun Liu and Rieser [40] state that a significant limitation of the platforms is that none of them use dialogue context for intent classification. This is problematic when a user speaks in fragment utterances that can only be correctly classified with knowledge of the previous utterances.

The platforms differ in how they handle multiple intents per input, if at all. Utterances which express more than one intent is common in spoken dialogue. An example is the utterance "Sounds good, what's their adress?", which expresses the intents *affirm* and *request.* Watson Assistant supports this, while Wit.ai supports it with the use of *traits* in combination with intents. LUIS and Amazon Lex facilitate multi-intent classification by returning scores for all the intents. RASA can detect mutliple intents if a specific classifier is used, but only the combinations of intents which are present in the training data. This is so that it should not be overly complicated to design dialogue flows, which could be difficult with arbitrary combinations of intents.

[6]`https://rasa.com`

Table 3.1: Features related to intent classification in NLU platforms. *Full CA* indicates whether the platform provides a full conversational agent or just a NLU service.

Platform	**Full CA**	**Multi-Intent Support**	**Patterns/ RegEx**	**Pre-Built Intents**	**Cloud-Based**	**Configurability**
Wit.ai		✓		✓	✓	
Dialogflow	✓			✓	✓	
LUIS		✓	✓	✓	✓	
Lex	✓	✓		✓	✓	
Watson	✓	✓		✓	✓	
RASA	✓	✓	✓			✓

The cloud-based platforms are secretive about what machine learning algorithms are used and what training data is used for pre-training. Dialogflow reveals that they use BERT-based language understanding models, while others only say they use deep learning algorithms. On the other hand there is RASA, the open-source alternative, which lets one create their own pipeline with different options for language models, intent classifiers, and more. RASA offers SVM, keyword-matching, and transformer-based intent classifiers.

3.1.2 Toolkits

This section presents toolkits that support development of dialogue systems or parts of dialogue systems. We focus on toolkits that have a modular design built around common dialogue system architecture, specifically including natural language understanding. Recent toolkits with active support are included. The toolkits are usually made for research and prototyping purposes as opposed to the platforms presented in Sect. 3.1.1 which are meant to create conversational agents for production.

Plato [26] is a flexible toolkit that supports any type of conversational agent architecture and aims to bridge the gap between state-of-the-art research and production. The toolkit abstracts away implementation details around training and evaluation to speed up the development process, offers common ground for testing of new ideas, and makes it easy to understand for people with different levels of expertise. It supports joint learning of modules, end-to-end learning, and is agnostic to the underlying learning frameworks. The modular design of Plato allows modules to run sequentially, in parallel, or in any combination of the two. A module can be a language understanding module, language generation module, dialogue state tracker, or any module of a conversational agent.

NeMo [20] is a toolkit for creating AI applications with neural modules and is developed around the principles of re-usability, abstraction, and composition of modules. A neural module is a piece of a neural network such as a language model, an encoder, a loss function, or other layers and functions. NeMo aims to separate the concerns of architecture definition, training procedure, analysis, and more which are often mixed in a single Python script. NeMo provides pre-built *collections* for automatic speech recognition and natural language processing, but users can easily create new collections. The *nemo_nlp* collection can be used for language modeling, sentence classification, and more. It also supports BERT pre-training and fine-tuning. NeMo is also a framework-agnostic toolkit.

ConvLab-2 [42] is a toolkit made for researchers so they can build task-oriented dialogue systems with state-of-the-art models and easily perform evaluation. In ConvLab-2, dialogue systems can be trained through different configurations of components or fully end-to-end. The toolkit supports most standard dialogue system components except for automatic speech recognition and text to speech synthesis. State-of-the-art models for the various components are already implemented, but researchers can add their own models by implementing the interfaces of the components.

PyDial [35] is a toolkit targeted at statistical dialogue systems. It aims to stimulate research and make it easier for people to get involved in the field. PyDial offers easy configuration and extension of the dialogue system modules. The toolkit supports multi-domain dialogues where a single conversation may span multiple topics.

3.2 Dialogue Acts & Domain-specific Intents

An intent in a task-oriented dialogue system is a label that captures the meaning of an utterance through the intention the user expresses in said utterance. Intents can be represented at different levels of granularity. The two main approaches for intent annotation in literature are dialogue acts and domain-specific intents.

A dialogue act in task-oriented dialogue captures the general intention or speech act behind an utterance independent of domain or dialogue system. Common dialogue acts include *inform*, *request*, *confirm*, *deny*, *request_alts*, and more. Some dialogue acts include slot-value pairs that gives information of entities in the utterance. An example is *inform(food="Italian")* which could belong to an utterance where a user is looking for Italian restaurants. Despite dialogue acts not being bound to a single domain or dialogue system, dialogue datasets apply different dialogue act schemas for annotation, complicating training with multiple datasets. Efforts have been made to create a universal dialogue act schema [41, 27]. Paul et al. [27] propose a universal dialogue act schema for task-oriented dialogues and align three existing datasets (DSTC2 [15], MultiWOZ2.0 [3], and M2M [33]) with their schema. The schema and alignment process is presented in detail in Sect. 4.1.1

The other main interpretation of an intent in task-oriented dialogue are domain-specific intents, which often describe the task the user wants to perform with their utterance. The domain-specific intents are usually very fine-grained, with one intent corresponding to one task an agent can perform (e.g., BookFlight or GetWeather). This gives agent-dependent intent "schemas" that may be difficult to align. Domain-specific intents are frequently used in the commercial platforms presented in Sect. 3.1.1.

3.3 Datasets

This section presents datasets we find relevant for dialogue act classification and intent classification. Dialogue act datasets are described more in depth than the datasets with domain-specific intents since focus is on dialogue acts in this thesis.

3.3.1 Task-oriented Dialogue Act Datasets

The following criteria were set to consider inclusion of a dataset:

- Dialogues must be task-oriented.
- Utterances must be labeled with dialogue acts.
- Dialogue must be multi-turn.
- Dialogues must involve a human (i.e., human-human or human-machine) or be paraphrased by a human, giving natural utterances.
- The size of the dataset should not be too small (preferably more than 1000 dialogues).

A summary with key properties of the datasets can be found in Table 3.2. The following paragraphs give an overview of the datasets collected.

DSTC 2 [15] and DSTC 3 [16] were released during the Dialog State Tracking Challenges 2 & 3. Combined the datasets are made up of 5,510 dialogues related to restaurant search. The dialogues were collected using various telephone-based dialogue systems which crowdworkers on Amazon Mechanical Turk[7] called. Transcription from audio to text was also performed by crowdworkers. The dialogues were labeled with dialogue acts by a semantic decoder which the authors corrected by hand. These two datasets will be treated as one in this thesis. The test set of DSTC 2 will be used for evaluation.

Frames [9] was proposed to study complex dialogue flows and decision-making behaviour. The dataset has 1,369 dialogues in the travel domain. Users were instructed to find a vacation package given some constraints in a search-and-compare process. The dialogues were collected in a Wizard-of-Oz (WOz) setting where a human is paired up with another human, or wizard, which takes the role of the dialogue system. The advantage of this setting is that the dialogues can display realistic behaviour not found in existing dialogue systems. The dialogues were written by only

[7]https://www.mturk.com/

Table 3.2: Statistics for task-oriented datasets

Dataset	#Dialogues	#Utterances	#Domains	Type
DSTC 2 & 3 [15, 16]	5,510	88,650	1	Spoken
Frames [9]	1,369	19,986	1	Written
MultiWOZ 2.3 [13]	10,438	143,048	7	Written
E2E [22]	10,087	74,686	3	Written
M2M [33]	3,008	27,120	2	Generated
SGD [30]	22,825	463,284	20	Generated
Total	53,237	816,774	24 unique	

12 participants. El Asri et al. [9] state that the advantage of few participants is that they know how to use the system, so they can focus on decision making and skip learning about system capabilities. Dialogues were annotated with dialogue acts by human experts. 10 random dialogues were selected to measure inter-annotator agreement on dialogue acts, which received an F1 score of 81.2 ± 3.1.

MultiWOZ 2.3 [13] is an updated version of the previous MultiWOZ datasets (2.0-2.2) with annotation corrections leading to significant improvements in natural language understanding and dialogue state tracking according to Han et al. [13]. The dataset contains 10,438 dialogues first presented in the MultiWOZ 2.0 [3] dataset. The dialogues range from requesting information to making bookings in 7 tourism related domains - *Attraction*, *Hospital*, *Police*, *Hotel*, *Restaurant*, *Taxi*, and *Train*. Budzianowski et al. [3] collected the dialogues in a WOz setting with the help of 1,249 crowdworkers. They state that having a large set of workers mitigates the problem of artificial encouragement of variety in dialogue from users. Goal changes were encouraged to model more realistic dialogue. Utterances were automatically labeled with user dialogue acts using heuristics and added in MultiWOZ 2.1 [10]. Han et al. [13] refined the dialogue act annotation using prediction and regular expressions, but imply that further improvements can be made as they hope to attract more research to further improve the quality of the dataset.

Microsoft Dialogue Challenge (E2E) [22] was introduced challenging participants to develop end-to-end task-completion dialogue systems. The challenge organizers, Li et al. [22], released a dataset of 10,087 dialogues, henceforth referred to as the E2E dataset. The dialogues are in the domains of movie-ticket booking, restaurant reservation and taxi ordering. The data was collected with crowdworkers on Amazon Mechanical Turk and human-annotated with dialogue acts.

Machines Talking To Machines (M2M) [33] is a framework proposed with the goal of reducing cost and effort to build dialogue datasets. Two datasets in the movie-ticket booking and restaurant reservation domain are released using the framework. The M2M framework generates dialogue outlines via self-play before they are paraphrased by crowdworkers. The datasets, Sim-M (Movie) and Sim-R (Restaurant), contain 3,008 dialogues and will be referred to as the M2M dataset in this work. The dataset is labeled with dialogue acts which the generated outlines are based on such that crowdworkers don't have to decode the meaning of the utterances and perform manual annotation. Shah et al. [33] argue that crowdsourcing using a Wizard-of-Oz setup is flawed as (i) crowdworkers might not cover all interactions an agent is meant to handle, (ii) crowdworkers might use overly simplistic or convoluted language, and (iii) it can cause errors in dialogue act annotation. They claim to achieve greater coverage of dialogue flows while keeping utterances realistic. However, they raise the concern that using the framework restricts generated dialogue flows to those engineered into the model. A setup with crowdworkers conversing could give dialogues not anticipated by the developer guiding the generation of the dialogue.

Schema-Guided Dialogue (SGD) [30] addresses the problem that existing task-oriented dialogue datasets don't sufficiently cover the large number of domains a virtual assistant in production is expected to handle. The dataset is made up of 22,825 dialogues in 20 domains, four of which are only present in the dev or test set. Among the domains are *Calendar*, *Events*, *Music*, *Weather*, *Travel*, and more. Dialogues were collected using a dialogue simulator. The simulator generates dialogue outlines which crowdworkers paraphrase to obtain conversational utterances. Utterances are automatically labeled with dialogue acts as a part of the outline generation. Rastogi et al. [30] argue that simulation-based collection is better than other

Table 3.3: Parts of example dialogue from SwDA with dialogue acts

Speaker	Utterance	Dialogue Act
A	Uh, let's see.	Hold before answer/agreement
A	How about, uh, let's see, about ten years ago,	Abandoned or Turn-Exit
A	Uh, what do you think was different ten years ago from now?	Open-Question
B	Well, I would say as fas as social changes go, uh, I think families were more together.	Statement-opinion
B	They, they did more things together.	Statement-opinion
A	Uh-huh.	Acknowledge (Backchannel)
	...	
B	I mean do you think, people really need two cars and –	Yes-No-Question
A	No.	No answers

approaches like Wizard-of-Oz because of fewer annotation errors, better coverage of dialogue flows and lower cost.

3.3.2 Other Dialogue Act Datasets

This section presents two popular dialogue act classification datasets that are not task-oriented.

Switchboard Dialogue Act Corpus (SwDA) [19] is made up of 1,155 telephone dialogues where a caller and receiver converse about various topics. The utterances are labeled with a set of 43 dialogue acts for general conversation. This corpus is not considered a task-oriented dataset since it doesn't follow the system/user setup. Table 3.3 shows parts of an example dialogue from the dataset.

MapTask Corpus [2] consists of 128 conversation between an instruction giver and follower, where the instruction giver tries to guide the instruction follower to draw a path on a map. The corpus is labeled with a set of 13 dialogue acts. Part of an example dialogue can be found in Table 3.4.

Table 3.4: Part of example dialogue from MapTask with dialogue acts

Speaker	Utterance	Dialogue Act
G	okay	ready
G	starting off we are above a caravan park	instruct
F	mmhmm	acknowledge
G	we are going to go due south straight south and then we're going to turn straight back round and head north past an old mill on the right hand side	instruct
F	due south and the back up again	check
G	yeah	reply_y
G	south and then straight back up again with an old mill on the right and you're going to pass it on the left-hand side of the mill	clarify

3.3.3 Domain-specific Intent Datasets

The following task-oriented dialogue datasets are popular datasets labeled with domain-specific intents. ATIS [14] is a dataset with recordings of flight reservations that has 21 intent labels in the flight domain. SNIPS [6] is a dataset labeled with seven intents related to a virtual assistant. HWU64 [40] is a dataset with 64 virtual assistant intents. CLINC150 [21] is a dataset with 150 intents spanning 10 domains like travel, dining, and small talk. Facebook's multilingual [32] dataset has utterances in the weather, alarm, and reminder domains with 12 intents.

