NLI Project Report

Bassotto, Cristian	Kormushev, Nikolay	Gheorghiu, Adrian	Cianci, Giuseppe
u210426	u234126	u234621	u234127

December 18, 2023

Contents

List of Tables

List of Figures

1 Introduction and Project Specification

This report explores the implementation of the core functionalities of a conversational agent, focusing on three key areas: domain identification and dialog act prediction, content extraction from user utterances, and agent move prediction. These components are crucial for creating an intelligent and responsive conversational agent.

In domain identification and dialog act prediction, our goal is to understand the context and intention behind user messages. For content extraction, we focus on semantic frame slot filling, which involves extracting key information from user input. The final task, agent move prediction, involves determining the agent's next action in a conversation. The next step would be to use Natural Language Generation techniques to produce the agent's responses. However, that is out of this project's scope.

As this project's goal is learning about different NLI techniques, we explore as many different approaches as possible.

2 Dataset Description

The dataset used for this project is MultiWOZ 2.2, an enhanced version of the original MultiWOZ dataset, which is a large-scale, multi-domain Wizard-of-Oz dataset for task-oriented dialogue modeling. The MultiWOZ dataset is a fully-labeled collection of human-human written conversations spanning multiple domains and topics, providing a robust foundation for natural language understanding and dialogue state tracking [Budzianowski et al., 2018, Zang et al., 2020].

The dataset encompasses dialogues from multiple domains, including hotels, restaurants, attractions, hospitals, police, taxis, and trains. For this project, our focus is narrowed to the hotel and restaurant reservation domains.

2.1 Data Collection Methodology

The MultiWOZ dataset was collected using the Wizard-of-Oz (WOZ) methodology [Budzianowski et al., 2018]. This approach simulates a dialogue system where one human (the "wizard") plays the role of the system or clerk, and another acts as the user.

Each dialogue is driven by a set of instructions specifying the user's goals, shared only with the user-side participant. The wizard tries to fulfill the request of the user. He is given a graphical user interface to query a database which contains information that the user asks for. This setup allows for the collection of natural, human-human dialogues at relatively low costs and with minimal time effort.

2.2 Annotation and Ontology

The ontology of the task-oriented dialogue system in MultiWOZ defines entity attributes (slots) and their possible values, divided into informable and requestable slots. Informable slots are attributes that users can specify to constrain their search (e.g., area, price range), while requestable slots are additional information that users can ask about an entity (e.g. phone number) [Budzianowski et al., 2018].

In addition to traditional dialogue state annotations, MultiWOZ 2.2 introduces slot span annotations for user and system utterances, active user intents, and requested slots for each user turn. This approach enhances the semantic representation of utterances and aids in developing more accurate models [Zang et al., 2020].

3 Task 1: Domain Identification/Dialog Act Prediction

3.1 Overview

The first task focuses on identifying the domain of the conversation, such as hotel or restaurant reservations, and predicting the dialog act or intention behind the user's message. The challenge lies in accurately classifying the dialog act, which can range from requests for information to confirmations or clarifications. This understanding is essential for the agent to navigate the conversation appropriately and provide relevant responses.
For this task, we evaluated 6 different model architectures. Furthermore, we tried different combinations of features and embeddings for some architectures.

1. **Extreme Gradient Boosting Model (XGBClassifier)**
2. **Support Vector Machine (SVM)**
3. **Multilayer Perceptron (MLP)**
4. **Long Short-Term Memory (LSTM)**
5. **Finetuning Pretrained Transformer Models (BERT)**
6. **Finetuning Pretrained Transformer Models (RoBERTa)**

3.2 Approach 1: Extreme Gradient Boosting Model (XGBClassifier)

Description

Extreme Gradient Boosting is an advanced implementation of the gradient boosting algorithm. Gradient boosting is a machine learning technique used for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models.

In gradient boosting, models are built in a stage-wise fashion, and individual models are added to correct the errors made by existing models. Extreme Gradient Boosting improves upon the standard gradient boosting technique by introducing a more regularized model formalization to control overfitting, leading to better performance [Chen and Guestrin, 2016].

We used XGBClassifier from the `xgboost` library which is an implementation of extreme gradient boosting for classification and `MultiOutputClassifier` from `scikit-learn` to fit one classifier for each class label.

Furthermore, we did some feature engineering and tried different combinations of features. In the end we selected the feature set highlighted in bold which had the best metrics on the validation set.

