Evaluating ChatGPT for Joint Intent Detection and Slot Filling: Zero-Shot vs. Few-Shot Prompting

Hannah Fong
De La Salle University
Manila, Philippines
hannah_regine_fong@dlsu.edu.ph

Ethel Ong
De La Salle University
Manila, Philippines
ethel.ong@dlsu.edu.ph

ABSTRACT

Natural language understanding transforms an unstructured text into a structured semantic representation through intent detection (ID) and slot filling (SF). Research shows that jointly performing the two tasks results in better performance due to the interdependencies between intent classes and slot labels. The current state-of-the-art joint ID and SF models are based on the pre-trained language model BERT. This raises the question of how large-scale pre-trained language models (LLMs) would perform on the joint task. LLMs are reported to achieve competitive performance in zero-shot prompting, and better in few-shot prompting. In this paper, we report our evaluation of ChatGPT's performance on the joint ID and SF task under zero-shot and different few-shot (i.e., 1-shot, 5-shot, 10-shot, 20-shot, 30-shot) prompt settings. Results showed that ChatGPT's performance generally improves when given more examples in the prompt. However, its performance lags behind other LLMs and SOTA joint models. It also has the tendency to exhibit unexpected behaviors such as generating unknown outputs, outputting incorrect number of slot labels, and annotating a different utterance from what was provided. These unexpected behaviors were observed more frequently when sending longer prompts, demonstrating the effect of prompt length to ChatGPT's performance.

KEYWORDS

joint intent detection and slot filling, large language models, ChatGPT, zero-shot prompting, few-shot prompting

1 INTRODUCTION

Natural language understanding (NLU) is the key component in a conversational agent architecture that transforms an unstructured input in human language into a structured semantic representation that can be processed by a machine [14]. It performs two important tasks: intent detection and slot filling [31]. Intent detection (ID) is a classification task that maps an utterance to a finite set of intent classes that signifies the intention or goal of that particular utterance. Slot filling (SF) is a sequence labeling task that attaches labels to the sequence of tokens in an utterance. Traditionally, these two tasks were treated independently and performed sequentially. However, it was discovered that the intent class and slot labels of an utterance have interdependencies that, when modelled together, can result in better performance [31]. This led research work that investigates the creation of joint ID and SF models to perform the two tasks simultaneously, leading to better performance compared to the traditional independent models.

The earliest approach to joint ID and SF models utilized statistical methods such as conditional random fields and hidden Markov models [2, 11]. These models outperformed the independent ID and SF models. With the emergence of deep learning, neural network architectures, such as convolutional neural networks (CNN) and recurrent neural networks (RNN), were applied to solve the joint task [15, 32]. In 2017, the transformer neural network architecture was introduced and became the building block of large-scale language models that can understand and generate human language [16, 28]. Bidirectional Encoder Representations from Transformers (BERT) is a transformer-based language model that produced state-of-the-art (SOTA) results in various natural language processing (NLP) tasks upon its release [6]. This prompted the implementation of BERT-based joint ID and SF models which currently exhibit SOTA performance on the joint task [27].

Recently, ChatGPT which is a conversational LLM fine-tuned from the generative pre-trained transformer (GPT) 3.5, has shown strong performance on various NLP tasks [25], including zero-shot performance on the joint ID and SF task [21]. A detailed analysis of the results revealed that ChatGPT performs relatively well in ID but poorly in SF. The authors experimented with adding slot descriptions and examples in the prompts and found these to improve the performance of ChatGPT in the SF task for the benchmark SNIPS NLU dataset [5]. However, the details of the experiment, such as the number of examples provided, were not documented.

In this paper, we report our systematic evaluation of ChatGPT's performance on the joint ID and SF task across zero-shot and different few-shot prompt settings for the Airline Travel Information System (ATIS) dataset, which is another widely used NLU benchmark dataset [21]. Results showed that ChatGPT's performance generally improves when given more examples in the prompt. However, its performance lags behind other LLMs and SOTA joint models. It also has a tendency to exhibit unexpected behaviors such as generating unknown outputs, outputting incorrect number of slot labels, and annotating a different utterance from what was provided. Our methodology and findings can serve as a reference for future works that intend to evaluate ChatGPT's performance on NLU tasks with varying prompt settings.

2 RELATED WORKS

In this section, we provide a review of early works and recent approaches in joint intent detection and slot filling, including the use of large language models.

2.1 Early Approaches in Joint ID and SF

The earliest work on the joint ID and SF task devised triangular-chain conditional random fields (TriCRFs) that jointly represent the ID and SF in a single graphical model, thereby naturally capturing the relationship between intent classes and slot labels [11]. Their proposed approach achieved higher F1-scores for both ID and SF over independent, cascaded, and reranked approaches. However, this approach requires a large amount of labeled training data, which is expensive to obtain [30]. Wang [30] showed that limited training data degrades the joint model's ID performance.

To address the problem of scarce labeled training data, Celikyilmaz and Hakkani-Tur [2] proposed leveraging both labeled and unlabeled data to jointly learn the intent classes and slot labels of utterances. Their approach is a semi-supervised generative multi-layer context model where an utterance is represented using hidden Markov models. Prior knowledge extracted from unlabeled web search queries are utilized to enhance the semantic component extraction from utterances. With this, a higher ID accuracy compared to the cascaded and TriCRFs approaches is achieved when the amount of labeled training data is small (i.e., 25% of the training data). However, it underperformed in SF, measured using the F1-score, compared to the two baseline approaches.

