Towards Automatic Evaluation Of Task-Oriented Dialogue Flows

Anonymous ACL submission

Abstract

Task-oriented dialogue systems are an important tool in conversational AI, relying on predefined dialogue flows to guide user-agent interactions. Currently, these flows are either manually created by domain experts or generated through AI techniques, leading to variability in their quality. This lack of standardization poses a challenge for conversational designers and automated techniques. To address this issue, we propose a novel framework for evaluating the quality of these dialogue flows. Our framework focuses on the complexity of the flow structure and its representation of historical data. To this end, we introduce FuDGE, a Fuzzy Dialogue-Graph Edit Distance metric that assesses the match between a flow and a set of real-world conversations. Through extensive experiments, we demonstrate the effectiveness of our framework and show that FuDGE can help standardize and improve dialogue flows for task-oriented dialogue systems.

1 Introduction

Conversational AI holds great promise in the field of Customer Service Automation, where the aim is to create a task-oriented dialogue system that can effectively address customer queries without requiring human intervention. These systems are often built on a knowledge base or backend systems to provide the necessary information to assist the customer. Over the past few years, various frameworks have been developed to make the creation of task-oriented dialogue systems easier, including Dialogflow CX[1], Rasa[2], and Amazon Lex[3], which are based on dialogue flows composed of user intents, agent actions, and other metadata.

There are two primary methods for building dialogue flows for task-oriented dialogue systems. The first is through handcrafting by conversation designers, involving an iterative process of historical data analysis to identify and define the dialogue flows that meet customers' concerns. This process can result in different dialogue flows due to various factors, including the designer's skill and experience, the time spent on development, and the historical dialogue transcripts. The second approach involves utilizing flow discovery algorithms, which enable the identification of dialogue flows from historical human-human dialogues (Bouraoui et al., 2019; Martínez and Nugent, 2022). Although this approach is critical for building task-oriented dialogue agents, it remains largely unexplored. Flow discovery methods comprise different hyperparameters, which contribute to the emergence of various dialogue flow types. The resulting dialogue flows are represented as directed acyclic graphs (DAGs) that may differ in terms of the number of user intents, the sequence of intents and actions, or the number of paths within the graph. To illustrate the concept of a dialogue flow and its corresponding task-oriented dialogue agent, refer to Figure 1.

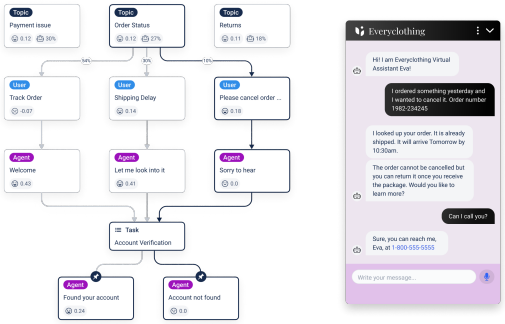

Figure 1: Illustration of a dialogue flow (left) used to configure a task-oriented dialogue agent for "Everyclothing", and a customer inquiring about canceling their order shown in the chat widget (right).

Despite the variability of the dialogue flows, there has been limited attention in natural language processing (NLP) research on evaluating these flows with respect to historical dialogues. While

[1]https://cloud.google.com/dialogflow/cx/docs/basics
[2]https://rasa.com
[3]https://aws.amazon.com/lex/

some studies have improved task-oriented dialogue agents through better intent discovery (Zhang et al., 2022; Shi et al., 2018; Perkins and Yang, 2019), which is a sub-problem in the dialogue flow discovery process, there is a need for automated evaluation of dialogue flows with respect to historical dialogue transcripts. In this paper, we present a novel evaluation framework to assess flow discovery algorithms and guide conversation designers in building flows. Our framework assigns a score to a dialogue flow and enables comparison between flows generated from the same corpus. When designing dialogue flows, there is a trade-off between compression and the amount of information retained. A graph containing all conversations potentially covers the entire corpus, but strictly imitates historical conversations and does not generalize well to unseen dialogues. Conversely, a simple graph such as "`start -> Hello -> Goodbye`" is still a dialogue flow, but it loses most of the information about the corpus. Our proposed evaluation framework aims to balance the compression and complexity of dialogue flows to ensure that important and representative conversations are covered while minimizing information loss. We propose FuDGE, **Fu**zzy **D**ialogue-**G**raph **E**dit distance, an algorithmic way to compute the distance between a conversation and a given flow path, measuring how well dialogues are represented and covered by the flow. We combine the FuDGE score with the complexity of the DAG representing the flow to produce the **Flow**-**F**1 (*FF1*) score, which captures the compression-complexity trade-off. Overall, our contributions are as follows:

1. We present a novel dialogue flow evaluation framework that quantifies the quality of a dialogue flow generated from a corpus while considering the trade-off between flow complexity and information content.

2. We propose the FuDGE distance, an efficient edit-distance metric and show that can be used in any dialogue evaluation task and effectively separates within-task and out-of-task conversations for a given set of dialogues.

3. We demonstrate that *FF1* score can help hyperparameter tuning, ranking, and pruning the dialogue flows to identify the most optimal flows for a given dialogue corpus.

2 Preliminaries

This section first defines key terms related to dialogue flows and presents the core concept behind automatic dialogue flow discovery methods. We then describe the Levenshtein distance algorithm (Levenshtein et al., 1966) as the basis for the FuDGE algorithm.

2.1 Automatic Flow Discovery

Automatic flow discovery involves obtaining a dialogue flow from a dialogue corpus, which is a collection of N dialogues, each containing a sequence of user and agent utterances. A dialogue flow is a directed acyclic graph (DAG) that represents the corpus in the form of nodes, where each node is a user intention/request or an agent response. A naive approach to obtain a graph from a dialogue corpus would be to assign a node to every utterance and connect the consecutive nodes. This would result in a graph that captures all the noise inherent in human-human conversations, making it impractical for initializing a dialogue agent or manual configuration. Moreover, it would be too large for practical, real-world datasets.