- Word unigram frequencies + Part of Speech (POS) unigram frequencies
  Accuracy: 63.3 %, Precision: 73.2 %, Recall: 52.8 %, F1-Score: 59.4 %
- Term Frequency-Inverse Document Frequency (TF-IDF)
  Accuracy: 65.7 %, Precision: 72.1 %, Recall: 54.5 %, F1-Score: 60.7 %
- TF-IDF + POS unigram frequencies
  Accuracy: 65.9 %, Precision: 73.9 %, Recall: 55.1 %, F1-Score: 61.6 %
- TF-IDF + Spacy embeddings
  Accuracy: 64.4 %, Precision: 69.5 %, Recall: 53.4 %, F1-Score: 59.2 %
- **TF-IDF + fastText embeddings**
  Accuracy: 76.8 %, Precision: 90.3 %, Recall: 78.9 %, F1-Score: 84.2 %

The final model uses 100 estimators with a maximum depth of 7, the 300 TF-IDF features with the best Analysis of Variance (ANOVA) score, and fastText embeddings.

In the following sections, only the metrics of the best performing model are displayed. All metrics were evaluated on the test set which is separate from the training and validation set.

Final Model Metrics

Class	Precision	Recall	F1-Score	Support
Hotel-Inform	0.924	0.822	0.870	1328
Hotel-Request	0.667	0.301	0.415	292
Restaurant-Inform	0.900	0.821	0.858	1322
Restaurant-Request	0.529	0.322	0.400	286
general-bye	0.977	0.942	0.959	225
general-greet	0.000	0.000	0.000	6
general-thank	0.972	0.941	0.956	693
other	0.911	0.816	0.861	2039
micro avg	0.903	0.789	0.842	6191
macro avg	0.735	0.621	0.665	6191
weighted avg	0.891	0.789	0.833	6191
samples avg	0.813	0.808	0.805	6191

Accuracy	0.768
Overall Precision (micro avg)	0.903
Overall Recall (micro avg)	0.789
Overall F1-Score (micro avg)	0.842
Label Ranking Avg Precision	0.824
Coverage Error	2.503 (worst: 8.000, best: 1.070)
Ranking Loss	0.196 (worst: 7.339, best: 0.000)

Table 1: Metrics Overview of XGBClassifier Model for Task 1

3.3 Approach 2: Support Vector Machine

Description

Support Vector Machine (SVM) is a supervised learning model used for classification and regression analysis. In the context of natural language processing, SVM provides an effective approach due to its ability to handle high-dimensional data and its robustness in model performance.

At its core, SVM works by finding a hyperplane in an `N`-dimensional space (where `N` is the number of features) that distinctly classifies the data points. For non-linearly separable data, SVM uses the kernel trick to transform the input space into a higher dimension where a hyperplane can be used for separation. This makes SVM particularly suited for text classification tasks, where data often involve complex and high-dimensional feature spaces [Cortes and Vapnik, 1995].

One of the key strengths of SVM in domain identification is its ability to provide clear margins of separation between different domains. This clear separation is crucial for accurately classifying utterances into their respective domains. Additionally, SVM's effectiveness in handling overfitting, ensures robust model performance.

We used the radial basis function kernel (RBF) because this non-linearly transforms the input space to a vector space of infinite dimension and is thus able to model complex relationships. Furthermore we tried different combinations of features like lemmatization and TF-IDF, as well as trying different

embeddings like fastText and selecting the top-k features based on the ANOVA score.

The final model uses an RBF kernel, the 300 best features according to the ANOVA score, and fastText embeddings.

Final Model Metrics

Class	Precision	Recall	F1-Score	Support
Hotel-Inform	0.933	0.803	0.863	1328
Hotel-Request	0.795	0.226	0.352	292
Restaurant-Inform	0.914	0.834	0.872	1322
Restaurant-Request	0.644	0.304	0.413	286
general-bye	0.962	0.911	0.936	225
general-greet	0.000	0.000	0.000	6
general-thank	0.957	0.935	0.946	693
other	0.958	0.789	0.865	2039
micro avg	0.931	0.772	0.844	6191
macro avg	0.771	0.600	0.656	6191
weighted avg	0.920	0.772	0.832	6191
samples avg	0.792	0.789	0.787	6191

Accuracy	0.755
Overall Precision (micro avg)	0.931
Overall Recall (micro avg)	0.772
Overall F1-Score (micro avg)	0.844
Label Ranking Avg Precision	0.809
Coverage Error	2.615 (worst: 8.000, best: 1.070)
Ranking Loss	0.214 (worst: 7.339, best: 0.000)

Table 2: Metrics Overview of SVM Model for Task 1

3.4 Approach 3: Multilayer Perceptron

Description

The Multilayer Perceptron (MLP) is a class of feedforward neural network, widely used in machine learning tasks for its ability to capture complex, non-linear patterns in data.

A MLP consists of at least three layers of nodes: an input layer, one or more hidden layers, and an output layer. Each node, or neuron, in these layers is connected with a certain weight and uses non-linear activation functions, allowing the MLP to model intricate relationships in data [Pal and Mitra, 1992].

In the architecture of our MLP, we incorporated Batch Normalization (BatchNorm) and Dropout layers to enhance the model's performance. BatchNorm stabilizes the learning process by normalizing the input of each layer, reducing the internal covariate shift. Meanwhile, Dropout layers mitigate overfitting by randomly setting a fraction of input units to zero during training, increasing the robustness of neurons. These additions help in accelerating training and improving the model's ability to generalize to new data.