3.4 Pre-trained Language Models

Since 2018, transformer-based pre-trained language models have contributed to performance gains on many downstream natural language processing tasks [29, 8]. They often use large unlabeled text corpora like Wikipedia, BooksCorpus or Reddit data for self-supervised training to gain language understanding that can be fine-tuned for specific tasks.

One such model is BERT, presented in Sect. 2.3.4, which is pre-trained on the two tasks *masked language modeling* and *next sentence prediction*. However, Henderson et al. [17] state that *response selection* is a more suitable pre-training task to learn representations of conversations. Response selection is the task of selecting the most appropriate response from a collection of possible responses, given the dialogue history. The authors present the pre-training framework ConveRT (Conversational Representations from Transformers), which uses Reddit data and the response selection task to pre-train a model. ConveRT achieves state-of-the-art performance on response selection tasks. The authors also show that the pre-trained representations transfer to the intent classification task by outperforming BERT-based baseline classifiers despite ConveRT training many times faster than BERT.

Mehri et al. [24] present ConvBERT, a model produced by further pre-training of BERT on 700 million conversations from online forums. Their input for pre-training includes the 3 last turns in the dialogue to model multi-turn dialogue. In a process named *task-adaptive pre-training*, the authors continue training with MLM on the target dataset on top of ConvBERT before fine-tuning. They also experiment with MLM training on the 7 task-oriented dialogue datasets of the DialoGLUE benchmark before fine-tuning for each task, which gives mixed results. ConvBERT achieves state-of-the-art results on the intent classification datasets of DialoGLUE with different combinations of their pre-training methods.

Gururangan et al. [12] attempt to adapt language models to domains and tasks with domain adaptive pre-training (DAPT) and task adaptive pre-training (TAPT). They continue training a BERT-based model with large amounts of unlabeled in-domain text for DAPT, and use available unlabeled data associated with the target task for TAPT. The authors find that both DAPT and TAPT consistently improve the BERT-based baseline on text classification tasks. TAPT uses far less pre-training data and is computationally cheaper to perform, but gives similar performance gains as DAPT. DAPT and TAPT combined achieves the best performance in all of their experiments.

TOD-BERT [39] is a BERT model further pre-trained on task-oriented dialogue. Wu et al. [39] hypothesize that self-supervised pre-training with task-oriented dialogue corpora can learn better representations and perform better on downstream tasks than existing pre-trained language models. For pre-training they collect 9 multi-turn, human-human, task-oriented dialogue datasets with over 100,000 dialogues spanning 60 domains. The authors fine-tune for 4 downstream tasks, including intent recognition and dialogue act prediction. They find that TOD-BERT outperforms BERT on all tasks. TOD-BERT has a clear advantage over BERT in a few-shot scenario where limited labeled training data is available.

3.5 Task-specific Architectures

One of the earlier works that took into use neural networks for intent classification is by Sarikaya et al. [31], where deep belief nets (DBNs) are applied to a natural language call-routing task. DBNs discover features through unsupervised learning which Sarikaya et al. use in a multi-layer feed forward neural network that is fine-tuned. Their approach produces better classification results than traditional classifiers like Maximum Entropy and Boosting classifiers.

More recent RNN and transformer-based approaches achieve state-of-the-art results by performing intent classification and slot filling, which fills the arguments of the intent, jointly. Wang et al. [37] introduce an asynchronously trained bi-model structure with shared internal state between two bi-directional LSTMs, one for intent classification and one for slot filling. After the recent shift to the pre-training paradigm, Chen et al. [4] propose a model based on the pre-trained language model BERT and achieve improved performance compared to non pre-trained models.

While the aforementioned models effectively solve the intent classification task on the single-turn ATIS dataset, classification of multi-turn dialogue, where system and user may refer to previous utterances, remains a challenge. Chen et al. [5] propose an RNN based memory network that encodes the dialogue history through an attention mechanism and achieve improved performance compared to models not incorporating context.

Qin et al. [28] propose a model incorporating contextual information based on a novel context-aware graph convolutional network (CGCN) which operates directly on a graph structure of dialogues. The input to the CGCN comes from bidirectional LSTM-encoded utterances and dialogues. Their approach improves upon the memory network of Chen et al. and other context inclusive architectures in dialogue act and intent classification on the M2M dataset.

Chapter 4

Method

This chapter describes alignment of the task-oriented dialogue act datasets from Sect. 3.3.1, henceforth referred to as the TODA corpus, to a universal DA schema. Architectures used in experiments are also presented.

4.1 Dialogue Act Schema Alignment

This section presents the universal DA schema and the alignment of the TODA corpus to the schema. Five datasets, previously not aligned by the authors of the schema, are aligned. This alignment is one of the main contributions of this thesis.

4.1.1 Universal DA Schema

Paul et al. [27] developed a universal dialogue act schema for task-oriented dialogue to enable supervised training of a universal dialogue act tagger with two datasets that have different dialogue act annotation.

The authors design their universal schema to cover all of the dialogue acts found in the DSTC 2 and M2M datasets. The schema is based on these datasets because they both have annotation schemas inspired by the CUED schema [41] for dialogue acts. They also align the MultiWOZ 2.0 dataset with their schema for evaluation of their tagger.

To create the universal schema the authors first take a union of the dialogue acts based on namespace and look at the distribution of the acts. They find that the

Table 4.1: Universal DA schema

ack, affirm, bye, deny, inform, repeat, reqalts, request, restart, thank_you, user-confirm, sys-impl-confirm, sys-expl-confirm, sys-hi, user-hi, sys-negate, user-negate, sys-notify-failure, sys-notify-success, sys-offer

datasets share few dialogue act names, and when they do there may be differences in semantics since the distributions of the acts can be very different.

Due to this, they perform a manual assessment of the semantics of the dialogue acts. After training a tagger with the manually aligned acts, some semantically similar acts that confused the tagger were found. Some of these acts were split and others were merged to improve the performance of the tagger. They ended up with a schema of 20 dialogue acts which can be found in Table 4.1. The alignment of the DSTC 2 and M2M datasets to the universal schema can be found in Table 4.2. Example dialogues from DSTC 2 and M2M can be found in Table 4.3 and 4.4.

4.1.2 Alignment of the TODA Corpus

Manual assessment of the semantics of dialogue acts is used to align the datasets of the TODA corpus to the universal DA schema. Dialogues from DSTC 2 and M2M were inspected to understand the semantics of each dialogue act in the universal schema. Following is an explanation of the not-so-obvious alignments of dialogue acts for each dataset. Even though we are only interested in classifying utterances with dialogue acts, accompanying slot-value pairs are included in the presentation and discussion of the alignment. This is because the same dialogue act with different slots and values can map to different acts in the universal schema. Frequency of user acts in the aligned TODA corpus can be found in Figure 4.1.

Frames has a user act *request_compare(x)* which requests a comparison of two or more options on a provided slot x. This act is mapped to *request* since the *request* act is for utterances where a value of a slot is requested (e.g., the utterance "How many days would I be in Kobe?" requests the value of the *duration* slot and is tagged with *request(duration)*).

The user act *moreinfo* accompanies utterances where a user requests more infor-

Table 4.2: Alignment of DSTC 2 and M2M with Universal DA Schema by Paul et al. [27]

Universal DA Schema	DSTC 2	M2M
inform	inform(x=y)	inform(x=y)
request	request(x)	request(x)
user-negate	negate(x=y)	negate(x=y)
sys-negate	negate(x=y)	negate(x=y)
user-hi	hello()	
user-hi + inform		greeting(x=y)
sys-hi	welcomemsg()	
reqalts	reqalts(), reqmore()	request_alts()
repeat	repeat()	cant_understand()
restart	restart()	
affirm	affirm()	affirm()
affirm + inform		affirm(x=y)
sys-impl-confirm	impl-conf(x=y)	
sys-expl-confirm	expl-conf(x=y), confirm-domain(x=y)	confirm(x=y)
user-confirm	confirm(x=y)	
sys-notify-failure	canthelp(), canthelp.exception()	notfiy_failure()
sys-notify-success		notify_success()
sys-offer	offer(x=y), select(x=y1,y2)	offer(x=y), select(x=y1,y2)
thank_you	thankyou()	thank_you()
bye	bye()	
ack	ack()	
deny	deny(x=y)	

Table 4.3: Example dialogue from DSTC 2 with dialogue acts

Actor	Utterance	Dialogue Acts
System	Hello, welcome to the Cambridge restaurant system. You can ask for restaurants by area, price range or food type. How may I help you?	welcomemsg()
User	west part of town cuban food	inform(area=west), inform(food=cuban)
System	Sorry there is no cuban restaurant in the west of town	canthelp()
User	italian	inform(food=italian)
System	prezzo is a nice restaurant in the west of town serving italian food	offer(name=prezzo), inform(food=italian), inform(area=west)
User	phone number	request(phone)
System	You are looking for a restaurant is that right?	confirm-domain()
User	yes	affirm()
System	prezzo is a nice restaurant in the west of town serving italian food	offer(name=prezzo), inform(food=italian), inform(area=west)
User	what is the phone number	request(phone)
System	The phone number of prezzo is 01799 521260	offer(name=prezzo), inform(phone=01799 521260)
User	good bye	bye()

Table 4.4: Example dialogue from M2M with dialogue acts

Actor	Utterance	Dialogue Acts
User	hi - buy 6 tickets for gimme danger at cinelux plaza theatre	greeting(), inform()
System	what date and time ?	request(time), request(date)
User	4:00 pm on wednesday	inform()
System	4:00 pm ?	confirm(time=4:00 pm)
User	yes	affirm()
System	i bought 6 tickets for gimme danger at cinelux plaza theatre for march 08 at 4:00 pm	notify_success()
User	bye	good_bye()

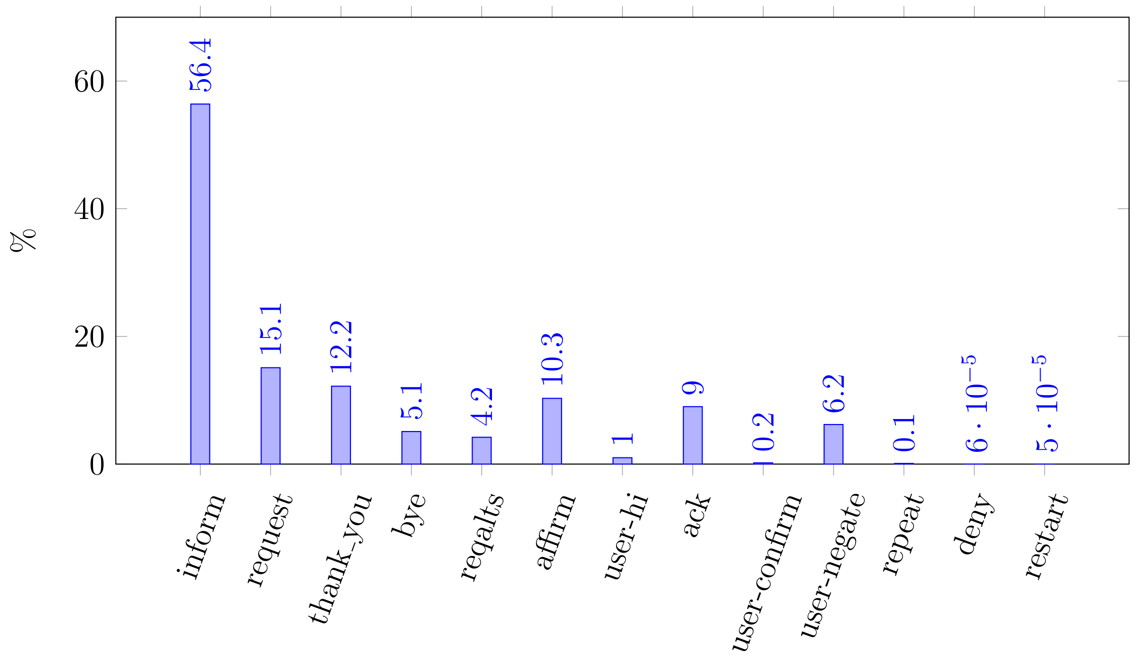


Figure 4.1: User act frequency in aligned TODA corpus

mation or a description of an option offered by the system (e.g., "What is the hotel like?"). In the E2E dataset, utterances like this are annotated with *request(other)* which leads us to map this act to *request*.

The system act *suggest* is split three ways when mapped to the universal schema due to its semantic diversity. (i) Utterances where the system asks if the user would like to book an offered option is annotated with *suggest(action=book)*. This is mapped to *sys-expl-confirm* since this act is for when the system asks for an explicit confirmation from the user. (ii) When the act is in the form *suggest(x=y)*, except for the situation in (i), the system offers values for one or more slots. In this situation the act is mapped to *sys-offer*. (iii) When the act contains a slot with no corresponding value, *suggest(x)*, the system asks if the user can provide a new value for the slot x (e.g., "What other cities are you considering?"). This is semantically similar to a *request* act and similar utterances in the E2E and MultiWOZ 2.3 datasets label these with *request*. This leads us to map *suggest(x)* to *request*.

Frames does not have its own act to notify success, but this can be extracted from the *inform* act when it contains the slot-value pair action=book. Utterances labeled with this informs the user that a booking has been completed and this is mapped to *sys-notify-success*.

The dataset contains about 600 utterances which are annotated with the *greeting* act. However, using a regular expression to match on common greeting words in the dataset, we find about 300 utterances that are not labeled with *greeting* that clearly should be. All of these are mapped to the *user-hi* or *sys-hi* act depending on whom the utterance comes from. The 300 utterances are labeled so that a model being trained with this data does not get punished for tagging an utterance like "Hi, I need to go somewhere warm!" with *user-hi*.