To create more condensed representations of the graph, automatic flow discovery techniques broadly follow two steps. The first step involves identifying the type of agent responses and user requests and assigning an intent label to each utterance. An intent label "`book hotel`" might include user utterances like "`I want to book a hotel`", "`Need a room in the Marriott for next month`", and "`Need accommodation this weekend`". Various techniques can be employed to achieve this, such as training a classifier on manually annotated user and agent utterances or using unsupervised or semi-supervised methods when annotated data is limited (Forman et al., 2015; Lin et al., 2020; Zhang et al., 2021). The second step involves using AI techniques to convert the conversation paths into a more condensed graph. This can be done by employing flow ranking strategies that discover the most important paths, and pruning strategies that remove less important nodes or edges from the graph.

While there is a strong desire for automatic flow discovery, the number of available discovery algorithms is limited. Graph2Bot, a discovery algorithm described in (Bouraoui et al., 2019), can detect large, complex graphs, but its over-generation of paths results in loops that make the dialogue

flows unsuitable for a dialogue system, despite the possibility of filtering. An alternative approach, presented in (Martínez and Nugent, 2022), involves creating an NFA by treating user utterances as states and the agent intents as transitions between states. The algorithm groups user utterances during graph minimization and employs several ranking and pruning techniques to create a more compressed graph. In this work, we use both the NFA-based algorithm described in (Martínez and Nugent, 2022) and another proprietary automatic flow discovery method to obtain dialogue flows. Our goal is to generate a dialogue flow that resembles the one depicted in Figure 1. We can use annotated data, such as agent and user utterance intent labels, or a clustering method to assign labels to the utterances for both algorithms.

2.2 Levenshtein Distance

The Levenshtein distance, also known as edit distance, is a fundamental concept in various search algorithms. It measures the minimum number of operations required to transform one string, $a = a_1a_2a_3 \dots a_n$, into another string, $b = b_1b_2b_3 \dots b_m$, where each sequence comprises individual characters. To compute the edit distance between substrings $a_{1:r}$ and $b_{1:s}$, we define $d_{r,s}$ as the minimum number of operations required. We can recursively calculate $d_{r,s}$ by considering the following three possibilities:

1. delete a_r, match $a_{1:r-1}$ with $b_{1:s}$.
2. insert b_s at the end of $a_{1:r}$, match $a_1a_2 \dots a_rb_s$ with $b_1b_2 \dots b_{s-1}b_s$ or equivalently $a_{1:r}$ and $b_{1:s-1}$.
3. substitute a_r with b_s, match $a_{1:r-1}$ and $b_{1:s-1}$.

The minimum number of operations $d_{r,s}$ is the minimum cost of the three previous steps:

$$d_{r,s} = min \begin{cases} d_{r-1,s} + c_{del}(a_r), & \text{case. 1} \\ d_{r,s-1} + c_{ins}(b_s), & \text{case. 2} \\ d_{r-1,s-1} + c_{sub}(a_r, b_s), & \text{case. 3} \end{cases} \quad (1)$$

Where c_{del}, c_{ins}, c_{sub} are the deletion, insertion and substitution cost respectively.The substitution cost c_{sub}, is 0 if the characters being compared are the same and 1 otherwise. The initial cases, $d_{0,s}$ and $d_{r,0}$, represent the cost of converting an empty string to a full string and a full string to an empty string, respectively. They are defined as follows:

$$d_{0,s} = \sum_{t=0}^{s} c_{ins}(b_t), d_{r,0} = \sum_{t=0}^{r} c_{del}(a_t) \quad (2)$$

The final edit distance between a and b is $d_{m,n}$, and it can be calculated efficiently using dynamic programming with complexity $\mathcal{O}(mn)$. Furthermore, tracking the steps at each iteration gives us the *alignment* between two strings, where each character of the first string is either matched with another character or a gap in the other string.

2.3 Problem Definition

In this paper, a dialogue flow is represented by a Directed Acyclic Graph (DAG) denoted by $G = (V, E)$ with its root node G_r. The graph G contains multiple paths starting from the root node, each representing a possible scenario of interaction between a user and an agent in a dialogue system. Every node in the graph is linked to an intent bucket $B^i \in B = \{B^1, B^2, \dots, B^M\}$, where a bucket $B^i = (actor, utterances)$ consists of a group of utterances conveying the same semantic meaning associated with either a user intent ($actor = user$) or an agent action ($actor = agent$). Throughout the paper, the term 'intent' is used for both users and agents. The root node G_r is associated with a dummy bucket and is assumed to be the beginning of all conversations. A flow path $P = \langle G_rP_1P_2...P_n \rangle$ is defined as a path in the graph G, where $P_i \in V$ and (P_i, P_{i+1}) is an edge in E. To generate a dialogue flow, human experts or automated dialogue flow discovery methods use a dialogue corpus $C = \{C^1, C^2, \dots, C^N\}$ that contains N recorded conversations between users and agents in a service center.

3 Methodology

Given a dialogue flow graph $G = (V, E)$ obtained from a dialogue corpus $C = \{C^1, C^2, \dots, C^N\}$, we evaluate the flow from two perspectives: (i) how well does the flow represent the dialogue corpus (information loss) (ii) how well is the representation compressed/denoised (complexity).