2.2 Deep Learning Models for Joint ID and SF

The emergence of deep learning prompted Xu and Sarikaya [32] to apply the neural network architecture in performing the joint task. They proposed a CNN-based TriCRFs model where features are automatically extracted from the utterances by the convolutional layers that are shared between the two tasks. The intent and slot dependencies are jointly modeled using TriCRFs. The proposed model outperformed the independent and TriCRFs approaches by achieving a lower ID error rate and higher SF F1-score.

The strong sequence modeling performance of RNNs prompted Liu and Lane [15] to use it for the joint ID and SF task in an encoder-decoder architecture. A bidirectional RNN encoder is shared by two unidirectional RNN decoders for ID and SF, respectively. Both RNNs use the Long Short-Term Memory (LSTM) cell as the basic RNN unit to model long-term dependencies. The proposed model achieved lower ID error rate and higher SF F1-score compared to the previous joint models (i.e., TriCRFs and CNN TriCRFs).

The deep learning-based models demonstrated strong performance on the joint ID and SF task. However, these require a vast amount of labeled training data which are time-consuming and costly to obtain [27]. The scarcity of labeled data diminishes the generalizability of these models. As an alternative, language models pre-trained on a large corpora of unlabeled text were developed to learn the linguistic patterns of a language [16]. Transformers are used for their efficient parallel processing capability [29]. One example is BERT, which has achieved SOTA performance in many NLP tasks. Shafi and Chachoo [27] investigated how BERT can improve the generalizability of the joint ID and SF model. They proposed a multi-stage framework trained on BERT with a bidirectional gated recurrent unit and self-attention mechanism as the ID decoder and a capsule network as the SF decoder. Their proposed model achieved a higher F1-score for both ID and SF compared to existing SOTA approaches.

2.3 Large Language Models for Joint ID and SF

The transformer architecture, increased computational power, and availability of vast datasets paved the way for large-scale language models (LLM) [16]. LLMs are pre-trained with billions of model parameters and terabytes of data. Through their massive scale, LLMs acquire emergent abilities that enable them to perform well in a variety of NLP tasks, including arithmetic reasoning, logical reasoning, and question answering without being specifically trained in these domains [3, 16].

ChatGPT is a conversational LLM fine-tuned from GPT 3.5 with reinforcement learning from human feedback to generate human-like responses to prompts [25]. With its demonstration of promising performance in different NLP tasks, research interests in evaluating the model's performance in ID and SF tasks began to surface. Pan et al. [21] conducted a preliminary study to evaluate ChatGPT's zero-shot performance on the joint ID and SF task. However, analysis of their experiments revealed that one-shot prompting may have been used instead based on the prompt template wherein one example input-output pair was included to specify the output format. Results showed that ChatGPT reached good ID accuracy but not at par with other LLMs (i.e., GPT-3.5 text-davinci-003 version and Codex) and the fine-tuned SOTA model (i.e., Co-Interactive Transformer). It exhibited the lowest F1-score in SF compared to the other models. The study also experimented with adding slot descriptions and examples to the prompt which increased the SF performance. This experiment was performed for the SNIPS NLU dataset [5].

3 TASK DESCRIPTION

In this section, we define the joint intent detection and slot filling task. We also describe the ATIS dataset which was used in the experiment. Prompt engineering is an important component in instructing ChatGPT to perform the joint ID and SF task; we briefly explain this component in relation to our requirements.

3.1 Joint Task

Intent detection is the task of identifying the main goal of a user utterance [7]. It classifies a given utterance x into one of the predefined intent classes y^i based on the feature/s extracted from the utterance. Slot filling is the task of assigning semantic labels to the words in a given utterance. Given an input word sequence $x = (x_1, ..., x_N)$, slot filling assigns each word a slot label $y^s = (y^s_1, ..., y^s_N)$ from a predefined list of slot labels. As a joint task, the relationship between the intent class and slot labels in a given utterance is modeled and used to inform the ID and SF tasks.

3.2 Dataset

The Airline Travel Information Systems (ATIS) dataset [21] is a standardized dataset widely used for the joint ID and SF task. It consists of 5,871 utterances in the air travel domain [10, 22]. The dataset can be divided into 4,978 utterances for the training set and 893 utterances for the test set. It has 21 unique intent classes such as *airfare*, *flight*, and *ground service*. It has 120 slot labels that can be categorized into *airline*, *airport*, *arrival date*, *departure date*, *city name*, and *flight*.

ATIS contains single-turn utterances associated with a single intent in a single domain (i.e., air travel). Table 1 shows one training sample for a given utterance: "*i want to fly from baltimore to dallas round trip*". It has a single intent, identified as "`atis_flight`". The slot labels follow the IOB tagging format where the "B" prefix indicates the beginning of a slot, while the "I" prefix indicates the word inside or at the end of a slot [8]. The “O” label indicates that a word does not correspond to any of the predefined slot labels. In the given example, the word “*baltimore*” is labeled as "`B-fromloc.city_name`" to indicate the source city, while the word “*dallas*” is labeled as “`B-toloc.city_name`” to indicate the destination city. The word “*round*” is labeled as “`B-round_trip`” as it is the first part of the compound word "*round trip*" while “*trip*” is labeled as “`I-round_trip`” because it is part of the previous slot label. The other words in the given utterance are labeled as “O” because they do not signify any of the predefined slot labels.