An example dialogue from Frames is presented in Table 4.5 and the complete alignment of Frames to Universal DA schema can be found in Table 4.7.

SGD has a user act *select* which is for utterances where a result offered by the system is selected. Some example utterances labeled with *select* are "Okay", "That sounds great", and "That works for me, please reserve the bus". The last example would also be tagged with *inform_intent* where the *select* act only refers to the positive feedback part of the utterance. This is similar to the act *ack* in DSTC 2

Table 4.5: Example dialogue from Frames with dialogue acts

Actor	Utterance	Dialogue Acts
User	Hi im in the middle of travelling and i want to go to Porto!	inform(intent=book), inform(dst_city=Porto), greeting()
System	Sure thing! What city will you be departing from?	request(or_city)
User	i can leave from Minneapolis	inform(or_city= Minneapolis)
System	I you leave from Minneapolis on the 17th of August I can book you 11 days in the four-star Golden Keep. The total cost of this package is 2362.21USD. Would you like me to book this package?	offer(), suggest(action=book)
User	Let's book it!	inform(intent=book), switch_frame()
System	Done! Enjoy!	inform(action=book)
User	thank you!	thankyou()

which has short utterances like "Good", "That'll do", and "Okay". The user act *select* is mapped to *ack* of the Universal DA schema.

The system act *offer_intent* is paired with utterances where the system asks if the user would like to make a reservation, booking, or similar. This is similar to *suggest(action=book)* in Frames and is mapped to *sys-expl-confirm.*

An example dialogue from SGD is presented in Table 4.6 and the complete alignment of SGD to Universal DA schema can be found in Table 4.7 along with the alignment of the Frames dataset.

E2E has a different idea of what a *request* act is compared to the other datasets. This becomes obvious when looking at the distribution of acts on the first user turn in each dialogue. In E2E 91.9% of the first user turns are labeled with *request* and only 3.3% with *inform.* In Frames it is the other way around with 1.1% *request* and 93.7% *inform.* In other datasets like M2M and SGD none of the first user turns are labeled with *request.* After making the corrections in the following paragraph the distribution of acts on the first user turn is flipped around to a more typical 88.4% *inform* and 6.7% *request.*

Table 4.6: Example dialogue from SGD with dialogue acts

Actor	Utterance	Dialogue Acts
User	I am searching to watch a movie like Fantasy kind and i need something with Colin Farrell in that movie	inform(genre=Fantasy), inform(actors=Colin Farrell), inform_intent(FindMovies)
System	I have selected 1 movie of your choice and what is your opinion on Dumbo?	offer(movie_name=Dumbo), inform_count(count=1)
User	Dumbo is fine as i need to rent the movie and have to watch it with no subtitles	inform(subtitle=None), inform_intent(RentMovie), select(movie_name=Dumbo)
System	Please give confirmation details: The movie you like to watch is Dumbo with no subtitles	confirm(movie_name=Dumbo), confirm(subtitle=None)
User	That is cool. Can i know the rental cost of that movie? who is the director of that movie?	affirm(), request(director), request(price)
System	Movie rented and you have to watch it within 3 days.The renting cost of the movie is $4 and the movie is directed by Tim Burton	inform(price=$4), inform(director=Tim Burton), notify_success()
User	Can you also tell me the weather o March 14th in Montara?	inform(date=March 14th), inform(city=Montara), inform_intent(GetWeather)
System	The average day temperature is running at 77 degrees Fahrenheit and the chances of rain is 21 percent	offer(temperature=77), offer(precipitation=21)
User	Fine and thanks for now	select(), goodbye()
System	Enjoy, be happy	goodbye()

Table 4.7: Alignment of Frames and SGD with Universal DA Schema

Universal DA Schema	Frames	SGD
inform	inform(x=y)	inform(x=y), inform_intent(y)
request	request(x), request_compare(x), moreinfo(), suggest(x)	request(x)
user-negate	negate(x=y)	negate(), negate_intent()
sys-negate	negate(x=y)	
user-hi	greeting()	
sys-hi	greeting()	
reqalts	request_alts()	request_alts(), req_more()
repeat	reject()	
affirm	affirm()	affirm(), affirm_intent()
sys-expl-confirm	confirm(x=y), hearmore(), suggest(action=book)	confirm(x=y), offer_intent()
user-confirm	confirm(x=y)	
sys-notify-failure	canthelp(), no_result()	notify_failure()
sys-notify-success	inform(action=book)	notify_success()
sys-offer	offer(x=y), suggest(x=y)	offer(x=y)
thank_you	thankyou()	thank_you()
bye	goodbye()	goodbye()
ack		select()

Upon inspection of the dialogues we find that the *request* act fits three different acts in the universal schema depending on accompanying slot-name and speaker. (i) User utterances labeled with *request(moviename)*, *request(ticket)*, *request(reservation)*, *request(restaurantname)*, or *request(taxi)* actually informs about the task the user wants performed (e.g., "I need movie tickets") or that the user wants an option booked or similar (e.g., "Great. make a reservation there"). In this case *request* is mapped to *inform*. (ii) System utterances labeled with *request(ticket)*, *request(reservation)*, or *request(taxi)* reads just like utterances labeled with *offer_intent* in the SGD dataset (e.g., "Hello! Would you like to make a restaurant reservation?"). This case is therefore mapped like *offer_intent* to *sys-expl-confirm*. (iii) In the case where an utterance is not covered by (i) nor (ii), the utterance fits the *request* act of the universal schema where the value of a slot is requested.

An example dialogue from E2E is presented in Table 4.8 and the complete alignment of E2E to Universal DA schema can be found in Table 4.9 with heuristics to fix the *request* act.

MultiWOZ 2.3 has the same annotation schema as MultiWOZ 2.0, which is why this alignment is based on the one of MultiWOZ 2.0 by Paul et al., but with a few additions.

When the first turn of a user in a dialogue is labeled with *general-greet*, it contains a greeting followed by information about what task they want to perform. Most of the time these utterances are only labeled with *general-greet*. These utterances are mapped to *user-hi* and *inform*.

An example dialogue from MultiWOZ 2.3 is presented in Table 4.10 and the complete alignment of MultiWOZ 2.3 to Universal DA schema can be found in Table 4.11.

DSTC 3 has a near identical annotation schema to that of DSTC 2. The dataset has a system act *canthelp.missing_slot_value* which we map to *sys-notify-failure*. The acts *impl-conf* and *confirm-domain* from DSTC 2 are not present in DSTC 3.

Table 4.8: Example dialogue from E2E with dialogue acts

Actor	Utterance	Dialogue Acts
User	Hi! are there any good foreign movies showing around Houma, Louisiana this week?	greeting(), request(moviename; genre=foreign; city=Houma)
System	What date would you like me to look for a reservation?	request(date)
User	How about the 9th?	inform(date=9th)
System	Unfortunately, there are no foreign movies playing at this time. Do you have another genre that you're interested in?	inform(result={}), request(genre)
User	Is there something that's maybe a good intelligent comedy?	request(moviename; genre=comedy)
System	Whiskey Tango Foxtrot is the only Adult comedy I see playing in your area. Would you like to try that?	inform(moviename= Whiskey Tango Foxtrot), inform(genre=comedy), confirm_question()
User	I guess I'll have to. Any night showing will be fine.	confirm_answer(), inform(date=any night)
System	Whisky Tango Foxtrot is playing at the AMC HOUMA PALACE 10 5737 W Park Ave., Houma, LA 70364 at 11:40am 2:15pm 5:00pm 7:40pm. Does one of those times work? How many tickets would you need?	inform(city=Houma, LA), inform(starttime=11:40am# 2:15pm#5:00pm#7:40pm), multiple_choice(starttime), request(numberofpeople)
User	740 for 4 would be great	inform(numberofpeople=4), inform(starttime=7:40pm)
System	I have purchased your tickets for you. Thank you.	inform(taskcomplete), thanks()
User	thank you.	thanks()
System	Thanks!	thanks()

Table 4.9: Alignment of E2E with Universal DA Schema including heuristics

Universal DA Schema	E2E	Heuristic
inform	inform(x=y)	
inform	request(x)	user utterances where x is 'moviename', 'ticket', 'reservation', 'restaurantname', or 'taxi'
request	request(x)	user utterances where x is not 'moviename', 'ticket', 'reservation', 'restaurantname', or 'taxi' AND system utterances where x is not 'ticket', 'reservation', or 'taxi'
user-hi	greeting(), request(greeting=y)	
sys-hi	greeting(), inform(greeting=y), request(greeting=y)	
affirm	confirm_answer()	
affirm + inform	confirm_answer(x=y)	
sys-expl-confirm	confirm_question(x=y)	
sys-expl-confirm	request(x)	system utterances where x is 'ticket', 'reservation', or 'taxi'
user-confirm	confirm_question(x=y)	
sys-notify-failure	inform(result={})	
sys-notify-success	inform(taskcomplete)	
sys-offer	multiple_choice(x=y1,y2)	
thank_you	thanks()	
deny	deny()	

Table 4.10: Example dialogue from MultiWOZ 2.3 with dialogue acts

Actor	Utterance	Dialogue Acts
User	I 'm looking for a restaurant that offers British food in a moderate price range , can you help me with that ?	Restaurant-Inform
System	Yes I have many in the center of town and on in the west . Do you have a preference for area ?	Restaurant-Request, Restaurant-Inform
User	Let 's try the west .	Restaurant-Inform
System	I have one in the west called Saint Johns Chop House . Would you like a reservation here ?	Restaurant-Inform, Booking-Inform(none=none)
User	For now , can I just get the phone number and postcode ?	Restaurant-Request
System	Their number is 01223353110 . Postcode is cb30ad .	Restaurant-Inform
User	Thanks , that 's everything I need .	general-thank
System	Thank you for choosing Cambridge TownInfo Centre .	general-bye

Table 4.11: Alignment of MultiWOZ 2.3 with Universal DA Schema including heuristics. The asterisk represents any sequence of characters.

Universal DA Schema	MultiWOZ 2.3	Heuristic
inform	Restaurant-Inform, Hotel-Inform, Attraction-Inform, Taxi-Inform, Train-Inform, Police-Inform, Hospital-Inform	doesn't end in Booking-Inform(none=none)
request	*-Request	
user-hi + inform	general-greet	turn = 1
reqalts	general-reqmore	
sys-expl-confirm	Train-OfferBook, Booking-Inform	
sys-notify-failure	Booking-NoBook, *-NoOffer	
sys-notify-success	Booking-Book, Train-OfferBooked	
sys-offer	*-Select, *-Recommend	
sys-offer	Restaurant-Inform, Hotel-Inform, Attraction-Inform, Taxi-Inform, Train-Inform	ends in Booking-Inform(none=none)
thank_you	general-thank	
thank_you	general-greet, general-bye, general-welcome	contains "*thank*"
bye	general-bye	
bye	general-greet, general-welcome, general-thank	contains "*bye"

4.2 BERT-based Classifier

The aligned TODA corpus is used to further pre-train a simple BERT-based classifier in a supervised fashion. We start with this simple architecture to investigate if the further pre-trained model fine-tuned on a target dataset performs better than a model trained on just the target dataset. We name the further pre-trained model TODA-BERT, short for task-oriented dialogue act BERT, and release the model and code[1].

Figure 4.2 illustrates the further pre-training/fine-tuning procedure which is described in the following paragraphs.

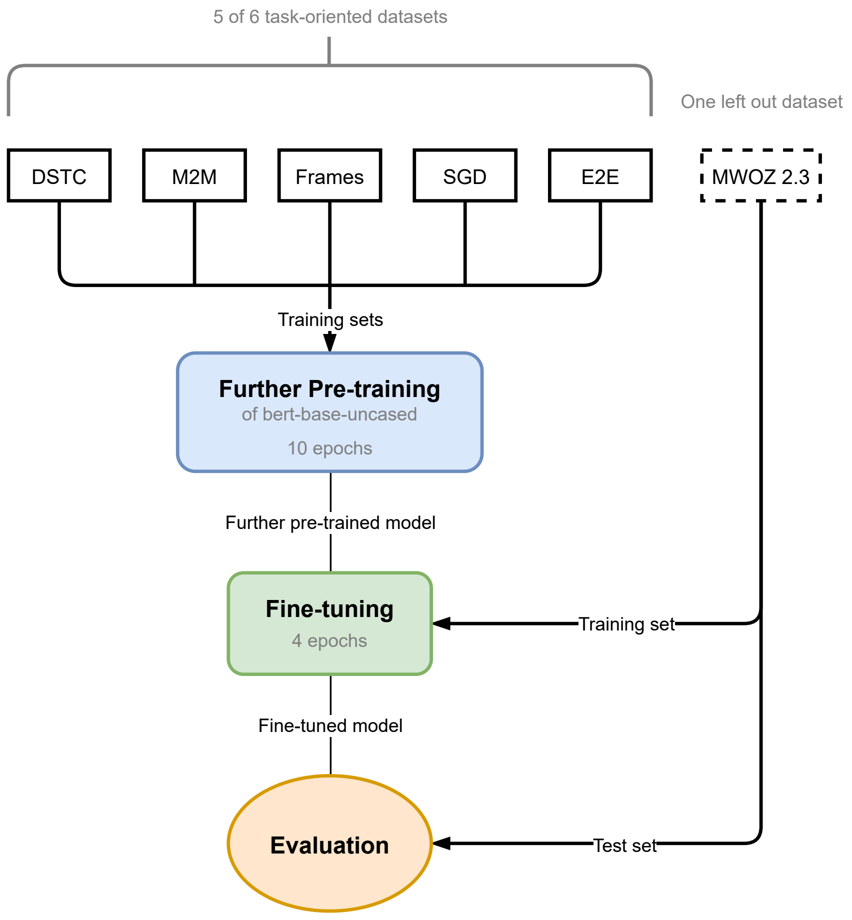


Figure 4.2: Pre-training/fine-tuning procedure.