3.1 Information Loss

Information loss measures the similarity between a dialogue corpus and the discovered flow from the corpus. It is calculated by determining the average distance between each dialogue in the corpus and

the flow. Essentially, the more similar the conversations in the corpus are to the paths in the flow, the lower the amount of information loss. To formally define this metric, we assume that $C^i \in C$ is a dialogue, and $F_G = \{P^k = \langle G_r P_1^k P_2^k ... P_{n_k}^k \rangle\}_K$ is the set of all flow paths in G starting at the root node G_r and ending at a leaf node. The fuzzy dialogue-graph distance between C^i and G is then defined as the minimum distance between C^i and any flow path $P^k \in F_G$:

$$FuDGE(C^i, G) = \min_{P^k \in F_G} dist(C^i, P^k) \quad (3)$$

Where $dist(C^i, P^k)$ denotes the edit distance between a dialogue and a single flow path, and is computed by pairing each node intent in the flow with an utterance in the conversation using insertion, deletion, or substitution operations of nodes in the path. Unlike the Levenshtein distance for strings, this process is challenging because a dialogue is a sequence of utterances while a flow path is a sequence of intents (which is a collection of utterances), instead of being a sequence of unit characters. The FuDGE algorithm takes into account the unique characteristics of dialogues and efficiently calculates the distance between a dialogue and a flow. We will provide more details about this algorithm in Section 4. The distance between the dialogue flow G and corpus C is then calculated as the average FuDGE distance across all N dialogues in the corpus:

$$f(C, G) = \frac{1}{N} \sum_{i=1}^{N} FuDGE(C^i, G) \quad (4)$$

3.2 Complexity

The complexity of the representation can be defined as the complexity of the graph representing the dialogue flow. Depending on the application, there are various ways to calculate the complexity of a graph. Here we define complexity as the number of nodes of a graph.

3.3 Flow-F1 Score (*FF1*)

To capture the trade-off between the information loss and the complexity, we propose using the harmonic mean of the normalized complexity and the FuDGE score. To normalize the complexity, we divide it by the total number of utterances in the dialogue corpus, which represents the upper bound for graph size if each utterance is a node. We normalize the FuDGE score by the average conversation length in the dialogue corpus, as the maximum score one can get is from an empty graph by inserting every utterance. The **F**low-**F**1 (FF1) is:

$$FF1 = \frac{2(1 - nc) \times (1 - nf)}{(1 - nc) + (1 - nf)} \quad (5)$$

where nc and nf are normalized complexity and normalized average FuDGE score (Equation 4).

4 Fuzzy Dialogue-Graph Edit Distance

Motivated from the Levenshtein distance, we focus on aligning a flow path with a dialogue. This is particularly challenging as we need to match a given utterance in the dialogue with an intent in the flow path. Intuitively, an utterance is a good match with an intent if it is semantically close to the majority of the utterances associated with the intent.

More precisely, given a dialogue flow $G = (V, E)$, its set of flow paths $F_G = \{P^k = \langle G_r P_1^k P_2^k ... P_{n_k}^k \rangle\}_K$, and a conversation $C^i = \langle u_1^i u_2^i \dots u_m^i \rangle$, we start with finding the edit distance between $C^i = \langle u_1^i u_2^i \dots u_m^i \rangle$ and a specific flow path $P^j = \langle G_r P_1^j P_2^j ... P_{n_j}^j \rangle$. Conversation C^i is a sequence of utterances $\langle u_1^i u_2^i \dots u_m^i \rangle$ produced by a set of actors $\langle a_1^i a_2^i \dots a_m^i \rangle$, and flow path $P^j = \langle G_r P_1^j P_2^j ... P_{n_j}^j \rangle$ is a sequence of nodes, where each node P_r^j is associated with an intent bucket $B^r = (actor, utterances)$. The dialogue flow path edit distance follows the logic described in the preliminaries section. It is similar to Equation 1, except that we need to define the substitution cost between an utterance and an intent. Once the distance between a single flow path and a dialogue is determined, the FuDGE distance can be computed using the formula in Equation 3.

4.1 Fuzzy Substitution Cost

To match an intent with an utterance, we need to ensure that the utterance is semantically similar to the utterances in the intent bucket. If the utterance and the intent are not similar, the intent should be replaced. However, we cannot simply replace an intent with an utterance. Instead, we propose to replace the intent with the nearest intent to the utterance in the set of universal intent buckets B. Intuitively, if the current node intent is dissimilar to u, it should be replaced with the most similar intent to u.

To find the most similar intent, we define B^* as the intent closest to the utterance. If the current node intent is dissimilar to the utterance, it

will be replaced with the most similar intent to the utterance. We define the substitution cost as the sum of two distances: the distance between the current intent bucket and the utterance ($d_1(B^r, u)$), and the distance between the current intent bucket and the most similar intent bucket ($d_2(B^r, B^*)$). The final substitution cost is then defined as the weighted sum of the two distances, with the weight determined by the hyperparameter α:

$$cost_{sub}(B^r, u) = \alpha(d_1(B^r, u) + d_2(B^r, B^*)) \quad (6)$$

Where α is set to 0.5 here to keep the substitution cost between 0 and 1.

We define the intent-utterance and intent-intent distance as a function of their distance in a semantic space. Namely we encode an utterance u and the intent utterances $B^r.utterances = \{u_1^r, \ldots, u_l^r\}$ into distributional representations e_u, and $\{e_1, e_2, \ldots, e_l\}$ using Sentence Bert Encoder (Reimers and Gurevych, 2019). We use the intent centroid as a representation for the intent, obtained by taking the average of the embeddings of all the utterances in the intent, i.e. $e_{B^r} = \frac{1}{l}\sum_{j=1}^{l} e_1$.

We define intent-utterance distance in two ways: (i) by calculating the cosine distance between an utterance embedding and an intent centroid, and (ii) by calculating the cosine distance between the utterance embedding and the nearest utterance within the intent. The intent-intent distance is calculated using the cosine distance between the centroids of two intents. A detailed explanation of these mathematical formulations can be found in the Appendix.

Actor Alignment. To ensure that an utterance produced by a user is not matched with an intent associated with an agent, and vice versa, we set the intent-intent and intent-utterance distance to ∞ if the actors mismatch. For example, we calculate the intent-intent distance, $d_2(B^r, B^s)$, using the cosine similarity between the embedding vectors e_{B^r} and e_{B^s}, but only if the actors for the two intents match. If the actors do not match, we set the intent-intent distance to ∞.