Like for the SVM, we tried different combinations of TF-IDF, embeddings, and top-k feature selection based on ANOVA score.

The final model uses fastText embeddings and the 300 best TF-IDF features according to the ANOVA score.

Final Model Metrics

Class	Precision	Recall	F1-Score	Support
Hotel-Inform	0.934	0.847	0.889	1328
Hotel-Request	0.764	0.387	0.514	292
Restaurant-Inform	0.934	0.831	0.879	1322
Restaurant-Request	0.647	0.385	0.482	286
general-bye	0.982	0.973	0.978	225
general-greet	0.500	0.667	0.571	6
general-thank	0.953	0.974	0.964	693
other	0.937	0.828	0.879	2039
micro avg	0.926	0.813	0.866	6191
macro avg	0.832	0.736	0.770	6191
weighted avg	0.917	0.813	0.859	6191
samples avg	0.842	0.831	0.833	6191

Accuracy	0.803
Overall Precision (micro avg)	0.926
Overall Recall (micro avg)	0.813
Overall F1-Score (micro avg)	0.866
Label Ranking Avg Precision	0.931
Coverage Error	1.278 (worst: 8.000, best: 1.070)
Ranking Loss	0.029 (worst: 7.339, best: 0.000)

Table 3: Metrics Overview of MLP Model for Task 1

3.5 Approach 4: LSTM

Description

Long Short-Term Memory (LSTM) networks, a type of recurrent neural network (RNN), are highly effective in processing sequential data, making them well-suited for natural language processing tasks. They excel due to their ability to remember dependencies, an essential feature for understanding the context in conversations.

Unlike traditional RNNs, LSTMs have a special gating mechanism that controls the flow of information. This mechanism includes three types of gates: input, forget, and output gates, which help the network to retain important information over long sequences while forgetting the irrelevant data. This capability is crucial in text processing where the context and meaning can depend on information earlier in the sequence [Hochreiter and Schmidhuber, 1997].

LSTMs are particularly effective in tasks where the sequence and order of words are critical for understanding the meaning, as in the case of domain identification in dialogue systems. Their ability to maintain context over long conversations without losing track of earlier inputs makes them a robust choice for classifying and analyzing conversational data.

Our implementation utilizes a stacked bidirectional LSTM architecture, combined with a loss weighting strategy to address class imbalance. We experimented with various configurations to optimize performance. The combination with the best performance metrics is highlighted in bold.

- BERT embeddings, with/without lemmatization (without was better)
  Accuracy: 77.7 %, Precision: 91.9 %, Recall: 70.4 %, F1-Score: 77.7 %
- fastText embeddings, with/without lemmatization (without was better)
  Accuracy: 77.7 %, Precision: 90.2 %, Recall: 69.2 %, F1-Score: 75.1 %
- **BERT embeddings, supplemented with utterance and dialog act history**
  Accuracy: 87.5 %, Precision: 93.4 %, Recall: 90.2 %, F1-Score: 91.8 %

The final LSTM model uses BERT embeddings as well as utterance and dialog act history.

Final Model Metrics

Class	Precision	Recall	F1-Score	Support
Hotel-Inform	0.910	0.947	0.928	1328
Hotel-Request	0.809	0.771	0.789	292
Restaurant-Inform	0.930	0.919	0.924	1322
Restaurant-Request	0.809	0.815	0.812	286
general-bye	0.991	0.991	0.991	225
general-greet	0.800	0.667	0.727	6
general-thank	0.937	0.986	0.961	693
other	0.989	0.854	0.917	2039
micro avg	0.934	0.902	0.918	6191
macro avg	0.897	0.869	0.881	6191
weighted avg	0.937	0.902	0.917	6191
samples avg	0.929	0.920	0.919	6191

Accuracy	0.875
Overall Precision (micro avg)	0.934
Overall Recall (micro avg)	0.902
Overall F1-Score (micro avg)	0.918
Label Ranking Avg Precision	0.966
Coverage Error	1.164 (worst: 8.000, best: 1.070)
Ranking Loss	0.013 (worst: 7.339, best: 0.000)

Table 4: Metrics Overview of LSTM Model for Task 1

3.6 Approach 5: Finetuning Pretrained Transformer Models (BERT)

Description

Fine-tuning Pretrained Transformer Models, is a state-of-the-art approach in natural language processing tasks. BERT (Bidirectional Encoder Representations from Transformers) revolutionized the NLP field with its deep bidirectional training and contextual word representations [Devlin et al., 2018].

BERT uses the transformer architecture, which allows it to capture the context from both directions (left and right of a word in a sentence), providing a more nuanced understanding of language. This capability is crucial for tasks like domain identification in conversational agents, where understanding context and the subtleties of language is key. Furthermore, the transformer architecture solves some problems of RNNs, like the vanishing gradient problem and it also enables parallelization.