[1]`https://github.com/hsyver/TODA-BERT`

Further Pre-training is done with the aligned TODA corpus. In the experiments where one of the six datasets is used for evaluation, the dataset is left out from pre-training. A BERT$_{\text{BASE}}$ model is further pre-trained on user turns of the TODA corpus in a supervised manner with the 13 classes of the universal DA schema present in user turns.

The architecture of the TODA-BERT model consists of the pooler output from the BERT$_{\text{BASE}}$ model, followed by a dropout layer (p=0.3) and feed-forward layer with 13 outputs, one for each possible dialogue act. The dropout probability is set to 0.3 based on results from a preliminary project.

Fine-tuning is carried out with four different architectures: **(a) BERT** which is the BERT$_{\text{BASE}}$ model, followed by a dropout layer (p=0.3) and a feed-forward layer with one output per dialogue act, **(b) TODA-BERT-replace** which is the pre-trained TODA-BERT model where the final feed-forward layer is replaced to fit the number of dialogue acts, **(c) TODA-BERT-add** which is the pre-trained TODA-BERT model with an added feed-forward layer with 13 inputs and one output per dialogue act, and **(d) TODA-BERT-filter** which is the pre-trained TODA-BERT model, where the outputs are post-filtered to only include the dialogue acts present in the dataset being fine-tuned on. The fourth architecture is only used when fine-tuning with the universal schema since further pre-training is done with this schema. The architectures are illustrated in Figure 4.3.

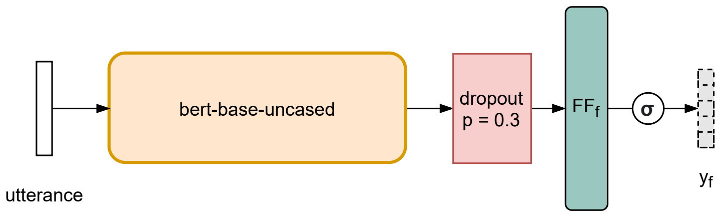
We experiment with both fine-tuning on the universal DA schema of the aligned dataset and fine-tuning on the original schema of the dataset. Experiments where only 20% of the training data is used for fine-tuning are also conducted.

In addition to fine-tuning on the individual datasets of the TODA corpus, we experiment with fine-tuning on the SwDA and MapTask datasets to test the further pre-trained model's ability to generalize to unseen datasets with different dialogue act schemas.

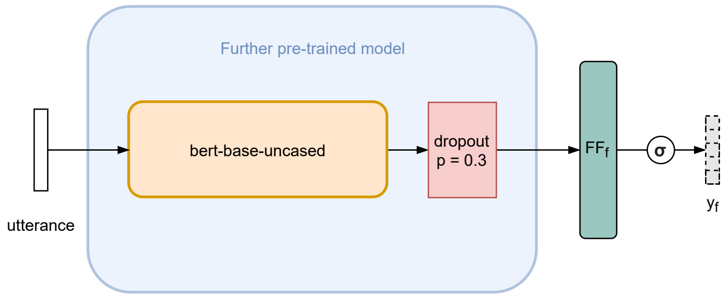
4.2.1 Experimental Setup

In both the BERT and TODA-BERT architecture, the bert-base-uncased pre-trained model from HuggingFace's transformer library [38] is used.

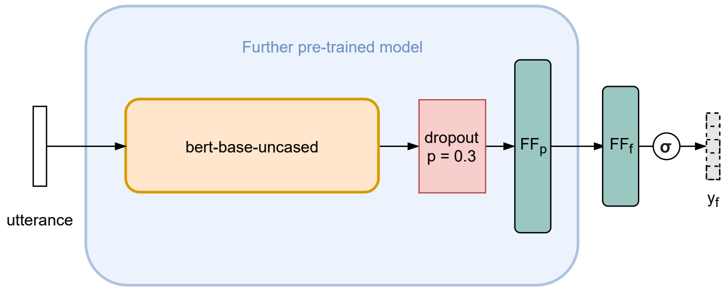
All models are trained using a binary cross entropy loss function. A sigmoid



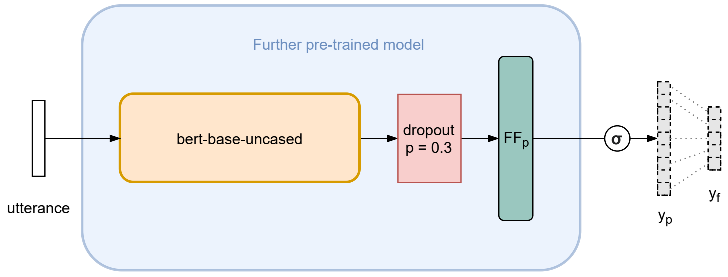
(a) BERT, fine-tuning without using a further pre-trained model.



(b) TODA-BERT-replace, fine-tuning on top of TODA-BERT where the feed-forward layer is replaced.



(c) TODA-BERT-add, fine-tuning on top of TODA-BERT where a feed-forward layer is added.



(d) TODA-BERT-filter, fine-tuning on top of TODA-BERT where outputs are post-filtered.

Figure 4.3: BERT-based fine-tuning architectures. Subscripts p and f denote the number of dialogue acts present when pre-training and fine-tuning, respectively.

function is applied to the outputs and a dialogue act is detected if the corresponding value is greater than or equal to 0.5.

Maximum sequence length is set to 80 as the longest utterances in the TODA corpus are about 80 tokens long. Shorter utterances are zero-padded to a length of 80 while the few longer utterances are truncated. Training is done with the AdamW optimizer [23] with a weight decay of 0.01. Hyperparameters are picked from the recommended values by Devlin et al. [8] for fine-tuning BERT: learning rate is set to 3e-5, batch size to 16 and number of epochs (passes of the entire training data) to 4. The learning rate is decayed to zero using a cosine annealing schedule without restarts. Further pre-training is stopped after 10 epochs based on the performance on a held out validation set.

Training, validation and dev splits are used for training, and the original test splits are used for evaluation when they are present. For the E2E dataset, 20% of the dialogues are chosen at random to create a test split. For the Frames dataset, we follow Abro et al. [1] and create a test split with the dialogues of participants with ids “U21E41CQP” and “U231PNNA3”. The test set of DSTC 2 is used for the combined dataset of DSTC 2 and 3.

When fine-tuning on MultiWOZ 2.3 with its original labels, we follow Wu et al. [39] and remove the domain information from the dialogue acts such that for example *Hotel-Request* becomes *Request*. This reduces the number of user acts from 32 to 12.

Experiments are conducted on one NVIDIA P100 or V100 GPU, depending on availability, provided by the NTNU IDUN/EPIC computing cluster [34].

4.3 U-DAT (Including Context Information)

The pre-training/fine-tuning procedure described in Sect. 4.2 is applied on the U-DAT architecture proposed by Paul et al. [27] that includes context information to detect dialogue acts in multi-turn dialogue. The U-DAT model pre-trained with the TODA corpus is named TODA-UDAT.

The experiments performed with the BERT-based classifier are also carried out with this architecture, e.g., fine-tuning on the universal schema and original schema with 100% and 20% of the training data.

The model architecture proposed by Paul et al. consists of four encoders: (1) an utterance encoder, (2) a dialogue encoder, (3) a past DA encoder, and (4) an agent encoder. The architecture is illustrated in Figure 4.4.

The utterance encoder is a bidirectional LSTM taking an utterance u_i as input and produces z_i which is a concatenation of the first hidden state of the backward LSTM, and the last hidden state of the forward LSTM:

$$z_i = \overleftarrow{LSTM}(u_i) \oplus \overrightarrow{LSTM}(u_i)$$

The dialogue encoder is a unidirectional LSTM that takes the encoded previous utterances in the dialogue $(z_1, ..., z_{i-1})$ as input and produces e_i:

$$e_i = LSTM(z_1, ..., z_{i-1})$$

The past DA encoder is a concatenation of the many-hot encoded vectors of past DAs $(d_1, ..., d_{i-1})$ resulting in p_i:

$$p_i = d_1 \oplus ... \oplus d_{i-1}$$

The agent encoder is simply a number g_i indicating whether the agent of the current utterance is a user $(g_i = 0)$ or the system $(g_i = 1)$. The concatenation of the encoded dialogue, past DAs, and agent produces the encoded context, C_i:

$$C_i = e_i \oplus g_i \oplus p_i$$

Finally, the encoded current utterance and encoded context are concatenated and passed through a feed forward layer, FF_j, giving y_j:

$$y_j = sigmoid(FF_j(z_i \oplus C_i))$$

Adjustments are made to the model architecture to facilitate the pre-training/fine-tuning procedure when fine-tuning TODA-UDAT on the original schema of a dataset. A feed forward layer is appended to the architecture with one output per dialogue act present in the target dataset and a feed forward layer with 20 outputs (one for each DA in the universal schema) is prepended to the architecture where past DA vectors $(d_1, ..., d_{i-1})$ are passed through before concatenation. When fine-tuning TODA-UDAT on a dataset with the universal schema, the output is post-filtered

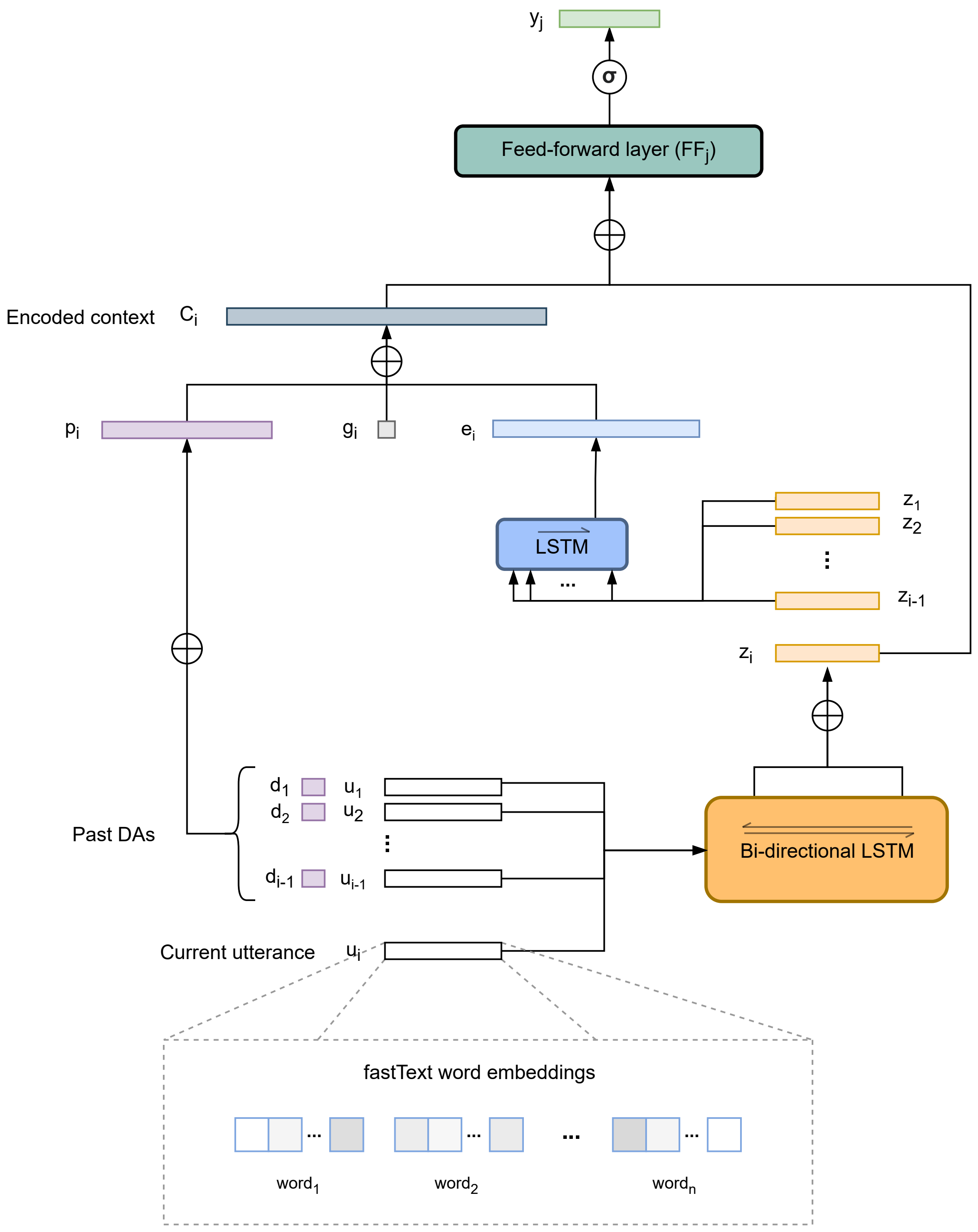


Figure 4.4: U-DAT model architecture [27].

such that only classes present in the dataset are considered when evaluating. Table 4.12 gives an overview of all the models presented in this chapter.

Table 4.12: Overview of model variants.