4.2 Efficient FuDGE Implementation & Complexity Analysis

The original edit distance algorithm uses dynamic programming to compute the Levenstein distance. In dynamic programming, the computed solutions to subproblems (e.g., $d_{i,j}$ in Equation 1) are stored in a memoization table so that these don't have to be recomputed. Even with a memoization table, computing the formula in Equation 3 results in a $\mathcal{O}(TKm)$ running time if we separately compute the distance between every single path in the flow $F_G = \{P^k = \langle G_r P_1^k P_2^k ... P_{n_k}^k \rangle\}_K, n_k \leq T$ and a conversation of length m. Calculating the distance between a dialogue corpus of size N with a flow graph (Equation 4) results in running time complexity of $\mathcal{O}(TKNm)$. This makes it impractical to calculate the distance between complex dialogue flows with many paths and large dialogue corpora.

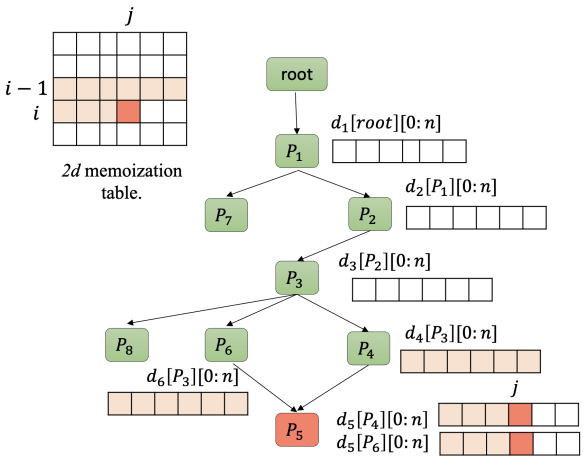

Figure 2: Efficient memoization used for FuDGE

To optimize the memoization process, we utilize the structure of a flow DAG. Our approach is based on the idea that paths in a dialogue graph often share overlapping sub-paths, which can be leveraged to improve efficiency. For example, in Figure 2, node P_3 has two children, P_4 and P_6, and the two paths stemming from it, `root->...->`P_3`->`P_4 and `root->...->`P_3`->`P_6 share the same prefix `root->...->`P_3, up to the node P_3. As a result, both paths require the same memoization information for calculating case 2 and 3 in Equation 1. Instead of using a regular memoization table, we keep an array for each node that contains the edit distance between a conversation and all paths that start from the root and end at that node.

Our approach combines *dfs*-traversal with memoization, using the parent edit distance array for the calculation in the current node. It is also important to note that a node may have multiple distance arrays, since multiple paths may end at a certain node (e.g., node P_5 in Figure 2). We compute the distance between a dialogue and all paths in the dialogue flow by taking the entire DAG into consideration, rather than looking at each path separately. The complexity of computing the FuDGE distance

between all flow paths and a conversation with length n is $\mathcal{O}((|V| + |E|)n)$. The implementation details of both the naive and efficient algorithms are provided in Algorithm 1 and Algorithm 2 in the Appendix. By exploiting the overlapping sub-paths in a flow DAG, we can optimize the memoization process and improve the efficiency of our algorithm.

5 Experiments

In this section, we explain the automatic flow discovery techniques, dialogue corpora, and evaluation datasets used to evaluate the performance of FuDGE and *FF1*.

5.1 Flow Discovery Methods

We utilized two existing automatic discovery algorithms to obtain dialogue flows from a dialogue corpus, as there are only a few algorithms available for this purpose. One of the discovery methods we used is the NFA-based discovery method proposed in (Martínez and Nugent, 2022), denoted as `ALG1`. Additionally, we used another proprietary software, currently unpublished, denoted as `ALG2`. Both algorithms make use of a set of user and agent intents, and can use fully annotated intents when available. In the absence of human-annotated intents, an intent discovery method is employed to find intent clusters. The `ALG1` method only requires agent intents initially, and user utterances are grouped as a by-product of graph minimization. In this work, we generated dialogue flows both with and without human-annotated intents, which we refer to as supervised and unsupervised flows, respectively. Both discovery algorithms utilize a sentence encoder for intent discovery and intent detection, and we used the same encoding method for consistency across both algorithms. The intent-intent and intent-utterance distances were also computed using this same encoder.

5.2 Datasets

In this work, we use two datasets, each consisting of a set of dialogues, where each dialogue is a conversation between two **actors** (i.e., user and agent) that consists of a sequence of **turns**. A turn is an utterance produced by one of the actors. Ideally, the best way to evaluate our framework is to compare a set of manually crafted gold flows, perfect in both coverage and compression, with automatically discovered flows from the dialogue corpus. To our knowledge, there is no public dataset with gold standard flows. Therefore, we propose to impose a level of supervision with human-annotated utterances. Many dialogue state tracking datasets (Williams et al., 2014; Bouraoui et al., 2019; Tian et al., 2021; Qi et al., 2022) have human annotated dialogue act utterances, none of which have fully annotated user intents. A dialog act is an utterance that serves as a function in the dialog, such as a question, a statement, or a request. In contrast, intents are more fine-grained and categorize a user intention. For example, "`I want to book a hotel room`" and "`I would like to order pizza`" have the same dialogue act `request` while their intents are different. Our first dataset, **Finance**, is a dataset with fully annotated agents and user intents. It constitutes the conversations between a user and the customer service of a financial agency. Our second dataset is created from **STAR** (Mosig et al., 2020), which is the only publicly available dialogue state tracking dataset partially annotated with agent intents. **STAR** is a schema-guided task-oriented dialog dataset consisting of 127,833 utterances across 5,820 task-oriented dialogs in 13 domains, from which we pick two domains, "`Hotel`" and "`Bank`" since they contain the most number of dialogues. We processed the dialogues in the STAR dataset and removed those with unlabeled agent utterances. Table 1 contains our datasets' statistics, including the number of dialogues, utterances, tasks, and intents. A complete list of tasks for each dataset is provided in the Appendix.