Finetuning a pretrained BERT model involves adjusting the model's parameters slightly on a specific dataset or task. This approach is highly effective because it leverages a model already trained on vast amounts of data, encompassing a broad linguistic understanding, and adapts it to the specific nuances of the conversation domain.

No additional feature engineering was done for the final model. The model does not use utterance and dialog act history either.

Final Model Metrics

Class	Precision	Recall	F1-Score	Support
Hotel-Inform	0.877	0.909	0.893	1328
Hotel-Request	0.795	0.425	0.554	292
Restaurant-Inform	0.935	0.872	0.902	1323
Restaurant-Request	0.680	0.462	0.550	286
general-bye	0.993	1.000	0.997	293
general-greet	0.833	0.833	0.833	6
general-thank	0.982	0.969	0.975	940
other	0.942	0.916	0.929	3307
micro avg	0.926	0.882	0.903	7775
macro avg	0.880	0.798	0.829	7775
weighted avg	0.921	0.882	0.898	7775
samples avg	0.892	0.892	0.889	7775

Accuracy	0.861
Overall Precision (micro avg)	0.926
Overall Recall (micro avg)	0.882
Overall F1-Score (micro avg)	0.903
Label Ranking Avg Precision	0.950
Coverage Error	1.200 (worst: 8.000, best: 1.055)
Ranking Loss	0.020 (worst: 7.266, best: 0.000)

Table 5: Metrics Overview of BERT Model for Task 1

3.7 Approach 6: Finetuning Pretrained Transformer Models (RoBERTa)

Description

RoBERTa builds upon the achievements of BERT . Initially introduced by Liu et al. in 2019, RoBERTa modifies key hyperparameters in BERT, leading to improved performance across a range of NLP tasks [Liu et al., 2019].
RoBERTa (Robustly Optimized BERT Approach) refines the BERT model by training on larger batches and learning rates, removing the next sentence prediction objective, and training with much larger mini-batches and longer sequences. This optimization enables RoBERTa to outperform BERT on various benchmarking tasks [Liu et al., 2019].

Finetuning RoBERTa involves adapting its pretrained parameters to specific datasets or tasks, just like finetuning the parameters of a pretrained BERT model.

Compared to the previous BERT model, the final RoBERTa model uses utterance and dialog act history as well. This enhances its performance even further. Out of all evaluated approaches, this model has the best metrics - they are almost perfect.

Final Model Metrics

Class	Precision	Recall	F1-Score	Support
Hotel-Inform	0.998	0.986	0.992	1328
Hotel-Request	0.975	0.945	0.960	292
Restaurant-Inform	0.988	0.998	0.993	1323
Restaurant-Request	1.000	0.930	0.964	286
general-bye	0.997	1.000	0.998	293
general-greet	0.000	0.000	0.000	6
general-thank	0.998	0.994	0.996	940
other	0.998	0.982	0.990	3307
micro avg	0.995	0.983	0.989	7775
macro avg	0.869	0.854	0.862	7775
weighted avg	0.995	0.983	0.989	7775
samples avg	0.996	0.990	0.992	7775

Accuracy	0.982
Overall Precision (micro avg)	0.995
Overall Recall (micro avg)	0.983
Overall F1-Score (micro avg)	0.989
Label Ranking Avg Precision	0.997
Coverage Error	1.067 (worst: 8.000, best: 1.055)
Ranking Loss	0.002 (worst: 7.266, best: 0.000)

Table 6: Metrics Overview of RoBERTa Model for Task 1

4 Task 2: Semantic Frame Slot Filling

4.1 Task Overview

Semantic frame slot filling involves extracting and categorizing specific pieces of information from user utterances. This task is crucial for understanding the details of the user's request, such as the type of cuisine they prefer or the dates for a hotel booking. Effective slot filling enables the conversational agent to gather all necessary details to fulfill the user's request or to ask follow-up questions if some information is missing.

We have split this task into three subtasks: 1. BIO Tagging (Beginning, Inside, Outside), 2. Question Labeling, and 3. Slot Value Normalization.

4.2 Subtask 1: BIO Tagging

The first subtask in slot filling involves BIO tagging, aimed at identifying the actual values of slots in a user's utterance. This is a sequence-to-sequence multiclass classification problem, where each word in the user's utterance is tagged with exactly one label indicating whether it's the beginning (B), inside (I), or outside (O) of a slot. This approach is crucial for extracting structured information from unstructured text.