3.3 Prompt Engineering

Prompts are used to guide an LLM in generating a response or performing a task [3]. Prompt engineering is the process of designing and optimizing prompts to obtain the desired output or response [3]. To instruct ChatGPT to perform the joint ID and SF task, we utilized zero-shot and few-shot prompting. In zero-shot prompting, an LLM must answer a question it was not trained on without being given any examples. Conversely, in few-shot prompting, examples are included in the prompt to guide the LLM in generating the desired response.

To automatically send queries to and receive responses from ChatGPT, we used the Chat Completions endpoint provided by OpenAI API [17]. The original prompt given to ChatGPT is shown in Table 2. The general structure of the prompt is patterned after the prompt formulated by Pan et al. [21] which consists of three (3) sections: intent class constraints, slot label constraints, and regulations. A persona or role (""*You are a joint intent detection and slot filling model.*") and a brief task description ("*You need to identify the intent and slot labels of a given sentence.*") were appended at the start of the prompt as prescribed by OpenAI [18]. The list of intent classes and slot labels, as provided in the ISEAR dataset, is shown in Table 2; the list is redacted for brevity.

There are five (5) regulations specified in the prompt. The first and second regulations inform ChatGPT to generate the correct number of outputs. For ID, only one intent class is provided as each utterance in the ATIS dataset only has a single intent. For SF, every word in a sentence must be given a slot label using IOB tagging. The third and fourth regulations restrict ChatGPT’s responses to the predefined list of intent classes and slot labels provided in the ATIS dataset. The fifth regulation instructs ChatGPT to generate the output in JSON format for easier parsing of responses. An output template is also provided for clarity.

4 METHOD

The performance of ChatGPT on the joint ID and SF task was evaluated under zero-shot and few-shot prompt settings using the ATIS dataset. For the few-shot prompting, five (5) experiments were conducted to insert five (5) different numbers of examples (i.e., 1, 5, 10, 20, 30) to see how these affect ChatGPT’s performance on the joint ID and SF task.

4.1 Dataset

Of the 4,978 utterances in the training set of ATIS, only the first 30 utterances were used as examples in the few-shot prompting experiment. The entire test set (i.e., 893 utterances) was used to evaluate the performance of ChatGPT in the joint task. ChatGPT was instructed to only choose from the 21 intent classes and 120 slot labels provided in the ATIS dataset for intent classification and slot filling, respectively. The utterances, intent classes, and slot labels from the ATIS dataset were used as is in the experiments.

4.2 Experiments

We performed the experiments using zero-shot prompting and few-shot prompting by inserting five (5) different numbers of examples (i.e., 1, 5, 10, 20, 30). Prompts are sent through the *messages* parameter of the OpenAI API call which takes an array of messages, each represented as a JSON object with two keys: role and content [19]. An example of a message JSON object is *{"role": "system", "content": "You are a helpful assistant."}*. The *system* role sets the behavior of ChatGPT as a helpful assistant. Another example is *{"role": "user", "content": "Who won the world series in 2020?"}*. In this example, the *user* role signifies that the prompt is sent by the user, which must be responded to by ChatGPT.

The Chat Completions endpoint was used to process the entire test set of the ATIS dataset by ChatGPT. The model employed was `gpt-3.5-turbo-1106` as it was the latest free model that supported the JSON mode [20]. The first message in the API call is a system message where the prompt as described in Table 2 was specified. For few-shot prompting, additional system messages were sent, named "`example_user`" and “`example_assistant`”. These system messages serve as “faked example messages” and are used to demonstrate the desired output to ChatGPT; thus, these messages are not part of a real conversation [26]. The final message in the API call is a user message that specifies the actual utterance to be annotated by ChatGPT. The *response_format* parameter was used to constrain the model’s output to a JSON object. Only one utterance is processed by ChatGPT per API call.

We then validated the output of ChatGPT for the six (6) prompt settings by checking that each utterance has been annotated with the correct number of intent classes and slot labels. To resolve any resulting inconsistencies, the prompts were rerun and modified if necessary. These are further discussed in Section 4.3.

4.3 Performance Evaluation

The performance of ChatGPT was evaluated using ID accuracy and SF F1-score. Specifically, the span-based micro-averaged F1-score was calculated for SF because it is the standard metric reported in literature [31]. A span includes all words with the same class of a slot label, regardless of their prefix. For example, the labels `B-airline_name` and `I-airline_name` comprise a span of class *airline_name*. The results were compared to the co-interactive transformer by Qin et al. [24]), a SOTA model. The zero-shot performance was benchmarked against other experiments with LLMs, i.e., ChatGPT and Codex [21]. However, the SF metric is changed to token-based macro-averaged F1-score as this was the metric used by Pan et al. [21].