	FINANCE	STAR
Conversations	447	527
Utterances	7392	7352
Tasks	12	5
Agent intents	55	41
User intents	47	-

Table 1: Dataset Statistics

5.3 FuDGE Evaluation

This experiment aims to evaluate the effectiveness of FuDGE as a distance metric. More specifically, given a dialogue flow created for a specific task, a good distance metric should provide lower scores for conversations that belong to the task than the out-of-task conversations. We picked "`Make Payment`" task from the Finance dataset with 150 conversations and "`Bank Fraud Report`" and "`Hotel Book`" from the STAR dataset, with 180

	Make Payment		Hotel Book		Bank Report Fraud	
Model	Positves	Negatives	Positves	Negatives	Positves	Negatives
ALG1-Centroid	0.27 ± 0.12	0.51 ± 0.13	0.40 ± 0.21	0.63 ± 0.22	0.42 ± 0.18	0.67 ± 0.22
ALG1-Min	0.21 ± 0.12	0.47 ± 0.13	0.34 ± 0.20	0.60 ± 0.21	0.33 ± 0.18	0.61 ± 0.22
ALG2-Centroid	0.14 ± 0.03	0.48 ± 0.16	0.08 ± 0.03	0.59 ± 0.18	0.09 ± 0.04	0.63 ± 0.19
ALG2-Min	0.08 ± 0.03	0.44 ± 0.16	0.04 ± 0.04	0.57 ± 0.19	0.02 ± 0.04	0.60 ± 0.20

Table 2: Average FuDGE score for within-task (Positives) vs out-of-task (Negative) conversations, indicating a clear separation.

and 145 conversations. We generated separate dialogue flows for each of these tasks using `ALG1` and `ALG2`. For each task, we also randomly sampled 50% of the in-task conversations and added the same number of out-of-task conversations. We evaluated each task-flow with the corresponding dialogue corpus and obtained the average FuDGE score for each dialogue corpus. The results are summarized in Table 2. The average score for within-task dialogues (positives) is significantly smaller than the out-of-task (negatives) dialogues. It can segregate within-task conversations from out-of-task conversations. These results suggest that FuDGE is an effective distance metric that can be used independently in any application involving the distance between a dialogue and any predefined DAG structured dialogue scheme. The Appendix provides examples of the dialogues and the best-matched flow path.

5.4 Parameter Optimization With *FF1*

Automatic flow discovery usually involves multiple steps, including clustering the utterances, creating the graph, ranking important paths, and pruning the graph accordingly. Each of these steps may add different hyperparameters to the entire discovery pipeline. The simplest clustering algorithms, such as K-means, require k as the number of clusters. While hyperparameter selection can drastically impact the quality of the final discovered flows, it has been done mainly by manual trial and error. In this experiment, we show that the *FF1* score is a practical framework for choosing the optimal set of hyperparameters. For this experiment, we use `ALG1` as the flow discovery algorithm. This algorithm consists of a ranking strategy that ranks the paths based on their importance and later keeps the k top-ranked paths as the discovered flow. In both supervised and unsupervised setup, we run `ALG1` task over the `Make Payment` task from the Finance dataset. We generate multiple flows for different values of k ranging from 1 to the maximum number of paths in the complete graph. The top row of Figure 3 shows the normalized FuDGE score obtained for each flow versus k. First, we can see the clear segregation between the within-task and out-of-task dialogues. Moreover, as more paths get added, the FuDGE score asymptotically decreases to a minima. The middle row depicts the normalized complexity score for different k values, which indicates a monotonic increase in the complexity, which is expected as we add more paths, and, thus, more nodes to the graph. The bottom row is the final *FF1* score. We see that the scores go up to an optimal point as we add more paths, but it starts declining. The peak in the graph is almost aligned with the point where the FuDGE score starts to stay constant.

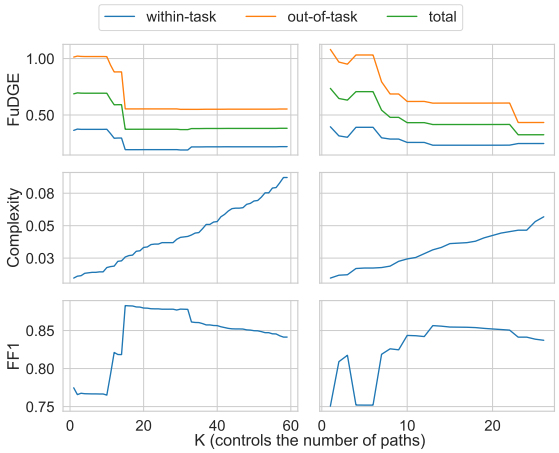

Figure 3: Parameter tuning with *FF1* for `ALG1` and "`Make Payment`" task. The left column is the scores from an unsupervised discovered flow, and the right column corresponds to the supervised flow. The optimal k is smaller for the supervised flow, indicating a better compression.

5.5 *FF1* Evaluation

In this section, we compare dialogue flows with and without annotated intents. The discovery algorithms need to use a clustering method to generate the intents without the labeled data. This process imposes some noise on the flow discovery, lead-

	STAR						FINANCE					
	Supervised			Unsupervised			Supervised			Unsupervised		
Model	FF1	FuDGE	Complexity	FF1	FuDGE	Complexity	FF1	FuDGE	Complexity	FF1	FuDGE	Complexity
ALG1-Min	0.75	0.27	0.23	**0.79**	0.24	0.18	**0.86**	0.21	0.07	0.81	0.23	0.15
ALG1-Centroid	0.71	0.35		**0.73**	0.34		**0.82**	0.27		0.78	0.27	
ALG2-Min	0.59	0.08	0.57	**0.73**	0.26	0.28	**0.90**	0.03	0.16	0.77	0.03	0.36
ALG2-Centroid	0.58	0.12		**0.71**	0.34		**0.87**	0.09		0.75	0.09	

Table 3: Results of FF1 flow comparison between supervised and unsupervised discovered flow.

ing to lower quality dialogue flows. This experiment aims to show that our evaluation approach can capture this phenomenon. We generated complete flows by running `ALG1` and `ALG2` over the entire dialogue corpus for both datasets. Then, we calculated the average FuDGE score and the complexity of each discovered flow and computed the *FF1* score. Table 3 summarizes the results of this experiment. As shown, for the Finance dataset, the score for the supervised discovered flows is higher than the unsupervised discovered flows, with only one exception. However, the *FF1* score for the unsupervised flows discovered by `ALG2` is significantly higher than the supervised flows. A manual investigation of the flows showed that the annotated labels were too fine-grained. Clustering led to more high-level intents, which eventually processed better-quality dialogue flows. We provide more discussion about this case in the Appendix.