For this task, we tried the following approaches where we finetuned transformer models with different preprocessing of the data:

- Finetuning BERT for Token Classification
  punctuation removal, splitting on spaces and then running the tokenizer on every split word
  Accuracy: 98.1 %, Precision: 91.3 %, Recall: 95.1 %, F1-Score: 93.2 %
- Finetuning RoBERTa for Token Classification
  balance loss per class based on effective number of samples (combat class imbalance), running the tokenizer on the whole utterance
  Accuracy: 97.1 %, Precision: 76.6 %, Recall: 96.8 %, F1-Score: 84.2 %
- **Finetuning RoBERTa for Token Classification**
  removing stop words, adding history, lemmatization, running the tokenizer on the whole utterance
  Accuracy: 99.9 %, Precision: 98.0 %, Recall: 99.4 %, F1-Score: 98.7 %

The following figures show the column-wise and row-wise normalized confusion matrices of the last RoBERTa model which performs the best.

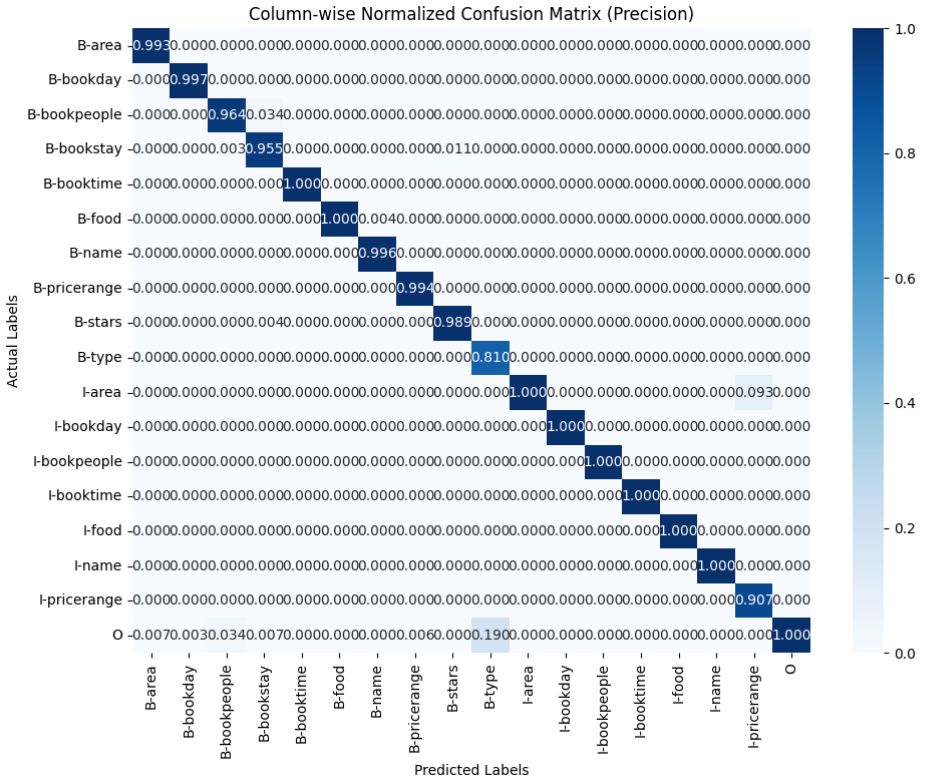

Figure 1: Column-wise Normalized Confusion Matrix (Precision) of the RoBERTa Model for Task 2.1