5 RESULTS AND FINDINGS

In this section, we present the performance evaluation results of ChatGPT across various prompt settings and when compared with

Table 1: ATIS Dataset Sample with Intent and Slot Annotation (IOB format)

Utterance	i	want	to	fly	from	baltimore	to	dallas	round	trip
Slot Label	O	O	O	O	O	B-fromloc.city_name	O	B-toloc.city_name	B-round_trip	I-round_trip
Intent Class	atis_flight									

Table 2: Prompt for Joint ID and SF

Section	Prompt
Task Description	You are a joint intent detection and slot filling model. You need to identify the intent and slot labels of a given sentence. The sentence you need to annotate will be given in the next prompt after this.
Intent Class Constraints	First, choose the intent of the sentence from the following intent list: [atis_abbreviation, atis_aircraft, . . .]
Slot Label Constraints	Next, annotate the sentence with slots from the following slot label list in IOB tagging: [B-aircraft_code, B-airline_code, . . .]
Regulations	Strictly follow the regulations below: 1. Give only one intent for the sentence. 2. Give the slot label of every word in the sentence based on IOB tagging. 3. Intents must only come from the provided intent list. 4. Slot labels must only come from the provided slot label list. 5. You need to output the intent and slot labels in JSON format: { "intent": "<insert chosen intent>", "slots": [{"word": "<insert word in the sentence>", "slot_label": "<insert chosen slot label>"}] } 6. The number of slots should be equivalent to the number of words in the sentence.

other LLMs and SOTA models. We also share our findings on misclassified intents and slot labels from analyzing the output generated by ChatGPT.

5.1 Zero-shot v.s. Few-shot Performance

The performance of ChatGPT across the six (6) prompt settings is presented in Table 3. It can be observed that ID accuracy and span-based micro-averaged SF F1-score generally increase as the number of examples provided to ChatGPT increases, with the 30-prompt setting achieving an ID accuracy of 87.46% and SF F1-score of 89.13%. This indicates that providing more context or examples to ChatGPT can improve its ability to produce the desired output.

However, 1-shot prompting is an exception, obtaining the lowest accuracy score for ID. Similar behavior was observed in the work of Zhong et al. [33] where ChatGPT did not perform well under 1-shot prompting for other NLU tasks. This illustrates the risk of providing only one example to ChatGPT as it may serve as noise that could skew its performance.

It is worth noting that ChatGPT had the steepest increase in SF F1-score under 1-shot prompting. This may be attributed to the multiple slot-value pairs that were given to ChatGPT. In multiple slot-value pairs, each word in an utterance has an associated slot label which provided more context that may have aided ChatGPT in performing the SF task, and thus, the increased performance. It can also be observed that score improvements become gradual upon reaching a certain point (i.e., 10-shot prompting). The SF F1-score of the 20-shot prompting is slightly lower than that of the 10-shot prompting. This may be attributed to the increased length of the prompt, as reported in the study of He et al. [9].

5.2 Comparison with Other Models

The zero-shot performance of ChatGPT was compared with the results reported by Pan et al. [21] using ID accuracy and token-based macro-averaged SF F1-score as seen in Table 4. The evaluation metric used for SF was changed to token-based macro-averaged F1-score as this was the metric used by Pan et al. [21]. This means that the slot F1-score was computed as the mean of the individual F1-scores per token-level slot label, where slot label classes with different prefixes are treated as distinct slot labels.

As can be seen in the table, ChatGPT's performance using the model version released in January 2024 is better compared to its version released in 2023. However, it performed worse compared to Codex, a 12B parameter GPT model fine-tuned on publicly available code from GitHub [4]. Our work was compared to Codex for a more direct comparison with the results reported by Pan et al. who also evaluated Codex [21].

Table 3: ChatGPT's Performance in Joint ID and SF

Prompt Setting	ID Accuracy (%)	SF F1-Score (%)
0-shot	76.15	80.07
1-shot	69.09	83.17
5-shot	71.44	85.68
10-shot	83.20	87.20
20-shot	86.45	86.99
30-shot	**87.46**	**89.13**

Table 4: Zero-shot Performance Comparison of ChatGPT with Other LLMs

Model	ID Accuracy (%)	SF F1-score (%)
Codex	**89.92**	**57.29**
ChatGPT 2023	75.22	15.71
This work (ChatGPT 2024)	76.15	47.32

It is interesting to note that the token-based **macro**-averaged F1-score is much lower compared to the span-based micro-averaged F1-score previously reported in Table 3 for SF. A micro-averaged F1-score takes label imbalance into account by computing the result based on the proportion of every class. On the other hand, a **macro**-averaged F1-score does not consider label imbalance as it simply gets the mean of the F1-score of each class. The slot labels of the ATIS dataset are highly imbalanced, which may have introduced bias to the results. Another factor that contributed to the higher micro-averaged F1-score is with the use of spans. Many of ChatGPT's mislabels belong to the same span or class, which means that a portion of the errors were semantically related. For example, the ground truth slot label "`B-depart_time.period_of_day`" was frequently mislabeled as "`I-depart_time.period_of_day`" which belongs to the same class (see Table 7).