6 Related Work

Related work for our work is relatively sparse. Although automatic evaluation of dialogue systems is an active field of research (Yeh et al., 2021; Khalid and Lee, 2022), most of the metrics and approaches focus on evaluating a dialogue in utterance level (Sun et al., 2021; Ghazarian et al., 2020). However, our work focuses on the evaluation of the dialogues in conversation level, mostly produced by an AI algorithms, such as Graph2Bot introduced by Bouraoui et al. (2019) and is a tool for assisting conversational agent designers. It could extract a graph representation from human-human conversations using unsupervised learning. More recently, (Qiu et al., 2020) used a Variational Recurrent Neural Network model with discrete latent states to learn dialogue structure in an unsupervised fashion. They evaluate their method by using Structure Euclidean Distance (SED) and Structure Cross-Entropy (SCE) based on the transition probabilities between nodes but found them to be unstable. SED and SCE also do not consider the semantic similarity between the node and the original conversation.

Word Confusion Networks (WCNs) (Mangu et al., 2000) has been used extensively to model the hypothesis of automatic speech recognition (ASR). Just like the dialogue flows, WCNs can also be represented as DAGs. A popular metric for identifying the quality of ASR has been word error rate which incorporates ideas of edit distance that can be derived through each path in the WCN that represents an ASR hypothesis. Lavi et al. (2021) introduced the notion of using edit distance (Wagner and Fischer, 1974) for dialog-dialog similarity. In their work, they used sentence-level embeddings (Cer et al., 2018; Reimers and Gurevych, 2019) to determine the similarity between two utterances within a dialogue and defining the edit distance substitution cost.

7 Conclusion

This paper presents a novel evaluation framework for a crucial task necessary for building task-oriented dialogue agents. This framework can be used with any flow discovery method and dialogue corpora as long as the generated dialogue flows can be represented as a DAG. We introduced the *FF1* metric, a harmonic mean of flow complexity and FuDGE distance, and demonstrated its efficacy as a tool to select hyperparameters of a flow discovery algorithm or process. We envisage it to be a useful guide for human conversational designers or as a measure to optimize an automatic flow discovery process. We also propose an efficient implementation of FuDGE distance with $\mathcal{O}((|V| + |E|)n)$, allowing it to scale to large datasets. This approach delivers a consistent baseline, thereby better versioning and tracking the progress of flows and corresponding automation with time. In the future, we hope to incorporate utterance characteristics for the insertion and deletion cost to account for the actual semantic cost of the operation.

8 Limitations

In this section, we discuss the limitations of our approach. Our implementation in FuDGE only assigns deletion and insertion costs as 0 or 1, without taking into account the characteristics of the utterance or intent. Additionally, we acknowledge that the current unavailability of the Finance dataset and a detailed description of the second discovery method limits the reproducibility of our work. However, we are optimistic that our work will provide valuable insights and lead to future research in this area.

References

Jean-Leon Bouraoui, Sonia Le Meitour, Romain Carbou, Lina M. Rojas Barahona, and Vincent Lemaire. 2019. Graph2Bots, unsupervised assistance for designing chatbots. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 114–117, Stockholm, Sweden. Association for Computational Linguistics.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

George Forman, Hila Nachlieli, and Renato Keshet. 2015. Clustering by intent: a semi-supervised method to discover relevant clusters incrementally. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 20–36. Springer.

Sarik Ghazarian, Ralph Weischedel, Aram Galstyan, and Nanyun Peng. 2020. Predictive engagement: An efficient metric for automatic evaluation of open-domain dialogue systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7789–7796.

Baber Khalid and Sungjin Lee. 2022. Explaining dialogue evaluation metrics using adversarial behavioral analysis. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5871–5883.

Ofer Lavi, Ella Rabinovich, Segev Shlomov, David Boaz, Inbal Ronen, and Ateret Anaby-Tavor. 2021. We've had this conversation before: A novel approach to measuring dialog similarity. *arXiv preprint arXiv:2110.05780*.

Vladimir I Levenshtein et al. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union.

Ting-En Lin, Hua Xu, and Hanlei Zhang. 2020. Discovering new intents via constrained deep adaptive clustering with cluster refinement. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8360–8367.

Lidia Mangu, Eric Brill, and Andreas Stolcke. 2000. Finding consensus in speech recognition: word error minimization and other applications of confusion networks. *Computer Speech & Language*, 14(4):373–400.

Javier Miguel Sastre Martínez and Aisling Nugent. 2022. Inferring ranked dialog flows from human-to-human conversations. In *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 312–324.

Johannes EM Mosig, Shikib Mehri, and Thomas Kober. 2020. Star: A schema-guided dialog dataset for transfer learning. *arXiv preprint arXiv:2010.11853*.

Hugh Perkins and Yi Yang. 2019. Dialog intent induction with deep multi-view clustering. 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)*, pages 4016–4025, Hong Kong, China. Association for Computational Linguistics.

Jiexing Qi, Jingyao Tang, Ziwei He, Xiangpeng Wan, Chenghu Zhou, Xinbing Wang, Quanshi Zhang, and Zhouhan Lin. 2022. Rasat: Integrating relational structures into pretrained seq2seq model for text-to-sql. *arXiv preprint arXiv:2205.06983*.