Model	**Pre-training**	**Architecture**	**Includes Context**
BERT	Self-supervised	Transformer-based	
TODA-BERT-replace	Supervised	Transformer-based	
TODA-BERT-add	Supervised	Transformer-based	
TODA-BERT-filter	Supervised	Transformer-based	
U-DAT	None	LSTM-based	✓
TODA-UDAT	Supervised	LSTM-based	✓

4.3.1 Experimental Setup

We follow the experimental setup of Paul et al. and set the hidden layer size of the bidirectional LSTM to 128 and the hidden layer size of the unidirectional LSTM to 256. Word embeddings are intialized with pre-trained fastText [25] embeddings which are fine-tuned during training. Utterances are tokenized with the basic english tokenizer from PyTorch[2] and a lookup table for embeddings is initialized with the unique tokens of the training data. During pre-training of TODA-UDAT and fine-tuning without TODA-UDAT, the embeddings are not fine-tuned for the first 5 epochs to avoid possible forgetting of the fastText embeddings as the model begins to learn.

Memory size, i.e., number of previous utterances to include, is set to 5 to limit training time. If an utterance has a dialogue history greater than 5, the utterances most distant in time are removed. Batch size is set to 16 and training is performed for 10 epochs. Following Paul et al., ADAM is used for optimization with default parameters and a learning rate of 0.001.

Since we are only concerned with classifying user utterances, g_i is always set to 0. In the past DA representation, observable system DAs and previously detected user DAs are used.

[2]`https://pytorch.org/text/_modules/torchtext/data/utils.html`

Chapter 5

Results & Discussion

5.1 BERT-based Classifier

This section presents and discusses the results of the experiments described in Sect. 4.2.

The results in this section are used to answer our first main research question, **RQ1**: How does the performance of a further pre-trained BERT model compare to a BERT model that's only fine-tuned?

5.1.1 Universal Schema vs. Original Schema

The results in this section are the basis for answering the research question **RQ1.1**: How does further pre-training affect performance when training and evaluating on the universal and original dialogue act schema?

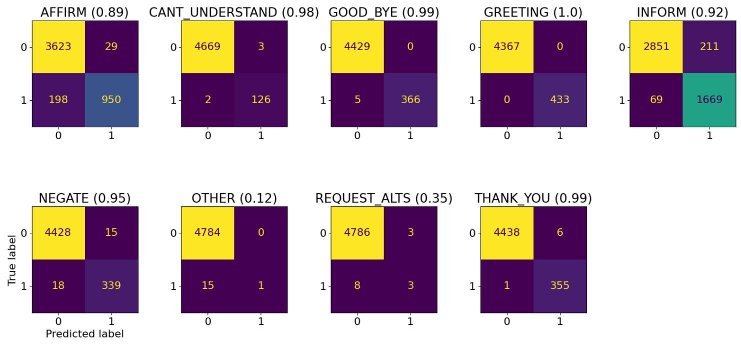
Results of experiments with the universal and original schema can be found in Table 5.1. When comparing the three different models trained on a further pre-trained model to the model that's only fine-tuned, the highest macro-F1 scores are often found in the pre-trained models. This trend is most prominent when fine-tuning with the original schema, where increases from 1.71 to 6.78 in macro-F1 score are found in four out of six datasets. When fine-tuning with the universal schema, a notable increase in macro-F1 score is found in two out of six datasets. When these increases are observed, the accompanying micro-F1 score usually remains at the level of the model that's only fine-tuned, but can have an decrease of up to 1.45.

Table 5.1: Micro-F1 and macro-F1 scores of the BERT-based classifier using 100% of the training data. Scores are averages of five runs.

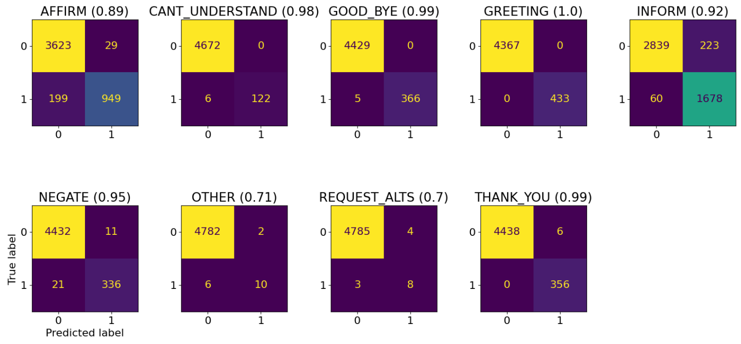
	Dataset	**BERT (a)**		**TODA-BERT-replace (b)**		**TODA-BERT-add (c)**		**TODA-BERT-filter (d)**	
		micro	**macro**	**micro**	**macro**	**micro**	**macro**	**micro**	**macro**
Universal schema	DSTC	98.13	81.33	98.16	83.14	**98.25**	**87.96**	98.19	84.63
	M2M	**96.09**	**93.62**	96.06	93.31	95.83	93.12	95.61	93.05
	Frames	**85.71**	62.94	84.71	62.02	84.76	61.63	84.26	**65.50**
	SGD	94.06	92.42	94.15	92.40	94.15	**92.45**	**94.17**	92.44
	E2E	**94.31**	**75.71**	94.12	74.74	94.10	74.92	94.01	74.55
	MWOZ	**98.00**	**92.34**	97.93	89.87	97.96	90.45	97.96	90.45
	Average	**94.38**	83.06	94.19	82.58	94.18	83.42	94.03	**83.44**
Original schema	DSTC	98.10	74.35	98.10	78.01	**98.15**	**81.13**		
	M2M	**94.29**	82.46	94.01	83.84	93.81	**85.80**		
	Frames	**79.35**	**52.58**	78.72	49.04	78.09	51.25		
	SGD	92.81	90.20	92.77	90.15	**92.87**	**90.31**		
	E2E	**91.03**	47.27	90.87	**48.98**	90.89	48.44		
	MWOZ	**97.85**	38.40	97.76	40.10	97.78	**42.17**		
	Average	**92.24**	64.21	92.04	65.02	91.93	**66.52**		

Analysis

This increase in macro-F1 score can be explained by improved performance on classes with few positive samples in the test set. A few correctly classified samples from a small class can increase the macro-F1 score without having a notable impact on the micro-F1 score since the macro score averages the F1 score of all the classes, while the micro-F1 score counts classifications globally. Improved performance on smaller classes can be observed when comparing the multi-label confusion matrices of BERT (Figure 5.1a) and TODA-BERT-add (Figure 5.1b) on M2M with the original schema. TODA-BERT-add is better at detecting the smallest classes *OTHER* and *REQUEST_ALTS* with higher numbers in the fourth quadrant, representing true positives. Similar observations are made in the confusion matrices of the other datasets where a small number of correctly classified samples from small classes increases the macro-F1 score.



(a) BERT

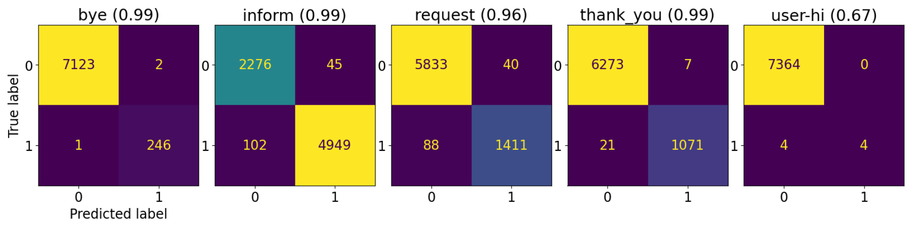


(b) TODA-BERT-add

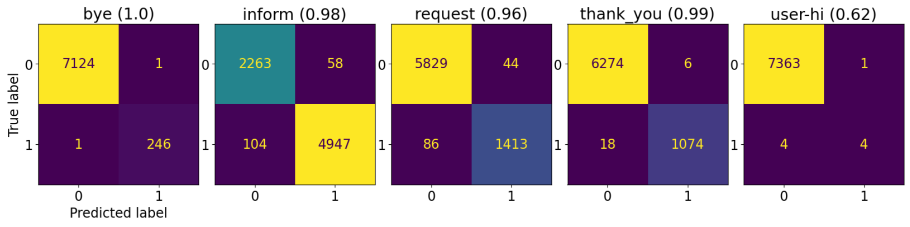
Figure 5.1: Confusion matrix and F1 score for each class after evaluation on M2M with the original schema. 100% of the training data was used for fine-tuning.

A likely explanation for the improvement on smaller classes is that they are learned during further pre-training and transfer to the model fine-tuned on top. A model that's only fine-tuned may have very few training samples for a dialogue act like *REQUEST_ALTS*, while a further pre-trained model takes advantage of the entire TODA corpus which has more samples in total with similar dialogue acts that have been unified.

A setup where the further pre-trained models fail at improving on the only fine-tuned model, but rather perform worse, is when fine-tuning on the MWOZ dataset with the universal schema. Comparing the confusion matrices of BERT (Figure 5.2a) and TODA-BERT-filter (Figure 5.2b), it can be observed that the decrease in macro-F1 score is mostly caused by a single false positive classification of the *user-hi* class. Some additional false positive classifications of the *inform* class can explain the slight decrease in micro-F1 score.



(a) BERT

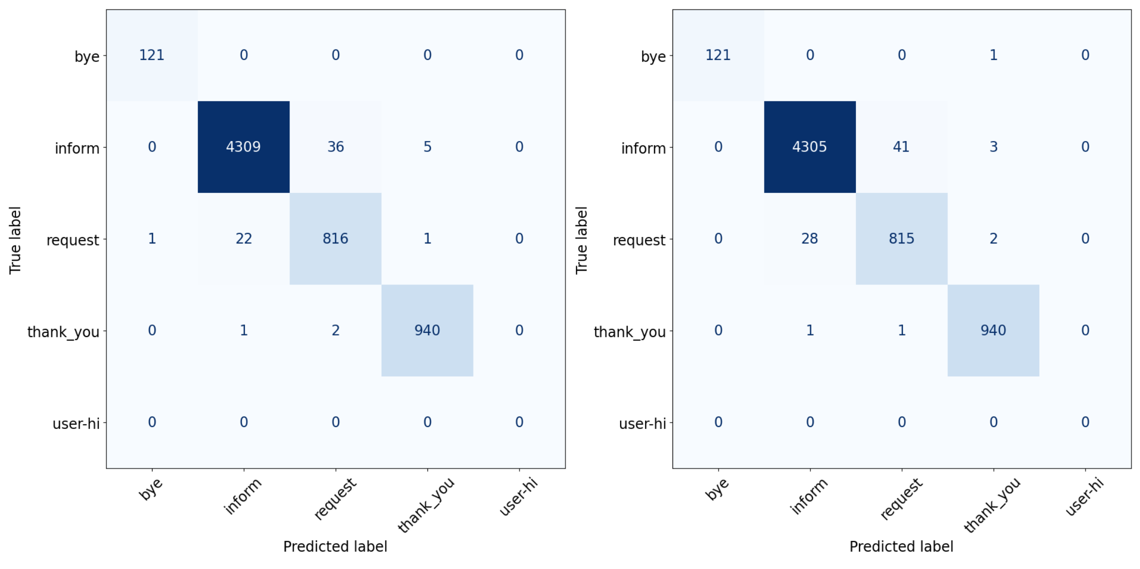


(b) TODA-BERT-filter

Figure 5.2: Confusion matrix and F1 score for each class after evaluation on MWOZ with the universal schema. 100% of the training data was used for fine-tuning.

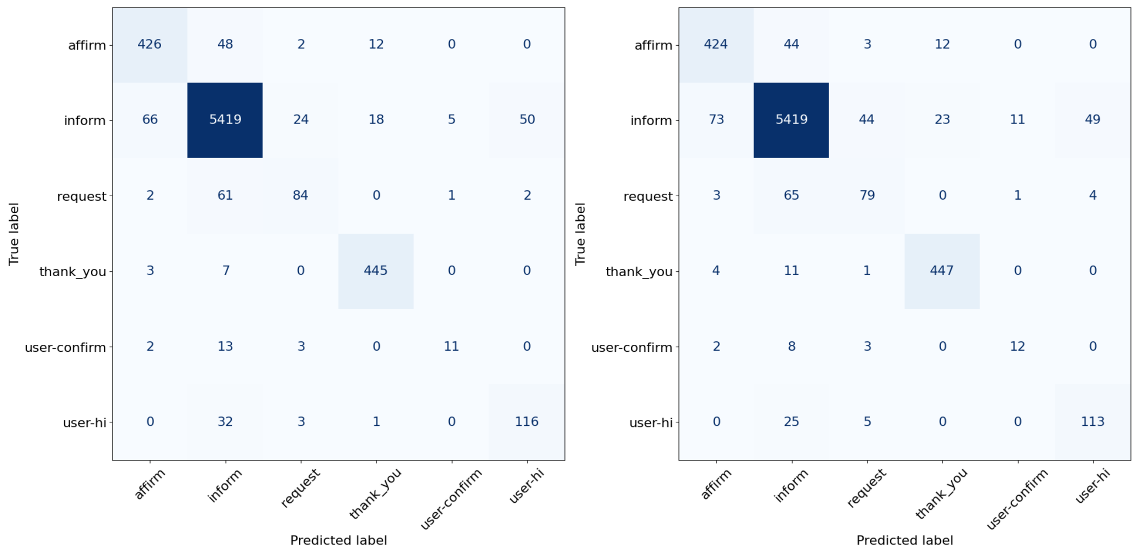
Which classes that are confused with each other can't be observed in the aforementioned figures since this is a multi-label classification problem where there is no way of telling which detected class is meant for which target class when there are multiple target classes for an utterance. However, in the test sets of this experiment, utterances with more than one ground truth dialogue act make up only 18.1% of the samples on average. By leaving these samples out, a single confusion matrix can be made for the remaining utterances with only one ground truth dialogue act. Figure 5.3 shows this confusion matrix for the two models discussed in the previous paragraph. These confusion matrices show that there is some confusion between the classes *inform* and *request*, more so in TODA-BERT-filter than BERT. Taking a closer look at these samples, we find that the models almost always detect both *inform* and *request* when only one of them is correct. One such example is the utterance "Can you tell me what time the train leaves?", where the ground truth label is *request*, but the model detects *inform* as well.