The highest scores obtained by ChatGPT under the 30-shot prompting were compared to results of the co-interactive transformer developed by Qin et al. [24]. As seen in the results in Table 5, it is evident that the SOTA model outperforms ChatGPT. However, it is worth noting that the SOTA model was trained on 4,978 utterances, whereas ChatGPT was only shown 30 sample annotations. Based on the results of varying the prompt settings in Table 3, there is a possibility that ChatGPT may achieve the same performance as the SOTA model using less training data. However, this is yet to be proven empirically as there are other factors affecting ChatGPT's performance such as input length.

5.3 Misclassified Intents

The top 5 misclassified intents with their corresponding top 5 frequently predicted values are listed in Table 6. It can be observed that the ground truth and predicted values are semantically related. The ground truth intent classes are usually mispredicted with combined intents that start with the same intent class (e.g., "`atis_flight`" is mispredicted as either "`atis_flight#atis_flight_time`" or "`atis_flight#atis_airfare`"). These combined intents were created so that utterances in ATIS will only be labeled with a single intent despite encompassing overlapping intents. There is inherent ambiguity in the intent annotations, which diminishes the performance of ChatGPT in ID. Relationships can still be gleaned from the other mispredictions that are not straightforward. For example, the "`atis_abbreviation`" intent class is usually mispredicted with the "`atis_fare_basis_code`" intent class, which is similar to abbreviations, or to the types of abbreviations in the queries (e.g., *fare code, restriction code, airport code, meal code*). These show that ChatGPT's misclassifications are logical and not random.

Table 5: ChatGPT's 30-shot Performance Comparison with a SOTA Model

Model	ID Accuracy (%)	SF F1-score (%)
SOTA Model (Co-interactive Transformer)	**98.00**	**96.10**
This work (ChatGPT 2024 with 30-shot Prompting)	87.46	89.13

Table 6: Top 5 Misclassified Ground Truth Intents and their Frequently Predicted Values

Ground Truth Intent	Frequently Predicted Intents
1. atis_flight	atis_flight#atis_flight_time, atis_flight#atis_airfare, atis_flight_time, atis_cheapest, atis_airline#atis_flight_no
2. atis_abbreviation	atis_fare_basis_code, atis_airfare, atis_restriction, atis_airport, atis_meal
3. atis_airfare	atis_airfare#atis_flight, atis_cheapest, atis_flight, atis_flight#atis_flight_time
4. atis_airline	atis_airline#atis_flight_no, atis_airline#atis_flight, atis_airline#atis_flight#atis_flight_no, atis_airline#atis_flight_time
5. atis_flight#atis_airfare	atis_airfare#atis_flight_time, atis_airfare, atis_flight, atis_airfare#atis_flight, atis_flight#atis_flight_time

5.4 Mislabeled Slots

The top 5 mislabeled slots with their corresponding top 5 frequently predicted values are listed in Table 7. Upon inspection, it is apparent that a lot of the mislabeled slots are predicted with slot labels that are more specific but still related (e.g., B-city_name is labeled as "`B-toloc.city_name`" or "`B-fromloc.city_name`"). The "O" label was the most mislabeled slot, partly because it has the highest frequency compared to the other slot labels. ChatGPT appears to have a tendency to consider a word as a slot despite not corresponding to any of the predefined slot labels. The mislabels of "`B-depart_time.period_of_day`" are not straightforward, but still logical. It is usually mispredicted with slot labels that are also pertaining to time, such as "`I-depart_time.period_of_day,`" and "`B-depart_time.time`". These illustrate that ChatGPT's errors in SF are sensible and not entirely stochastic.

5.5 Unexpected Behaviors of ChatGPT

Throughout the conduct of the experiments, ChatGPT exhibited unexpected behaviors deviant to the specified instructions in the prompt. These unexpected behaviors can be classified into three:

Table 7: Top 5 Mislabeled Ground Truth Slots and their Frequently Predicted Values

Ground Truth Slot	Frequently Predicted Slots
1. O	B-flight, B-transport_type, B-toloc.city_name, I-transport_type, B-aircraft_code
2. B-city_name	B-toloc.city_name, B-fromloc.city_name, I-toloc.city_name, O, B-stoploc.city_name
3. I-airport_name	I-fromloc.airport_name, I-toloc.airport_name, I-fromloc.city_name, B-fromloc.airport_name, I-stoploc.airport_name
4. I-city_name	I-toloc.city_name, I-fromloc.city_name, O, I-fromloc.airport_name, I-stoploc.city_name
5. B-depart_time.period_of_day	I-depart_time.period_of_day, B-depart_time.time, B-depart_time.period_mod, B-flight_time, B-arrive_time.period_of_day

(1) unknown intent classes and slot labels, (2) incorrect number of slots, and (3) annotating incorrect utterances.

5.5.1 Unknown Intent Classes and Slot Labels. ChatGPT outputted several unknown intent classes and slot labels despite being instructed to only select from the predefined lists specified in the prompt. Table 8 lists the top 5 most frequent unknown intent classes and their corresponding ground truth value/s. It can be seen that the unknown intent classes are mostly more specific versions of the general "`atis_flight`" intent class. ChatGPT also invented a more specific "`atis_abbreviation`" intent class, that is, "`atis_fare_basis_code`", which is a frequently asked type of abbreviation.