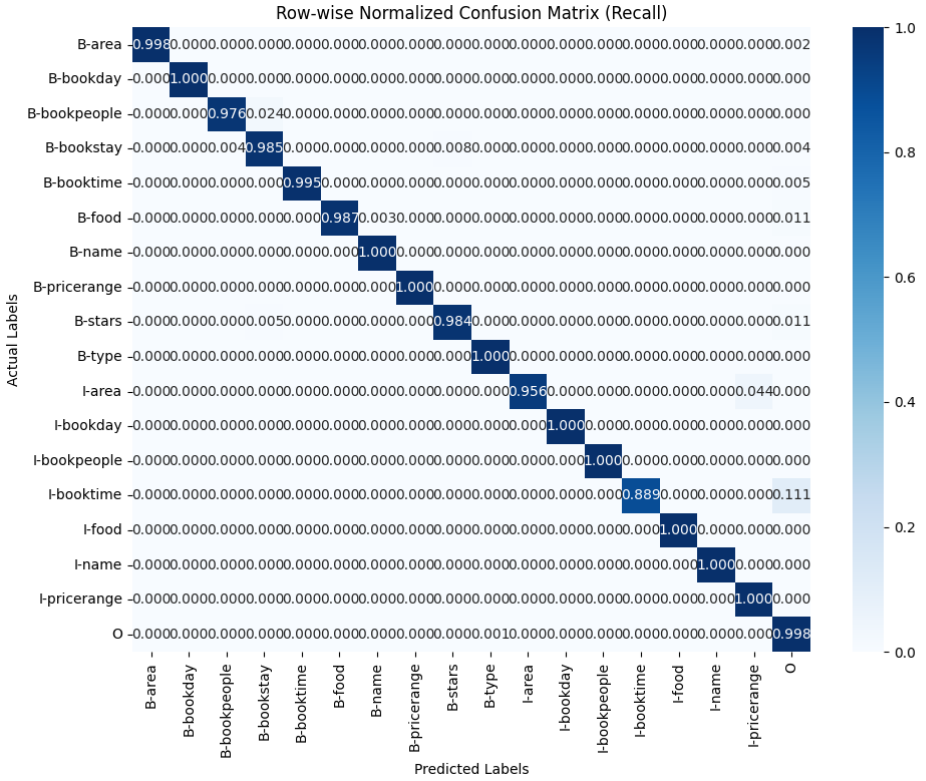

Figure 2: Row-wise Normalized Confusion Matrix (Recall) of the RoBERTa Model for Task 2.1

4.3 Subtask 2: Question Labeling

The second subtask involves labeling questions in user utterances which is a multi-label classification task, since each utterance may contain multiple slot queries. For instance, a user may inquire about both the phone number and cuisine of a restaurant in a single utterance.

We tried the following approaches for this task. Across all models, the same preprocessing as before was used.

- LSTM with history
  Accuracy: 92.4 %, Precision: 87.7 %, Recall: 76.8 %, F1-Score: 81.9 %
- GPT-2 (Generative Pre-trained Transformer 2) with history
  Accuracy: 96.7 %, Precision: 77.6 %, Recall: 93.3 %, F1-Score: 83.6 %
- ConvBERT (Convolutional BERT) with history
  Accuracy: 97.6 %, Precision: 84.1 %, Recall: 97.9 %, F1-Score: 89.9 %
- **RoBERTa with history**
  Accuracy: 83.5 %, Precision: 93.3 %, Recall: 89.1 %, F1-Score: 91.2 %

Judging by the F1-Score, the RoBERTa model performs best. The following table shows more detailed performance metrics of this model.

Final Model Metrics

Class	Precision	Recall	F1-Score	Support
hotel-address	0.915	0.796	0.851	54
hotel-area	0.952	0.741	0.833	27
hotel-internet	0.933	0.875	0.903	32
hotel-name	0.917	0.815	0.863	27
hotel-parking	0.963	0.839	0.897	31
hotel-phone	0.947	0.973	0.959	73
hotel-postcode	0.917	1.000	0.957	55
hotel-pricerange	1.000	0.765	0.867	34
hotel-ref	0.978	1.000	0.989	44
hotel-stars	0.000	0.000	0.000	7
hotel-type	0.857	0.462	0.600	13
restaurant-address	0.914	0.950	0.932	101
restaurant-area	0.750	0.643	0.692	14
restaurant-food	0.889	0.800	0.842	20
restaurant-name	0.917	0.880	0.898	25
restaurant-phone	0.894	0.974	0.933	78
restaurant-postcode	0.941	0.941	0.941	85
restaurant-pricerange	1.000	0.684	0.813	19
restaurant-ref	1.000	0.983	0.992	60
micro avg	0.933	0.891	0.912	799
macro avg	0.878	0.796	0.830	799
weighted avg	0.926	0.891	0.904	799
samples avg	0.926	0.914	0.913	799

Accuracy	0.835
Overall Precision (micro avg)	0.933
Overall Recall (micro avg)	0.891
Overall F1-Score (micro avg)	0.912
Label Ranking Avg Precision	0.959
Coverage Error	1.641 (worst: 19.000, best: 1.399)
Ranking Loss	0.011 (worst: 24.182, best: 0.000)

Table 7: Metrics Overview of RoBERTa Model for Task 2.2

4.4 Subtask 3: Slot Value Normalization

The final subtask, albeit smaller in scope, is crucial for data consistency. It involves mapping diverse slot values to unified forms. For example the same entity (e.g. the number one) can be referred to by multiple names (e.g. `one`, `One`, `1`, or misspellings like `onne`).

To solve this subtask, we use a rule-based approach using regular expressions. This process ensures uniformity in slot values, which improves the performance of subsequent models in the pipeline and thus facilitates more accurate and coherent interactions by the conversational agent.

5 Task 3: Agent Move Prediction

5.1 Task Overview

Agent move prediction involves determining the most appropriate next step for the conversational agent in the dialog. This could include deciding whether to ask for more information, make a recommendation, or confirm details with the user. The success of this task depends on the agent's ability to understand the current state of the conversation, what information it has gathered so far, and what information is still needed to conclude the dialog successfully.

This task is split into three sequential subtasks which are all multi-label classification problems: 1. Prediction of Slots to be Retrieved, 2. Agent Dialog Act Prediction, and 3. Prediction of Slots to be Requested.

5.2 Subtask 1: Prediction of Slots to be Retrieved

The initial subtask involves determining the slots that the agent needs to retrieve from the database to present to the user. This is a predictive task where the agent, based on the current dialog context and the user's requests, identifies relevant slots to fetch from the database. Similar to the BIO tagging in Task 2, this is a sequence prediction problem, though the focus here is on database interaction rather than user utterance analysis.