Liang Qiu, Yizhou Zhao, Weiyan Shi, Yuan Liang, Feng Shi, Tao Yuan, Zhou Yu, and Song-Chun Zhu. 2020. Structured attention for unsupervised dialogue structure induction. *arXiv preprint arXiv:2009.08552*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Chen Shi, Qi Chen, Lei Sha, Sujian Li, Xu Sun, Houfeng Wang, and Lintao Zhang. 2018. Auto-dialabel: Labeling dialogue data with unsupervised learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 684–689, Brussels, Belgium. Association for Computational Linguistics.

Weiwei Sun, Shuo Zhang, Krisztian Balog, Zhaochun Ren, Pengjie Ren, Zhumin Chen, and Maarten de Rijke. 2021. Simulating user satisfaction for the evaluation of task-oriented dialogue systems. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2499–2506.

Xin Tian, Liankai Huang, Yingzhan Lin, Siqi Bao, Huang He, Yunyi Yang, Hua Wu, Fan Wang, and Shuqi Sun. 2021. Amendable generation for dialogue state tracking. *arXiv preprint arXiv:2110.15659*.

Robert A Wagner and Michael J Fischer. 1974. The string-to-string correction problem. *Journal of the ACM (JACM)*, 21(1):168–173.

Jason D Williams, Matthew Henderson, Antoine Raux, Blaise Thomson, Alan Black, and Deepak Ramachandran. 2014. The dialog state tracking challenge series. *AI Magazine*, 35(4):121–124.

Yi-Ting Yeh, Maxine Eskenazi, and Shikib Mehri. 2021. A comprehensive assessment of dialog evaluation metrics. *arXiv preprint arXiv:2106.03706*.

Hanlei Zhang, Hua Xu, Ting-En Lin, and Rui Lyu. 2021. Discovering new intents with deep aligned clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 14365–14373.

Yuwei Zhang, Haode Zhang, Li-Ming Zhan, Xiao-Ming Wu, and Albert Lam. 2022. New intent discovery with pre-training and contrastive learning. *arXiv preprint arXiv:2205.12914*.

A Mathematical Formulations

This section provides the exact mathematical formulation to calculate the fuzzy substitution cost. For the representation of an intent bucket, we average the embedding of the utterances in the bucket.

$$e_{B^r} = \frac{1}{|B^r.utterances|} \sum_{u^j \in B^r.utterances} e_{u^j} \quad (7)$$

The distance between an intent B^r and utterance u is d_1 and is defined as:

$$d_1(B^r, u) = \begin{cases} \phi(B^r, u) & B^r.actor = u.actor \\ \infty, & \text{otherwise} \end{cases} \quad (8)$$

Where ϕ is a function of intent and utterance embeddings and is defined in two ways:

1. **Min**. The minimum distance between u and the nearest utterance in $B^r.utterances$.

$$\phi(B^r, u) = \min_{u^j \in B^r.utterances} cosine(e_{u^j}, e_u)$$

2. **Centroid**. The cosine distance between e_{B^r} and e_u.

$$\phi(B^r, u) = cosine(e_u, e_{B^r})$$

While the **Min** approach often produces a smaller FuDGE score, and therefore larger *FF1* (Table 2, Table 3), the **Centroid** method is more robust to the effect of outliers in the intent clusters. For the intent-intent distance d_2 we use the formula in Equation 7 to obtain two intent representation, d_2 then is:

$$d_2(B^r, B^s) = \begin{cases} \varphi(B^r, B^s) & B^r.actor = B^s.actor \\ \infty, & \text{otherwise} \end{cases} \quad (9)$$

Where $\varphi(B^r, B^s)$ is defined as the cosine distance between the bucket representations e_{B^r} and e_{B^s}.

B Dataset Detail

Table 4 contains the dataset tasks and the number of conversations per task. The tasks with the most number of conversations are selected for the second and third sections in the experiments section. We are publishing the STAR dataset, including the dialogue corpora used for flow discovery and evaluation, and the discovered flows from the STAR dialogue corpus using `ALG1` and `ALG2`.

Finance	
`Make Payment`	150
`Investigate Charges`	66
`Check Card Balance`	51
`Credit Limit Request`	36
`Replace Card`	36
`Loan Status Request`	24
`Lock Card Request`	24
`Lost Card`	12
`Activate Card`	12
`Lost Card`	12
`Unlock Card`	12
`Card Status`	12
STAR	
`Bank Fraud Report`	180
`Hotel Book`	145
`Hotel Search`	108
`Bank Balance`	43
`Hotel Service Request`	31

Table 4: Tasks in each datasets with the number of conversation within each task

C Implementation Detail

In 1, we provide a detailed explanation of the naive implementation of the FuDGE score. The FUZZYEDITDISTANCE function calculates the edit distance between a single dialogue path and a given dialogue. As a result, the naive implementation calculates the edit distance score separately for each pair of dialogue and flow path. In contrast, the efficient algorithm described in Section 4, as demonstrated in 2, combines memoization with *dfs*-traversal. The experiments were conducted using a `CPU` with a `3.5 GHz Dual-Core Intel Core i7` and `16 GB 2133 MHz LPDDR3` memory. We conducted runtime analysis for two tasks. The first task, "`Bank Fraud Report`", involved 180 conversations with an average conversation length of 15.6 (number of sentences) that generated a DAG with 1635 nodes. The total runtime for this task was 4 minutes and 41 seconds, and the average runtime per conversation was 1.56 seconds. For the second task, "`Hotel Book`", we used 144 conversations with an average conversation length of 14.9 (number of sentences), resulting in a DAG with 1309 nodes. The total runtime for this task was 3 minutes and 3 seconds, and the average runtime per conversation was 1.27 seconds.