Table 9 shows the top 5 most frequent unknown slot labels and their corresponding ground truth value/s. It can be observed that ChatGPT can infer new slot labels from the words found in an utterance as it invents slot labels for those that should have been tagged as "O" or null. ChatGPT also experiences difficulty in identifying the correct prefix of slot labels and has a tendency to label words with the "I-" prefix instead of "B-" despite it being the first word identified for that slot's span of text.

5.5.2 Incorrect Number of Slot Labels. We found many instances where ChatGPT outputs an incorrect number of slot labels. We clustered them into four (4) different scenarios as shown in Table 10 and provided an example. To address these scenarios, we included a 6th regulation to our original prompt by explicitly stating that *the number of slot labels outputted must be equal to the number of words in the given sentence.* When this did not fully resolve the problem, the explicit instruction, "*Do not leave out any word until the end of the prompt*" is sometimes added to the prompt. Some occurrences of concatenations were separated resulting in inconsistent output. For this, the prompt was modified to include a 7th regulation, stating that "*contractions must not be separated*".

Table 8: Unknown Intent Classes and Their Corresponding Ground Truth Values

Unknown Intent	Ground Truth Intent
1. atis_flight#atis_flight_time	atis_flight, atis_flight#atis_airfare, atis_flight_no, atis_airfare
2. atis_airfare#atis_flight	atis_flight, atis_flight_no#atis_airline, atis_flight_no, atis_city, atis_flight#atis_airline
3. atis_flight#atis_airline	atis_flight, atis_city
4. atis_airline#atis_flight	atis_flight, atis_city, atis_airline
5. atis_fare_basis_code	atis_abbreviation

Table 9: Unknown Slot Labels and their Corresponding Ground Truth Value/s

Unknown Slot	**Ground Truth Slot**
1. B-flight	O, B-flight, I-round_trip
2. I-aircraft_code	B-aircraft_code, O
3. I-depart_date.day_name	O, B-depart_date.day_name, B-day_name
4. B-capacity	O
5. I-toloc.state_code	B-toloc.state_code, B-state_code

5.5.3 Annotating Incorrect Utterances. In the 20-shot and 30-shot prompting, ChatGPT exhibited several instances where it annotated a different utterance from what was provided by the user. These utterances were mostly previous utterances that were annotated by it already. This indicates that longer prompts could trigger unexpected behaviors in ChatGPT. This issue was resolved by rerunning the prompt.

6 DISCUSSION

LLMs have shown competitive performance in a variety of tasks across different domains using zero-shot or few-shot prompting against SOTA models [23]. Different prompting strategies can also enhance the responses of LLMs such as chain-of-thought, self-question, clue and reasoning prompting, and persona specification [12, 18]. We found three main factors that may affect ChatGPT's performance in the joint ID and SF task.

Table 10: Scenarios Causing Incorrect Number of Outputted Slot Labels and their Examples

Scenario	Utterance	Wrong Annotation Example
A new word is annotated	saturday flight on american airlines from milwaukee to phoenix	{'intent': 'atis_flight', 'slots': [{'word': 'saturday', 'slot_label': 'B-depart_date.day_name'}, {'word': 'flight', 'slot_label': 'O'}, {'word': 'on', 'slot_label': 'O'}, {'word': 'american', 'slot_label': 'B-airline_name'}, {'word': 'airlines', 'slot_label': 'I-airline_name'}, {'word': 'from', 'slot_label': 'O'}, {'word': 'milwaukee', 'slot_label': 'B-fromloc.city_name'}, {'word': 'to', 'slot_label': 'O'}, {'word': 'phoenix', 'slot_label': 'B-toloc.city_name'}, {'word': 'to', 'slot_label': 'O'}]}
A word is repeatedly annotated	what types of ground transportation are available in denver	{'intent': 'atis_ground_service', 'slots': [{'word': 'what', 'slot_label': 'O'}, {'word': 'types', 'slot_label': 'O'}, {'word': 'of', 'slot_label': 'O'}, {'word': 'ground', 'slot_label': 'O'}, {'word': 'ground', 'slot_label': 'B-transport_type'}, {'word': 'transportation', 'slot_label': 'I-transport_type'}, {'word': 'are', 'slot_label': 'O'}, {'word': 'available', 'slot_label': 'O'}, {'word': 'in', 'slot_label': 'O'}, {'word': 'denver', 'slot_label': 'B-city_name'}]}
A word is not annotated	what does ua mean	{'intent': 'atis_abbreviation', 'slots': [{'word': 'what', 'slot_label': 'O'}, {'word': 'does', 'slot_label': 'O'}, {'word': 'ua', 'slot_label': 'B-abbreviation'}]}
Contractions are separated	list the distance in miles from new york's la guardia airport to downtown new york city	{'intent': 'atis_distance', 'slots': [{'word': 'list', 'slot_label': 'O'}, {'word': 'the', 'slot_label': 'O'}, {'word': 'distance', 'slot_label': 'O'}, {'word': 'in', 'slot_label': 'O'}, {'word': 'miles', 'slot_label': 'O'}, {'word': 'from', 'slot_label': 'O'}, {'word': 'new', 'slot_label': 'B-fromloc.city_name'}, {'word': 'york', 'slot_label': 'I-fromloc.city_name'}, {'word': "'s", 'slot_label': 'O'}, {'word': 'la', 'slot_label': 'B-fromloc.airport_name'}, {'word': 'guardia', 'slot_label': 'I-fromloc.airport_name'}, {'word': 'airport', 'slot_label': 'I-fromloc.airport_name'}, {'word': 'to', 'slot_label': 'O'}, {'word': 'downtown', 'slot_label': 'B-toloc.city_name'}, {'word': 'new', 'slot_label': 'I-toloc.city_name'}, {'word': 'york', 'slot_label': 'I-toloc.city_name'}, {'word': 'city', 'slot_label': 'I-toloc.city_name'}]}