This increased confusion between *inform* and *request* when using a further pre-trained model can also be observed when fine-tuning on other datasets. In Figure 5.4 it can be observed in the E2E dataset. The elevated confusion when fine-tuning on top of a pre-trained model could be caused by inconsistencies among the *inform* and *request* classes in the TODA corpus. An example of a possible source for this is the E2E dataset which had the classes completely mixed up before an imperfect unification of dialogue act schemas was done.



(a) BERT **(b)** TODA-BERT-filter

Figure 5.3: Confusion matrix and F1 score for single-class samples after evaluation on MWOZ with the universal schema. 100% of the training data was used for fine-tuning.



(a) BERT **(b)** TODA-BERT-filter

Figure 5.4: Confusion matrix and F1 score for single-class samples after evaluation on E2E with the universal schema. 100% of the training data was used for fine-tuning.

Summary

The expected outcome of the experiment was that bigger improvements would be observed using the universal schema than the original schema, since further pre-training is done with the universal schema. Transfer learning from TODA-BERT was thought easier with the universal schema as it only had to be fit to the subset of classes present in the fine-tuning dataset. The task of fine-tuning with the original schema was expected to be more difficult since a mapping from the universal schema back to the original schema had to be learned as well. Surprisingly, bigger improvements are found using the original schema. Different observations are made when only fine-tuning with 20% of the training data, however, which is presented and discussed in the following section.

5.1.2 Limited Training Data

The results in this section are the basis for answering the research question **RQ1.2**: How does further pre-training affect performance when there is limited training data available?

Table 5.2: Micro-F1 and macro-F1 scores of the BERT-based classifier using 20% of the training data. Scores are averages of five runs.

	Dataset	BERT (a)		TODA-BERT-replace (b)		TODA-BERT-add (c)		TODA-BERT-filter (d)	
		micro	**macro**	**micro**	**macro**	**micro**	**macro**	**micro**	**macro**
Universal schema	DSTC	97.44	60.77	**97.65**	61.10	96.79	66.16	97.55	**73.09**
	M2M	**95.08**	81.58	94.17	79.28	93.87	83.05	93.62	**89.41**
	Frames	81.35	39.09	81.36	39.16	79.68	36.86	**82.37**	**61.00**
	SGD	**93.59**	**91.69**	93.50	91.46	93.47	91.44	93.42	91.30
	E2E	**93.60**	67.99	93.51	67.96	93.29	68.10	93.27	**72.39**
	MWOZ	97.42	90.46	**97.44**	**91.11**	**97.44**	90.92	97.43	88.80
	Average	**93.08**	71.93	92.94	71.68	92.42	72.76	92.94	**79.33**
Original schema	DSTC	97.57	55.07	**97.64**	56.09	95.41	**57.55**		
	M2M	**91.45**	**72.42**	90.67	69.75	88.90	70.51		
	Frames	72.43	25.03	**72.93**	**25.55**	68.13	24.76		
	SGD	**92.16**	**89.12**	92.01	89.01	92.11	**89.12**		
	E2E	**90.07**	**43.89**	89.94	43.65	89.45	43.65		
	MWOZ	**97.32**	36.39	97.27	**37.43**	97.18	35.24		
	Average	**90.17**	**53.65**	90.08	53.58	88.53	53.47		

Results of the experiments with 20% of the training data available can be found in Table 5.2. When comparing the three different further pre-trained models to the model that's only fine-tuned, increased macro-F1 scores are found in some of the setups. Unlike in the experiment from Sect. 5.1.1, which uses 100% of the available training data for fine-tuning, elevated macro-F1 scores are found when fine-tuning with the universal schema in this experiment. Increases between 4.4 and 21.9 in macro-F1 score are found in four out of six datasets. Like in Sect. 5.1.1, micro-F1 score tends to decrease a little or stay about the same as the micro-F1 score of the model that's only fine-tuned.

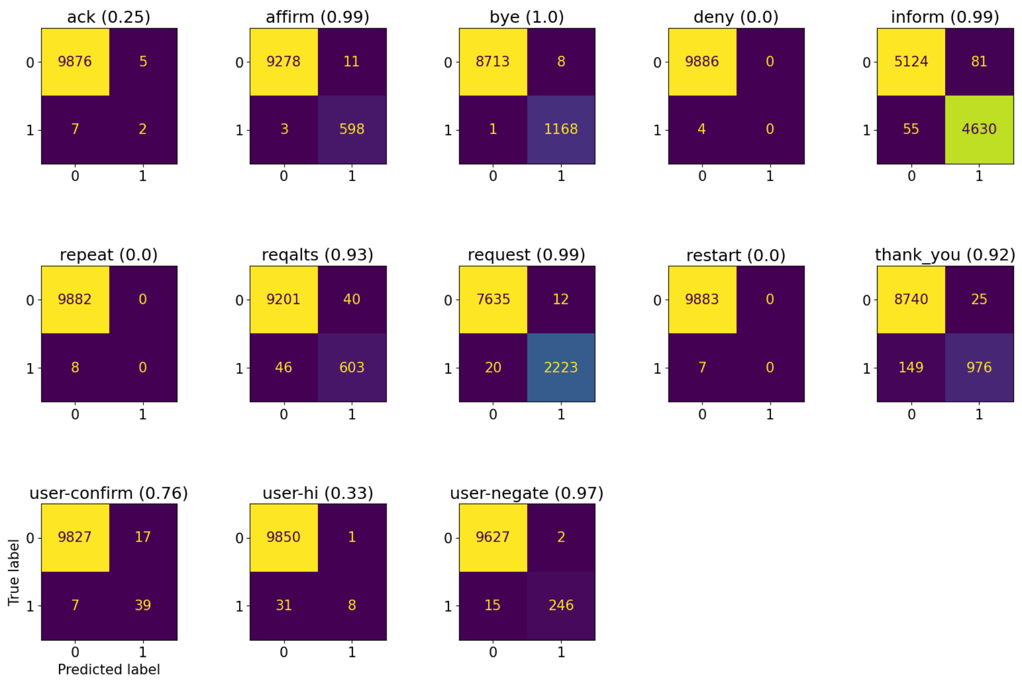
Analysis

Looking at the multi-label confusion matrices of BERT (Figure 5.5a) and TODA-BERT-filter (Figure 5.5b), when fine-tuning on DSTC with the universal schema, improved performance on the smaller classes can be observed like in the previous experiment. Specifically, improvements are made in the *repeat*, *user-hi*, and *reqalts* classes. Performance is good on the classes with many positive samples in both architectures.

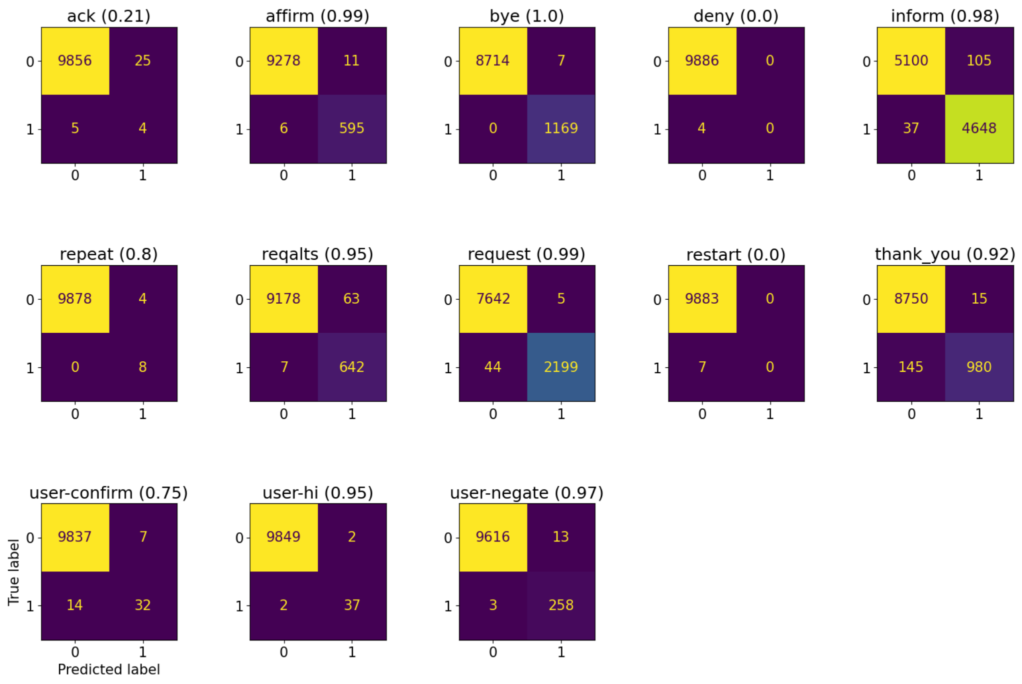
The two datasets that don't see an improvement in macro-F1 score with a further pre-trained model are SGD and MWOZ. These two datasets are by far the largest of the six (see Table 3.2) and have a larger number of positive samples in their smallest classes compared to the other datasets. Even when limiting training data to 20%, these datasets remain fairly large. SGD and MWOZ also don't include some of the typical small classes like *user-confirm* or *repeat*. These points are likely the cause of good performance overall on these two datasets when only fine-tuning. The additional small-class samples from the TODA corpus doesn't help improve performance when using a further pre-trained model.

Summary

The expected results of this experiment was that bigger improvements would be seen with the further pre-trained models compared to the experiment which used 100% of the training data. The reasoning behind this is that the TODA corpus and TODA-BERT will help more when BERT struggles with limited training data. This is indeed what was observed with much larger improvements in macro-F1 score, although improvements were found with the universal schema in this experiment and mostly with the original schema in the experiment with 100% training data. A possible explanation for the lack of improvement with the original schema in this experiment could be that the mapping from the universal schema to the original schema wasn't sufficiently learned when fine-tuning TODA-BERT because of limited fine-tuning data. However, these results are more in line with the expectation from the previous experiment where it was expected that improvements with the universal schema would be greater than with the original schema.



(a) BERT



(b) TODA-BERT-filter

Figure 5.5: Confusion matrix and F1 score for each class after evaluation on DSTC with the universal schema. 20% of the training data was used for fine-tuning.

5.1.3 Adapting Further Pre-trained Model to Target Dataset

The results in this section are the basis for answering the research question **RQ1.3**: How do different architectures for fine-tuning a further pre-trained model to datasets with varying annotation schemas affect performance?

To answer this question, one has to look at the results from the two previous experiments in Table 5.1 and Table 5.2, specifically in the three rightmost columns of the tables where the different architectures for fine-tuning TODA-BERT are found.

When fine-tuning with 100% of the training data, TODA-BERT-add consistently outperforms TODA-BERT-replace when looking at macro-F1 score, while micro-F1 is about the same for the two architectures. The improvement with an added layer against replacing the last layer is not as consistent when fine-tuning with 20% of the training data, but it performs better more often than not with respect to macro-F1 score. In the cases where the macro-F1 score is worse for TODA-BERT-add, the micro-F1 score is also usually lagging behind TODA-BERT-replace.

Analysis

The datasets where TODA-BERT-add performs notably worse is on Frames with the universal schema and on Frames and MWOZ with the original schema. Although, on these datasets, some of the five runs that are averaged in Table 5.2 perform as well as TODA-BERT-replace and sometimes better. This can be observed in Table 5.3 which shows the macro-F1 scores from fine-tuning with 20% of training data along with standard deviation of the five runs. TODA-BERT-add generally has a lot higher standard deviation and is less stable than TODA-BERT-replace. TODA-BERT-add has much higher scores in some of the runs, but also has runs with worse performance, pulling the average down.

Summary

Overall it looks like TODA-BERT-add performs better than TODA-BERT-replace. A possible explanation for this could be that retaining as much as possible of the pre-trained TODA-BERT model is beneficial, leaving the last added layer to learn the mapping from the pre-training classes to the classes in the target dataset. Replacing the last classification layer of TODA-BERT likely removes pre-trained weights

Table 5.3: Macro-F1 scores with standard deviation of five runs of the BERT-based classifier using 20% of the training data.

	Dataset	BERT (a)	TODA-BERT-replace (b)	TODA-BERT-add (c)	TODA-BERT-filter (d)
		macro (SD)	**macro (SD)**	**macro (SD)**	**macro (SD)**
Universal schema	DSTC	60.77 (2.27)	61.10 (5.09)	66.16 (**9.06**)	73.09 (1.54)
	M2M	81.58 (0.97)	79.28 (2.70)	83.05 (**3.73**)	89.41 (0.76)
	Frames	39.09 (1.76)	39.16 (2.62)	36.86 (**5.80**)	61.00 (1.44)
	SGD	91.69 (**0.14**)	91.46 (0.07)	91.44 (0.12)	91.30 (0.10)
	E2E	67.99 (0.37)	67.96 (**1.27**)	68.10 (0.86)	72.39 (0.71)
	MWOZ	90.46 (1.26)	91.11 (1.26)	90.92 (0.81)	88.80 (**1.83**)
	Average	71.93 (1.13)	71.68 (2.17)	72.76 (**3.40**)	79.33 (1.06)
Original schema	DSTC	55.07 (0.59)	56.09 (2.90)	57.55 (**8.31**)	
	M2M	72.42 (1.64)	69.75 (**3.37**)	70.51 (1.67)	
	Frames	25.03 (1.81)	25.55 (1.34)	24.76 (**3.67**)	
	SGD	89.12 (0.18)	89.01 (**0.21**)	89.12 (0.16)	
	E2E	43.89 (0.09)	43.65 (0.31)	43.65 (**0.51**)	
	MWOZ	36.39 (0.76)	37.43 (0.65)	35.24 (**2.43**)	
	Average	53.65 (0.85)	53.58 (1.46)	53.47 (**2.79**)	

important for classification.