We tested the following models for this task:

- Dictionary based model
  This uses a dictionary to store training data and makes predictions by directly looking up input data in this dictionary, returning either the corresponding output data or a zero vector if the input data was not seen during training
  Accuracy: 52.4 %, Precision: 33.5 %, Recall: 26.1 %, F1-Score: 29.4 %
- LSTM
  Accuracy: 29.9 %, Precision: 23.5 %, Recall: 8.2 %, F1-Score: 10.8 %
- **RoBERTa**
  Accuracy: 46.7 %, Precision: 79.4 %, Recall: 56.7 %, F1-Score: 66.2 %
- Classical models using indicator features
  - SVM
    Accuracy: 59.7 %, Precision: 78.8 %, Recall: 19.0 %, F1-Score: 30.6 %
  - XGBClassifier
    Accuracy: 59.5 %, Precision: 75.3 %, Recall: 19.8 %, F1-Score: 31.3 %
  - MLP
    Accuracy: 58.1 %, Precision: 33.3 %, Recall: 8.6 %, F1-Score: 15.5 %
- Classical models using BERT embeddings averaged out as features
  - SVM
    Accuracy: 41.7 %, Precision: 0.00 %, Recall: 0.00 %, F1-Score: 0.00 %
  - XGBClassifier
    Accuracy: 43.1 %, Precision: 53.6 %, Recall: 9.2 %, F1-Score: 15.0 %
  - MLP
    Accuracy: 41.9 %, Precision: 13.8 %, Recall: 0.4 %, F1-Score: 0.8 %

Final Model Metrics

Class	Precision	Recall	F1-Score	Support
hotel-address	0.867	0.348	0.497	112
hotel-area	0.583	0.135	0.219	259
hotel-availability	0.869	0.899	0.883	978
hotel-internet	0.722	0.163	0.265	160
hotel-name	0.543	0.300	0.387	466
hotel-parking	0.881	0.234	0.370	158
hotel-phone	0.887	0.685	0.773	92
hotel-postcode	0.808	0.545	0.651	77
hotel-pricerange	0.000	0.000	0.000	253
hotel-ref	0.000	0.000	0.000	2
hotel-stars	1.000	0.014	0.028	214
hotel-type	0.818	0.027	0.051	339
none	0.876	0.824	0.849	1421
restaurant-address	0.844	0.488	0.618	166
restaurant-area	0.538	0.179	0.269	318
restaurant-availability	0.870	0.919	0.894	919
restaurant-food	0.495	0.457	0.475	346
restaurant-name	0.618	0.715	0.663	442
restaurant-phone	0.897	0.788	0.839	99
restaurant-postcode	0.893	0.758	0.820	99
restaurant-pricerange	0.629	0.085	0.150	258
restaurant-ref	0.000	0.000	0.000	5
micro avg	0.794	0.567	0.662	7183
macro avg	0.665	0.389	0.441	7183
weighted avg	0.749	0.567	0.596	7183
samples avg	0.819	0.675	0.710	7183

Accuracy	0.467
Overall Precision (micro avg)	0.794
Overall Recall (micro avg)	0.567
Overall F1-Score (micro avg)	0.662
Label Ranking Avg Precision	0.859
Coverage Error	3.410 (worst: 22.000, best: 2.166)
Ranking Loss	0.041 (worst: 40.758, best: 0.000)

Table 8: Metrics Overview of RoBERTa Model for Task 3.1

5.3 Subtask 2: Agent Dialog Act Prediction

The second subtask is the prediction of the agent's dialog act, such as making a recommendation or asking for clarification. This involves classifying the agent's potential responses based on the dialog's progression, similar to the question labeling in Task 2. The challenge lies in accurately predicting the most contextually appropriate and informative action for the agent to take. Like the other subtasks, this is a multi-label classification problem, so the agent can take multiple actions in one utterance.

For this task, we evaluated the following models:

- LSTM
  Accuracy: 49.1 %, Precision: 82.9 %, Recall: 65.6 %, F1-Score: 73.2 %
- BERT
  Accuracy: 29.3 %, Precision: 75.2 %, Recall: 55.4 %, F1-Score: 63.8 %

- **RoBERTa**
  Accuracy: 55.0 %, Precision: 83.6 %, Recall: 68.3 %, F1-Score: 75.2 %

Final Model Metrics

5.4 Subtask 3: Prediction of Slots to be Requested

The final subtask centers on identifying which slots the agent should request from the user in its next dialog act. For instance, when tasked with making a restaurant reservation, the agent needs to ascertain details like the number of people, date, and time. This subtask is somewhat analogous to the slot value normalization in Task 2, where the focus was on unifying diverse slot values for consistency. Here, the emphasis is on anticipating the user's information needs based on the current dialog state. We evaluated the following models for this task:

- LSTM
  Accuracy: 68.5 %, Precision: 71.3 %, Recall: 63.2 %, F1-Score: 65.0 %
- **BERT**
  balance loss per class based on effective number of samples (combat class imbalance)
  Accuracy: 71.4 %, Precision: 87.1%, Recall: 69.1 %, F1-Score: 77.1 %
- RoBERTa
  Accuracy: 41.1 %, Precision: 71.5 %, Recall: 59.3 %, F1-Score: 64.8 %

Thanks to the balancing of the loss based on the number of samples per class, the BERT model performs better than the RoBERTa model for which we did not use this loss balancing during training. This is because the `NOTHING` class has a dominating size: 18869 instances were of this class in the training set. The following table shows the corresponding weights for class size based loss.

Class Name	Class Size	Class Weight
NOTHING	18869	0.002
booking-bookday	52	0.042
booking-bookpeople	46	0.047
booking-bookstay	45	0.049
booking-booktime	46	0.047
hotel-area	1517	0.003
hotel-bookday	697	0.004
hotel-bookpeople	385	0.007
hotel-bookstay	573	0.005
hotel-booktime	24	0.09
hotel-internet	156	0.015
hotel-name	61	0.036
hotel-parking	204	0.012
hotel-pricerange	981	0.003
hotel-stars	301	0.008
hotel-type	193	0.012
restaurant-area	818	0.004
restaurant-bookday	402	0.006
restaurant-bookpeople	231	0.01
restaurant-bookstay	4	0.535
restaurant-booktime	558	0.005
restaurant-food	1339	0.003
restaurant-name	44	0.05
restaurant-pricerange	669	0.004

Table 9: Balancing of the Loss based on Class Size

Final Model Metrics

Class	Precision	Recall	F1-Score	Support
NOTHING	0.932	0.913	0.923	2168
booking-bookday	0.000	0.000	0.000	5
booking-bookpeople	0.000	0.000	0.000	3
booking-bookstay	0.000	0.000	0.000	0
booking-booktime	0.000	0.000	0.000	5
hotel-area	0.630	0.690	0.659	168
hotel-bookday	0.600	0.464	0.523	97
hotel-bookpeople	0.516	0.291	0.372	55
hotel-bookstay	0.636	0.298	0.406	94
hotel-booktime	0.000	0.000	0.000	0
hotel-internet	1.000	0.059	0.111	17
hotel-name	1.000	0.600	0.750	5
hotel-parking	1.000	0.048	0.091	21
hotel-pricerange	0.222	0.018	0.033	111
hotel-stars	0.000	0.000	0.000	38
hotel-type	0.000	0.000	0.000	24
restaurant-area	0.444	0.086	0.144	93
restaurant-bookday	0.778	0.175	0.286	80
restaurant-bookpeople	0.444	0.087	0.145	46
restaurant-bookstay	0.000	0.000	0.000	0
restaurant-booktime	0.679	0.190	0.297	100
restaurant-food	0.840	0.527	0.647	169
restaurant-name	0.375	0.750	0.500	4
restaurant-pricerange	0.458	0.136	0.210	81
micro avg	0.871	0.691	0.771	3384
macro avg	0.440	0.222	0.254	3384
weighted avg	0.802	0.691	0.719	3384
samples avg	0.751	0.743	0.744	3384

Accuracy	0.714
Overall Precision (micro avg)	0.871
Overall Recall (micro avg)	0.691
Overall F1-Score (micro avg)	0.771
Label Ranking Avg Precision	0.887
Coverage Error	1.621 (worst: 24.000, best: 1.122)
Ranking Loss	0.020 (worst: 25.500, best: 0.000)

Table 10: Metrics Overview of BERT Model for Task 3.3

References

[Budzianowski et al., 2018] Budzianowski, P., Wen, T.-H., Tseng, B.-H., Casanueva, I., Ultes, S., Ramadan, O., and Gašić, M. (2018). MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In Riloff, E., Chiang, D., Hockenmaier, J., and Tsujii, J., editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics.

[Chen and Guestrin, 2016] Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754.

[Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. (1995). Support vector networks. *Machine Learning*, 20:273–297.

[Devlin et al., 2018] Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

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

[Liu et al., 2019] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

[Pal and Mitra, 1992] Pal, S. and Mitra, S. (1992). Multilayer perceptron, fuzzy sets, and classification. *IEEE Transactions on Neural Networks*, 3(5):683–697.

[Zang et al., 2020] Zang, X., Rastogi, A., Sunkara, S., Gupta, R., Zhang, J., and Chen, J. (2020). MultiWOZ 2.2 : A dialogue dataset with additional annotation corrections and state tracking baselines. In Wen, T.-H., Celikyilmaz, A., Yu, Z., Papangelis, A., Eric, M., Kumar, A., Casanueva, I., and Shah, R., editors, *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 109–117. Association for Computational Linguistics.