Contextual information. The number of examples provided to ChatGPT influences its performance in the joint task. The results showed that providing ChatGPT with more examples generally improves its performance for both ID and SF. However, the improvement becomes gradual upon reaching a certain point. It is also worth mentioning that the 1-shot performance is worse compared to zero-shot for ID, which shows ChatGPT's sensitivity to noise.

Metrics for evaluation. Metric selection is another crucial factor to obtain an accurate evaluation of ChatGPT's performance, as evidenced by the big gap between the span-based micro-averaged slot F1 score and the token-based macro-averaged slot F1 score in this study. The former metric accounts for data imbalance and semantically related slot labels (i.e., belonging to the same class), and is the standard used in literature [31].

Unexpected behaviors. Lastly, ChatGPT tends to exhibit unexpected behaviors that deviate from the specified instructions. These unexpected behaviors include generating unknown outputs, outputting incorrect number of slot labels, and annotating previous prompts instead of the current one. These unexpected behaviors were observed more frequently with longer prompts, which shows that prompt length has an impact on ChatGPT's performance. ChatGPT also demonstrated unexpected behaviors in other tasks such as emotion classification [13] and code generation [1].

7 CONCLUSION

In this study, we compared ChatGPT's performance on the joint ID and SF task under zero-shot and different few-shot (i.e., 1-shot, 5-shot, 10-shot, 20-shot, and 30-shot) prompt settings. Results showed that giving more examples to ChatGPT generally improves its performance, but becomes gradual upon reaching a certain threshold. Future work could investigate the number of examples at which ChatGPT's performance begins to decline on the joint ID and SF task. This would also reveal whether ChatGPT can be at par, or even surpass the performance of SOTA models by providing more examples to it. This raises the need for a mechanism to handle long prompts, such as compression, due to the token limits of LLMs. This may also mitigate ChatGPT's performance issues associated with long prompts.

REFERENCES

[1] Alessio Buscemi. 2023. A Comparative Study of Code Generation using ChatGPT 3.5 across 10 Programming Languages. *ArXiv* abs/2308.04477 (2023). https://api.semanticscholar.org/CorpusID:260735527

[2] Asli Celikyilmaz and Dilek Hakkani-Tur. 2012. A joint model for discovery of aspects in utterances. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Haizhou Li, Chin-Yew Lin, Miles Osborne, Gary Geunbae Lee, and ong C. Park (Eds.). Association for Computational Linguistics, Jeju Island, Korea, 330–338. https://aclanthology.org/P12-1035

[3] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. 2024. A survey on evaluation of large language models. *ACM Trans. Intell. Syst. Technol.* (jan 2024). https://doi.org/10.1145/3641289 Just Accepted.

[4] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code. arXiv:2107.03374 [cs.LG]

[5] Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, and Joseph Dureau. 2018. Snips Voice Platform: An embedded spoken language understanding system for private-by-design voice interfaces. arXiv:1805.10190 [cs.CL]

[6] Jacob Devlin, Ming-Wei Chang, and Kristina Toutanova Kenton Lee. 2019. 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)*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[7] Mauajama Firdaus, Shobhit Bhatnagar, Asif Ekbal, and Pushpak Bhattacharyya. 2018. *Intent detection for spoken language understanding using a deep ensemble model: 15th Pacific Rim International Conference on Artificial Intelligence, Nanjing, China, August 28–31, 2018, Proceedings, Part I.* 629–642. https://doi.org/10.1007/978-3-319-97304-3_48

[8] Daniel Guo, Gokhan Tur, Wen tau Yih, and Geoffrey Zweig. 2014. Joint semantic utterance classification and slot filling with recursive neural networks. In *2014 IEEE Spoken Language Technology Workshop (SLT)*. 554–559. https://doi.org/10.1109/SLT.2014.7078634

[9] Mutian He and Philip N. Garner. 2023. Can ChatGPT detect intent? Evaluating large language models for spoken language understanding. In *Proc. INTERSPEECH 2023*. 1109–1113. https://doi.org/10.21437/Interspeech.2023-1799

[10] Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The ATIS spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*. https://aclanthology.org/H90-1021

[11] Minwoo Jeong and Gary Geunbae Lee. 2008. Triangular-chain conditional random fields. *IEEE Transactions on Audio, Speech, and Language Processing* 16 (2008), 1287–1302. https://api.semanticscholar.org/CorpusID:14617149

[12] Katikapalli Subramanyam Kalyan. 2024. A survey of GPT-3 family large language models including ChatGPT and GPT-4. *Natural Language Processing Journal* 6 (2024), 100048. https://doi.org/10.1016/j.nlp.2023.100048