Post-filtering of outputs, which is only possible with the universal schema, solves the instability issue of TODA-BERT-add and performs a lot better than both architectures overall when fine-tuning with 20% of the training data. It looks like adding a layer is unnecessary when fine-tuning with the universal schema. Simply filtering the output to only include the classes in the target dataset leaves less room to introduce errors in an additional layer. TODA-BERT-filter doesn't improve as much when fine-tuning with 100% of the training data. In this case the approach performs similarly to TODA-BERT-add, except for on the Frames and DSTC datasets, where the difference in macro-F1 score is 3.87 and -3.33, respectively.

5.1.4 Different Dialogue Act Schemas

The results in this section are the basis for answering the research question **RQ1.4**: How does further pre-training affect performance on datasets with different dialogue act schemas?

Results of experiments on the SwDA and MapTask datasets can be found in Table 5.4, which shows that performance is worse on both datasets when fine-tuning TODA-BERT. Both TODA-BERT-replace and TODA-BERT-add produce lower accuracy scores. TODA-BERT-filter was not used here since the datasets are not unified to the universal schema.

A possible explanation for the decreased performance of fine-tuned TODA-BERT models on these datasets could be that the utterances are too different from the TODA corpus. The general conversational language of SwDA and MapTask, exemplified in Sect. 3.3.2, might have more in common with the pre-training corpus of the BERT base model: BooksCorpus and English Wikipedia.

Another reason for the decreased performance could be that the dialogue act schemas of SwDA and MapTask are too different from the universal schema. Although they have some semantically similar acts like *Acknowledge* and *Thanking*, task-specific features learned from the TODA corpus are likely not very transferable to SwDA and MapTask.

Table 5.4: Accuracy scores (%) of the BERT-based classifier trained on the SwDA and MapTask datasets. Scores are averages of five runs.

Dataset	BERT (a)	TODA-BERT-replace (b)	TODA-BERT-add (c)
SwDA	**72.10**	71.92	71.95
MapTask	**62.73**	62.42	62.05
Average	**67.42**	67.17	67.00

5.2 U-DAT (Including Context Information)

This section presents and discusses the results of the experiments described in Sect. 4.3.

The results in this section are used to answer our second main research question, **RQ2**: How does the performance of a pre-trained dialogue context inclusive model compare to a model that's only fine-tuned?

5.2.1 Universal Schema vs. Original Schema

The results in this section are the basis for answering the research question **RQ2.1**: How does further pre-training affect performance when training and evaluating on the universal and original dialogue act schema?

Results of experiments with the universal and original schema can be found in Table 5.5, where improvements in both micro and macro-F1 score can be observed across all the datasets when fine-tuning TODA-UDAT. When fine-tuning with the universal schema, improvements in micro-F1 scores are between 2.34 and 5.44 in four out of the six datasets. Macro-F1 scores improve between 6.05 and 14.67 in five of the datasets. When fine-tuning with the original schema, improvements in micro-F1 scores range from 1.13 to 4.03 in four out of six datasets, while macro-F1 scores improve between 1.33 and 4.96 in all datasets. In the cases that TODA-UDAT doesn't improve upon U-DAT, the performance is about the same.

Analysis

A result that stands out is the improved scores for the SGD dataset with the original schema. With this dataset pre-trained models failed to make any improvements

Table 5.5: Micro-F1 and macro-F1 scores of the U-DAT-based classifier using 100% of the training data. Scores are averages of five runs.

	Dataset	**U-DAT**		**TODA-UDAT**	
		micro	**macro**	**micro**	**macro**
Universal schema	DSTC	97.05	64.77	**97.09**	**70.82**
	M2M	91.74	81.60	**95.08**	**90.57**
	Frames	71.53	36.77	**76.51**	**51.44**
	SGD	**94.69**	**92.12**	94.63	91.84
	E2E	88.08	64.73	**93.52**	**76.32**
	MWOZ	92.47	75.69	**94.81**	**82.45**
	Average	89.26	69.28	**91.94**	**77.24**
Original schema	DSTC	**97.15**	60.93	97.05	**62.26**
	M2M	89.47	74.02	**93.50**	**77.43**
	Frames	66.86	27.99	**70.22**	**32.95**
	SGD	93.22	91.60	**94.99**	**94.76**
	E2E	88.49	43.29	**88.62**	**44.94**
	MWOZ	93.08	32.00	**94.21**	**34.14**
	Average	88.05	54.97	**89.77**	**57.75**

with the BERT-based classifier. However, this context inclusive architecture improves both micro and macro-F1 scores notably when comparing the pre-trained TODA-UDAT to the non pre-trained U-DAT, outperforming the BERT-based classifier. This architecture is usually not competitive with the BERT-based classifier, producing lower scores in most datasets. A possible explanation for the improvement could be that the model looks at previous DAs in the dialogue history to better classify the current utterance. Since the SGD dataset is generated from dialogue outlines made up of dialogue acts, patterns of preceding DAs could be easy to learn and helpful for classification. An example is that the *NEGATE_INTENT* user act is always preceded by the *OFFER_INTENT* act. In a run of TODA-BERT-filter the F1-score of the *NEGATE_INTENT* class is about 86, while TODA-UDAT gets an F1-score of about 97 for this class, indicating that this context inclusive model manages to learn the relationship between the two classes. That TODA-UDAT improves upon U-DAT with SGD indicates that relationships between classes from the TODA corpus are successfully learned and applied to the target dataset.

As opposed to the BERT-based classifier, consistent improvements are made in micro-F1 score as well. This improvement could be attributed to the models ability to learn context.

Summary

Like in the corresponding experiment with the BERT-based classifier, results were expected to show bigger improvements when fine-tuning with the universal schema than the original schema. The same reasoning lies behind this expectation: that transfer learning from the pre-trained model should be easier with the universal schema, because this is the schema of the TODA corpus used for further pre-training. Unlike with the BERT-based classifier, where only results from fine-tuning with 20% training data supports this, performance is clearly better overall with the universal schema when using 100% of the training data. Improvements are also found with the original schema, although smaller. Experiments with 20% training data also point this way, which the results of can be found in the following section.

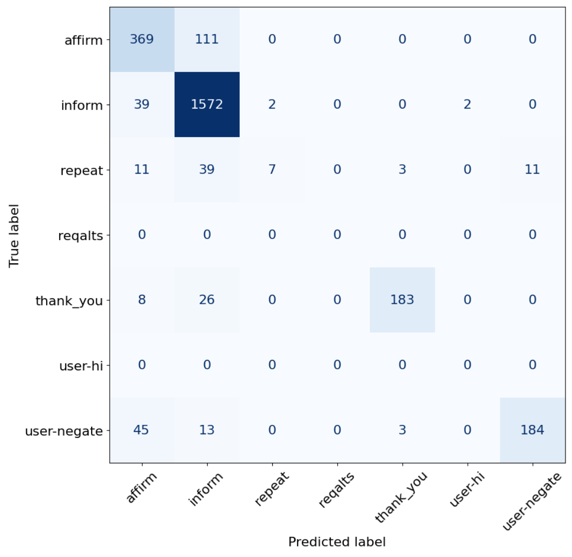
Table 5.6: Micro-F1 and macro-F1 scores of the U-DAT-based classifier using 20% of the training data. Scores are averages of five runs.

	Dataset	**U-DAT**		**TODA-UDAT**	
		micro	**macro**	**micro**	**macro**
Universal schema	DSTC	93.98	53.46	**95.60**	**62.98**
	M2M	81.33	58.56	**93.66**	**83.46**
	Frames	65.70	27.66	**75.33**	**48.81**
	SGD	93.23	90.24	**93.73**	**90.73**
	E2E	89.50	58.24	**92.05**	**69.48**
	MWOZ	90.71	71.44	**93.45**	**74.68**
	Average	85.74	59.93	**90.64**	**71.69**
Original schema	DSTC	93.71	**49.89**	**94.03**	49.20
	M2M	77.48	50.03	**85.85**	**60.73**
	Frames	56.56	16.57	**63.45**	**21.09**
	SGD	92.26	91.00	**93.58**	**92.67**
	E2E	87.06	40.13	**87.91**	**41.21**
	MWOZ	89.50	28.50	**92.39**	**30.00**
	Average	82.76	46.02	**86.20**	**49.15**

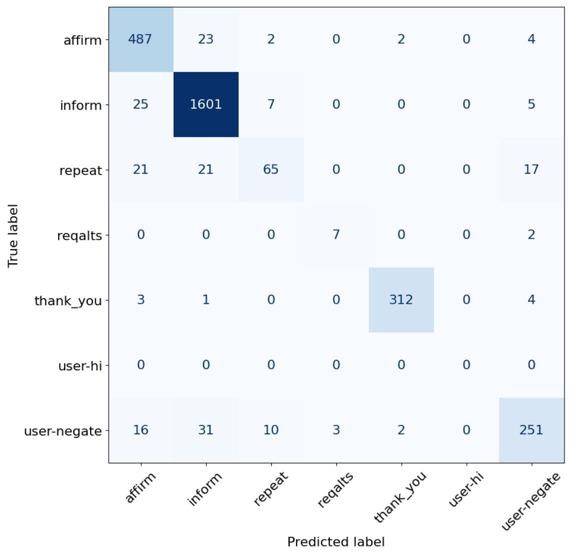
5.2.2 Limited Training Data

The results in this section are the basis for answering the research question **RQ2.2**: How does further pre-training affect performance when there is limited training data available?

Results of the experiments with 20% of the training data available can be found in Table 5.6, which also show improvements in micro and macro-F1 scores for TODA-UDAT when compared to U-DAT. When fine-tuning with the universal schema, improvements in micro-F1 score range between 2.55 and 12.33 in four out of six datasets. Macro-F1 scores improve between 3.24 and 24.9 in five datasets. When fine-tuning with the original schema, improvements in micro-F1 score are between 1.32 and 8.37 in four out of six datasets. Improvements in macro-F1 scores range from 1.08 to 10.7 in five datasets. When TODA-UDAT doesn't improve notably better than U-DAT, it performs about the same or just slightly better.



(a) U-DAT



(b) TODA-UDAT

Figure 5.6: Confusion matrix and F1 score for single-class samples after evaluation on M2M with the universal schema. 20% of the training data was used for fine-tuning.

Analysis

To get a closer look at the results on M2M with the universal schema, one can look at Figure 5.6 which shows confusion matrices with single-class samples such that confusion between classes can be observed in a single matrix. TODA-UDAT improves greatly on the small classes like *reqalts* and *repeat*. In U-DAT, some confusion between the classes *affirm* and *inform* can be observed. For example, one can see in Figure 5.6a that the model detects *inform* when the ground truth label is *affirm* 111 times. Many of these 111 wrongly detected utterances are on the form "Yes.", "That's correct.", or similar. The reason U-DAT fails on these simple utterances could be that half of the *affirm* utterances in the M2M training set are also labeled with *inform*. This can make it hard for the model to understand which part of the utterance is related to *affirm* and which part is related to *inform*. TODA-UDAT clears up most of this confusion, which can be seen in Figure 5.6b. TODA-UDAT doesn't wrongly classify the short simple *affirm* utterances as *inform*. Additional *inform* and *affirm* samples from the TODA corpus likely helps the pre-trained TODA-UDAT model better distinguish the two classes from each other.

Summary

As in the corresponding experiment with the BERT-based classifier, it was expected that larger improvements would be seen with 20% training data than with 100% training data, for the same reason. The results support this on both the universal and original schema with greater improvements in both micro and macro-F1 score compared to fine-tuning with 100% training data.

5.2.3 TODA-UDAT vs. TODA-BERT

Finally, results from TODA-UDAT and TODA-BERT are presented together for a direct comparison in Table 5.7.

Average micro-F1 scores are about two points higher with TODA-BERT than TODA-UDAT across all setups, while average macro-F1 scores range between 4.32 and 8.77 points higher.

Analysis

A likely explanation for TODA-BERTs superior performance over TODA-UDAT is that TODA-BERT builds on BERT which is already pre-trained in a self-supervised fashion for language understanding. On the other hand, TODA-UDAT is trained from scratch. This gives TODA-BERT a head start that's hard for TODA-UDAT to compete with. This could also be an explanation for why TODA-UDAT improves more on U-DAT than TODA-BERT does on BERT. Since BERT already is a very strong baseline, it's harder for TODA-BERT to improve than it is for TODA-UDAT. It remains to be seen if a context inclusive variant of TODA-BERT would improve performance more and if including context in supervised further pre-training is beneficial in this case.

Summary

All in all, TODA-BERT-filter is the best performing architecture when using the universal schema, while TODA-BERT-add performs best when using the original schema of a dataset. This is why we recommend TODA-BERT-filter and TODA-BERT-add for dialogue act classification depending on whether the label set follows the universal schema or not. Either way, the further pre-trained TODA-BERT model is used, just with small differences in the fine-tuning architecture.

Table 5.7: Micro-F1 and macro-F1 scores of TODA-BERT and TODA-UDAT with both 20% and 100% training data. TODA-BERT-filter is used for the universal schema, while TODA-BERT-add is used for the original schema. Scores are averages of five runs.

		20% Data				**100% Data**			
		TODA-BERT		**TODA-UDAT**		**TODA-BERT**		**TODA-UDAT**	
	Dataset	**micro**	**macro**	**micro**	**macro**	**micro**	**macro**	**micro**	**macro**
Universal schema	DSTC	97.55	**73.09**	95.60	62.98	**98.19**	**84.63**	97.09	70.82
	M2M	93.62	**89.41**	**93.66**	83.46	**95.61**	**93.05**	95.08	90.57
	Frames	**82.37**	**61.00**	75.33	48.81	**84.26**	**65.50**	76.51	51.44
	SGD	93.42	**91.30**	**93.73**	90.73	94.17	**92.44**	**94.63**	91.84
	E2E	**93.27**	**72.39**	92.05	69.48	**94.01**	74.55	93.52	**76.32**
	MWOZ	**97.43**	**88.80**	93.45	74.68	**97.96**	**90.45**	94.81	82.45
	Average	**92.94**	**79.33**	90.64	71.69	**94.03**	**83.44**	91.94	77.24
Original schema	DSTC	**95.41**	**57.55**	94.03	49.20	**98.15**	**81.13**	97.05	62.26
	M2M	**88.90**	**70.51**	85.85	60.73	**93.81**	**85.80**	93.50	77.43
	Frames	**68.13**	**24.76**	63.45	21.09	**78.09**	**51.25**	70.22	32.95
	SGD	92.11	89.12	**93.58**	**92.67**	92.87	90.31	**94.99**	**94.76**
	E2E	**89.45**	**43.65**	87.91	41.21	**90.89**	**48.44**	88.62	44.94
	MWOZ	**97.18**	**35.24**	92.39	30.00	**97.78**	**42.17**	94.21	34.14
	Average	**88.53**	**53.47**	86.20	49.15	**91.93**	**66.52**	89.77	57.75

Chapter 6

Conclusion

In this chapter, the thesis is concluded with the contribution of the thesis and further work.

6.1 Contribution

In this thesis, we aligned several task-oriented dialogue datasets to a universal dialogue act schema to enable supervised pre-training.

Architectures for pre-training and fine-tuning have been proposed and implemented. We made our further pre-trained BERT-based model, TODA-BERT, publicly available[1]

The final contribution of this thesis comes from insights gained through experimental comparison. The insight is summarized in the answers to our two main research questions:

RQ1 How does the performance of a further pre-trained BERT model compare to a BERT model that's only fine-tuned?

We find that further pre-trained BERT models often, but not always, perform better than a basic BERT model. This is observed in increased macro-F1 scores, indicating that further pre-training improves performance on small classes with few positive samples.

[1]`https://github.com/hsyver/TODA-BERT`

Further pre-training improves performance the most when there is limited training data available. However, this is only the case when fine-tuning with the universal schema and for four out of the six evaluation datasets. The two datasets that don't see improvements are still relatively large when limiting training data to 20% which supports that further pre-training helps the most in low resource situations. That improvements weren't observed when fine-tuning with original schemas could be explained by insufficient learning of the mapping from the universal schema to the original schema with limited data.

We find that the fine-tuning architectures that are as similar as possible to the pre-training architecture perform best.

Further pre-training with the TODA corpus doesn't improve performance on datasets that don't use a typical task-oriented dialogue act schema.

RQ2 How does the performance of a pre-trained dialogue context inclusive model compare to a model that's only fine-tuned?

Results show that the pre-trained context inclusive TODA-UDAT model consistently outperforms the non pre-trained U-DAT model by a large margin. TODA-UDAT improves the most when fine-tuning with the universal schema, but also improves when fine-tuning with original schemas.

The largest improvements are found when 20% of the training data is available, further supporting that the supervised pre-training is beneficial in low resource situations. Whether the improvement is mostly caused by context relationships learned during pre-training or just because U-DAT is a weak baseline compared to BERT is unclear.

6.2 Further Work

In this section, ideas for further work we didn't find time to pursue in this thesis are presented.

6.2.1 Supervised Pre-training for Intent Classification

While this thesis revolves supervised pre-training for dialogue act classification, it would be interesting to see what results could be made with the same approach for the task of intent classification. However, an obstacle for performing supervised pre-training for intent classification is that schemas of domain-specific intents vary greatly among datasets. A possible approach could be to unify schemas within domains before pre-training. Another approach could be to pre-train with all classes in the pre-training corpus, without aligning them.

6.2.2 Adapting Pre-trained Model to Original Schema

In this thesis pre-trained models were fine-tuned to the original schemas of target datasets with variations of a feed forward neural network at the end of the architecture. In the limited training data setup, TODA-BERT-add, which in the other setups had performed best for the original schemas, struggled with instability. Alternative approaches to adapt the pre-trained model to the original schema of a dataset could be explored to avoid the instability problem TODA-BERT-add has when there is limited training data.

6.2.3 Context Inclusive TODA-BERT

As stated in Sect. 5.2.3, it remains to be seen if further pre-training of a context inclusive variant of TODA-BERT would improve performance more. An idea for further work is therefore to implement a context inclusive variant of TODA-BERT and investigate if supervised further pre-training is beneficial in this case.

Bibliography

[1] W. A. Abro, G. Qi, Z. Ali, Y. Feng, and M. Aamir. Multi-turn intent determination and slot filling with neural networks and regular expressions. *Knowledge-Based Systems*, 208, 2020.

[2] A. H. Anderson, M. Bader, E. G. Bard, E. Boyle, G. Doherty, S. Garrod, S. Isard, J. Kowtko, J. McAllister, J. Miller, C. Sotillo, H. S. Thompson, and R. Weinert. The HCRC map task corpus. *Language and Speech*, 34(4):351–366, 1991.

[3] P. Budzianowski, T.-H. Wen, B.-H. Tseng, I. Casanueva, S. Ultes, O. Ramadan, and M. Gašić. MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, EMNLP '18, pages 5016–5026, 2018.

[4] Q. Chen, Z. Zhuo, and W. Wang. BERT for joint intent classification and slot filling. *CoRR*, abs/1902.10909, 2019.

[5] Y.-N. Chen, D. Hakkani-Tür, G. Tur, J. Gao, and L. Deng. End-to-end memory networks with knowledge carryover for multi-turn spoken language understanding. In *Interspeech 2016*, pages 3245–3249, 2016.

[6] A. Coucke, A. Saade, A. Ball, T. Bluche, A. Caulier, D. Leroy, C. Doumouro, T. Gisselbrecht, F. Caltagirone, T. Lavril, M. Primet, and J. Dureau. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *CoRR*, abs/1805.10190, 2018.

[7] J. Deriu, A. Rodrigo, A. Otegi, G. Echegoyen, S. Rosset, E. Agirre, and

M. Cieliebak. Survey on evaluation methods for dialogue systems. *Artificial Intelligence Review*, 54:755–810, 2020.

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, NAACL '19, pages 4171–4186, 2019.

[9] L. El Asri, H. Schulz, S. Sharma, J. Zumer, J. Harris, E. Fine, R. Mehrotra, and K. Suleman. Frames: a corpus for adding memory to goal-oriented dialogue systems. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, SIGDIAL '17, pages 207–219, 2017.

[10] M. Eric, R. Goel, S. Paul, A. Sethi, S. Agarwal, S. Gao, and D. Hakkani-Tür. Multiwoz 2.1: Multi-domain dialogue state corrections and state tracking baselines. *CoRR*, abs/1907.01669, 2019.

[11] J. Gao, M. Galley, and L. Li. Neural approaches to conversational AI. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, ACL '18, pages 2–7, 2018.

[12] S. Gururangan, A. Marasović, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey, and N. A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, ACL '20, pages 8342–8360, 2020.

[13] T. Han, X. Liu, R. Takanobu, Y. Lian, C. Huang, W. Peng, and M. Huang. Multiwoz 2.3: A multi-domain task-oriented dataset enhanced with annotation corrections and co-reference annotation. abs/2010.05594, 2020.

[14] C. T. Hemphill, J. J. Godfrey, and G. R. Doddington. The atis spoken language systems pilot corpus. In *Proceedings of the DARPA Speech and Natural Language Workshop*, DARPA '90, pages 96–101, 1990.

[15] M. Henderson, B. Thomson, and J. D. Williams. The second dialog state tracking challenge. In *Proceedings of the 15th annual meeting of the special interest group on discourse and dialogue*, SIGDIAL '14, pages 263–272, 2014.

[16] M. Henderson, B. Thomson, and J. D. Williams. The third dialog state tracking challenge. In *2014 IEEE Spoken Language Technology Workshop*, SLT '14, pages 324–329, 2014.

[17] M. Henderson, I. Casanueva, N. Mrkšić, P.-H. Su, T.-H. Wen, and I. Vulić. ConveRT: Efficient and accurate conversational representations from transformers. In *Findings of the Association for Computational Linguistics: EMNLP '20*, pages 2161–2174, 2020.

[18] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[19] D. Jurafsky, E. Shriberg, and D. Biasca. Switchboard SWBD-DAMSL shallow-discourse-function annotation coders manual, draft 13, 1997.

[20] O. Kuchaiev, J. Li, H. Nguyen, O. Hrinchuk, R. Leary, B. Ginsburg, S. Kriman, S. Beliaev, V. Lavrukhin, J. Cook, et al. NeMo: a toolkit for building AI applications using neural modules. abs/1909.09577, 2019.

[21] S. Larson, A. Mahendran, J. J. Peper, C. Clarke, A. Lee, P. Hill, J. K. Kummerfeld, K. Leach, M. A. Laurenzano, L. Tang, and J. Mars. An evaluation dataset for intent classification and out-of-scope prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, EMNLP-IJCNLP '19, pages 1311–1316, 2019.

[22] X. Li, S. Panda, J. Liu, and J. Gao. Microsoft dialogue challenge: Building end-to-end task-completion dialogue systems. *CoRR*, abs/1807.11125, 2018.

[23] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, ICLR '19, 2019.

[24] S. Mehri, M. Eric, and D. Hakkani-Tur. Dialoglue: A natural language understanding benchmark for task-oriented dialogue. abs/2009.13570, 2020.

[25] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin. Advances in pre-training distributed word representations. In *Proceedings of the Inter-*

national Conference on Language Resources and Evaluation, LREC '18, pages 52–55, 2018.

[26] A. Papangelis, M. Namazifar, C. Khatri, Y.-C. Wang, P. Molino, and G. Tür. Plato dialogue system: A flexible conversational ai research platform. *CoRR*, abs/2001.06463, 2020.

[27] S. Paul, R. Goel, and D. Hakkani-Tür. Towards universal dialogue act tagging for task-oriented dialogues. In *Interspeech 2019*, 2019.

[28] L. Qin, W. Che, M. Ni, Y. Li, and T. Liu. Knowing where to leverage: Context-aware graph convolutional network with an adaptive fusion layer for contextual spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:1280–1289, 2021.

[29] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019.

[30] A. Rastogi, X. Zang, S. Sunkara, R. Gupta, and P. Khaitan. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, AAAI '20, pages 8689–8696, 2020.

[31] R. Sarikaya, G. E. Hinton, and B. Ramabhadran. Deep belief nets for natural language call-routing. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing*, ICASSP '11, pages 5680–5683, 2011.

[32] S. Schuster, S. Gupta, R. Shah, and M. Lewis. Cross-lingual transfer learning for multilingual task oriented dialog. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, NAACL '19, pages 3795–3805, 2019.

[33] P. Shah, D. Hakkani-Tür, G. Tür, A. Rastogi, A. Bapna, N. Nayak, and L. P. Heck. Building a conversational agent overnight with dialogue self-play. *CoRR*, abs/1801.04871, 2018.

[34] M. Själander, M. Jahre, G. Tufte, and N. Reissmann. EPIC: An energy-efficient, high-performance GPGPU computing research infrastructure, 2019.

[35] S. Ultes, L. M. Rojas-Barahona, P.-H. Su, D. Vandyke, D. Kim, I. Casanueva, P. Budzianowski, N. Mrkšić, T.-H. Wen, M. Gašić, and S. Young. PyDial: A multi-domain statistical dialogue system toolkit. In *Proceedings of ACL 2017, System Demonstrations*, ACL '17, pages 73–78, 2017.

[36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, NIPS '17, pages 5998–6008, 2017.

[37] Y. Wang, Y. Shen, and H. Jin. A bi-model based RNN semantic frame parsing model for intent detection and slot filling. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, NAACL '18, pages 309–314, 2018.

[38] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, EMNLP '20, pages 38–45, 2020.

[39] C.-S. Wu, S. C. Hoi, R. Socher, and C. Xiong. TOD-BERT: Pre-trained natural language understanding for task-oriented dialogue. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, EMNLP '20, pages 917–929, 2020.

[40] P. S. Xingkun Liu, Arash Eshghi and V. Rieser. Benchmarking natural language understanding services for building conversational agents. In *Proceedings of the Tenth International Workshop on Spoken Dialogue Systems Technology*, IWSDS '19, pages 165–183, 2019.

[41] S. Young. CUED standard dialogue acts. *Report, Cambridge University Engineering Department, 14th October*, 2007.

[42] Q. Zhu, Z. Zhang, Y. Fang, X. Li, R. Takanobu, J. Li, B. Peng, J. Gao, X. Zhu, and M. Huang. ConvLab-2: An open-source toolkit for building, evaluating, and diagnosing dialogue systems. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, ACL '20, pages 142–149, 2020.