[13] Jan Kocoń, Igor Cichecki, Oliwier Kaszyca, Mateusz Kochanek, Dominika Szydło, Joanna Baran, Julita Bielaniewicz, Marcin Gruza, Arkadiusz Janz, Kamil Kanclerz, Anna Kocoń, Bartłomiej Koptyra, Wiktoria Mieleszczenko-Kowszewicz, Piotr Miłkowski, Marcin Oleksy, Maciej Piasecki, Łukasz Radliński, Konrad Wojtasik, Stanisław Woźniak, and Przemysław Kazienko. 2023. ChatGPT: Jack of all trades, master of none. *Information Fusion* 99 (2023), 101861. https://doi.org/10.1016/j.inffus.2023.101861

[14] Sheetal Kusal, Shruti Patil, Jyoti Choudrie, Ketan Kotecha, Sashikala Mishra, and Ajith Abraham. 2022. AI-Based conversational agents: A scoping review from technologies to future directions: Conversational agents. *IEEE Access* 10 (Aug 2022), 92337–92356. https://doi.org/10.1109/ACCESS.2022.3201144

[15] Bing Liu and Ian R. Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. In *Interspeech 2016, 17th Annual Conference of the International Speech Communication Association, San Francisco, CA, USA, September 8-12, 2016*, Nelson Morgan (Ed.). ISCA, 685–689. https://doi.org/10.21437/INTERSPEECH.2016-1352

[16] Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. 2024. A comprehensive overview of large language models.

[17] OpenAI. [n. d.]. https://platform.openai.com/docs/quickstart?context=python

[18] OpenAI. [n. d.]. https://platform.openai.com/docs/guides/prompt-engineering

[19] OpenAI. [n. d.]. https://platform.openai.com/docs/guides/text-generation

[20] OpenAI. 2023. https://openai.com/blog/new-models-and-developer-products-announced-at-devday#OpenAI

[21] Wenbo Pan, Qiguang Chen, Xiao Xu, Wanxiang Che, and Libo Qin. 2023. A preliminary evaluation of chatGPT for zero-shot dialogue understanding. arXiv:2304.04256 [cs.CL]

[22] Jangwon Park. 2021. Jointbert: Pytorch implementation of Jointbert: "Bert for joint intent classification and slot filling". https://github.com/monologg/JointBERT/tree/master

[23] Chengwei Qin, Aston Zhang, Zhuosheng Zhang, Jiaao Chen, Michihiro Yasunaga, and Diyi Yang. 2023. Is ChatGPT a general-purpose natural language processing task solver?. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Singapore, 1339–1384. https://doi.org/10.18653/v1/2023.emnlp-main.85

[24] Libo Qin, Tailu Liu, Wanxiang Che, Bingbing Kang, Sendong Zhao, and Ting Liu. 2021. A co-interactive transformer for joint slot filling and intent detection. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 8193–8197. https://doi.org/10.1109/ICASSP39728.2021.9414110

[25] Partha Pratim Ray. 2023. ChatGPT: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope. *Internet of Things and Cyber-Physical Systems* 3 (2023), 121–154. https://doi.org/10.1016/j.iotcps.2023.04.003

[26] Ted Sanders. 2023. How to format inputs to CHATGPT models. https://cookbook.openai.com/examples/how_to_format_inputs_to_chatgpt_models

[27] Nahida Shafi and Manzoor Ahmed Chachoo. 2023. Fine-Tuned BERT with attention-based Bi-GRU-CapsNet framework for joint intent recognition and slot filing. In *2023 International Conference on Advancement in Computation Computer Technologies (InCACCT)*. 369–374. https://doi.org/10.1109/InCACCT57535.2023.10141744

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc., 1–11. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[29] Haifeng Wang, Jiwei Li, Hua Wu, Eduard Hovy, and Yu Sun. 2023. Pre-Trained language models and their applications. *Engineering* 25 (2023), 51–65. https://doi.org/10.1016/j.eng.2022.04.024

[30] Ye-Yi Wang. 2010. Strategies for statistical spoken language understanding with small amount of data - an empirical study. In *Proceedings of Interspeech* (proceedings of interspeech ed.). International Speech Communication Association. https://www.microsoft.com/en-us/research/publication/strategies-for-statistical-spoken-language-understanding-with-small-amount-of-data-an-empirical-study/

[31] Henry Weld, Xiaoqi Huang, Siqu Long, Josiah Poon, and Soyeon Caren Han. 2022. A survey of joint intent detection and slot-filling models in natural language understanding. *Comput. Surveys* 55, 8 (2022), 1–38. https://doi.org/10.1145/3547138

[32] Puyang Xu and Ruhi Sarikaya. 2013. Convolutional neural network based triangular CRF for joint intent detection and slot filling. *2013 IEEE Workshop on Automatic Speech Recognition and Understanding* (2013), 78–83. https://api.semanticscholar.org/CorpusID:10532715

[33] Qihuang Zhong, Liang Ding, Juhua Liu, Bo Du, and Dacheng Tao. 2023. Can ChatGPT understand too? A comparative study on ChatGPT and fine-tuned BERT. arXiv:2302.10198 [cs.CL]