Algorithm 1: Naive Fudge

input: C^i (Dialogue); $F_G = \{P^k = rP_1^kP_2^k...P_{n_k}^k\}_k$ (A set of dialogue flow paths); FuzzyEditDistance (function: returns the fuzzy edit distance between a dialogue and a single path)

```
def NaiveFUDGE(C^i, F_G):
    min_dist ← ∞
    for j ← 1 to k do
        dist ← FuzzyEditDistance(C^i, P^j)
        if min_dist < d then
            min_dist ← dist
    return min_dist
```

D Flow Investigation

In this section, we provide a more detailed discussion of the results in Table 3 and the reason why for the STAR dataset, the ff1 score is higher for the unsupervised flows compared to the supervised flows. After a manual investigation, we concluded that the agent labels were too fine-grained. Table 5 is providing an example of this scenario. The three intents in the supervised setup are clustered together by the clustering algorithm in the unsupervised setup. This is often the case that these three intents occur in a different order within a conversation; therefore, having one intent instead of three results in a more compact denoised flow. In the supplementary material in folder `data/nfa_flows`, we are providing a visualization of the flows, generated by `ALG2`. We provide flow visualization for both supervised and unsupervised settings referred to as `labeled` and `unlabeled`.

E Examples Of Matched Paths

This section provides some examples of conversations, the path picked up by the FuDGE algorithm, and the operations and costs needed to convert a path to a conversation. The best-matched path to a conversation is a path selected from the paths that start from the root node, end at one of the leaf nodes, and have the lowest FuDGE score. To find the best-matched paths, we look into the distance array kept for each leaf node at `node2dist` map. (refer to Algorithm 2, and we peak the node with the smallest edit distance at `node2dist[leaf][-1]`. After we picked the leaf node with the smallest distance, we tracked the path by reversing the steps from the child node to the parent node until the root node. Table 6 contains examples of three conversations with their best-matched paths. More examples can be found in the supplementary directory `alignment`.

Algorithm 2: Efficient Fudge

input: $C^i = u_1^iu_2^i...u_n^i$ (Dialogue); $G = (V, E)$ (A set of dialogue flow paths); node2dist ← {} (A map of dialogue flow nodes to a list of tuples (l, d) with l being path length from root to the node and d the path-conversation edit distance.); $r \leftarrow G.r$ (Current node in dfs traversal); $p \leftarrow$ NaN (Current node's parent in dfs traversal)

```
def FUDGE(C, G, r, p, node2dist):
    DFSEditDistance(C,G,G.r, NaN, node2dist)
    min_dist ← FindMinPath(G, node2dist)
    return min_dist

def DFSEditDistance(C, G, r, p, node2dist):
    if p = NaN then
        dist ← []
        n ← len(C)
        for i ← 1 to n + 1 do
            dist ← dist + [i]
        node2dist[r] ← [(0, dist)]
    else
        for l, d in node2dist[p] do
            dist ← [l + 1]
            for u ← u_1^i to u_n^i do
                dist ← min(d[i + 1] +
                  del_cost(r.intent), dist[−1]+
                  insert_cost(u), d[i] +
                  subs_cost(r.intent, u))
            node2dist[r] ←
              node2dist[r] + [(l + 1, dist)]
        for c in r.children do
            DFSEditDistance(C,G,c,r, node2dist)
    return
```

Unsupervised Intent	Supervised Intent	Utterances
	bank_ask_mothers_maiden_name	What was your mother's maiden name?
date birth	bank_ask_dob	Could you provide your date of birth, please?
	bank_ask_childhood_pet_name	And what was the name of the pet you had as a child?

Table 5: Tasks in each dataset with the number of conversations within each task

Conversation	Path Intent Names	Operations	Cost
u.Hi there, I need to reserve a hotel room!	Reserve Hotel Room	replace	0.346
a.What hotel would you like to stay at?	Hotel Like Stay	replace	0.346
u.Good question. I wanted to say the Hilton, but my friend recommends the Old Town Inn, so lets try that	Town Inn	replace	0.407
a.When are you arriving?	Arriving Arriving	replace	0.407
u.12-May	May 12 Arrive	replace	0.513
a.When will you be leaving again?	Leaving Leaving	replace	0.513
u.Actually never mind the Old Town Inn, my personal favorite blog says the Hyatt is the bees knees. Let's do that instead	Hyatt Bees Knees	replace	0.513
a.When will you be leaving again?	When Will You	replace	0.513
u.Oh yeah the 24th, this blog is the bomb!	24Th Blog Bomb	replace	0.513
a.May I have your name, please?	May Name Please	replace	0.540
u.Would you believe this.... my wife just sent me a text saying my brother in law is getting married in London, ironically on the 24th... so scratch this month and lets do the 8th to the 26th next month.	Getting Married London	replace	0.540
a.May I have your name, please?	May Name Please	replace	0.566
u.Oh yeah sorry Ben with a B	Yeah Sorry Ben	replace	0.566
a.Alright, the Hyatt Hotel ticks all of your boxes, can I book this room for you?	Alright Hyatt Hotel	replace	0.566
u.Yes please. Let's be honest here nobody really likes weddings right?	Really Likes Weddings	replace	0.566
a.OK, I've successfully completed this Hotel booking for you!	Successfully Completed Hotel	replace	0.566
u.Ok great thanks a lot	Ok Great Thanks	replace	0.886
a.Hello I need to reserve a room. My friend is having a big party.	town inn	replace	0.383
u.Hello	Hello Hello Hello	replace	0.387
a.Hello, how can I help?	Hello Help	replace	0.387
u.I need to reserve a room. My friend is having a big party.	Want Resevation	replace	0.959
a.May I have your name, please?	May Name Please	replace	0.985
u.Angela	John Angela Alexis	replace	1.259
a.What hotel would you like to stay at?	Hotel Like Stay	replace	1.259
u.Old Town Inn is my favorite. Hopefully it is available.	Hilton Hyatt Hyatt	replace	1.826
a.When are you arriving?	Arriving Arriving Arriving	replace	1.826
u.May 8th. It is also my birthday. I am a stubborn Taurus.	Arriving 11Th	replace	2.319
a.When will you be leaving again?	When Will You	replace	2.319
u.May 23rd I will be leaving.	Request Extra Towels	replace	2.759
a.Do you have any special requests?	Do you have any special requests?	insert	3.759
u.No. I am a simple earth sign.	No. I am a simple earth sign.	insert	4.759
a.I'm very sorry, but there is no room available at the Old Town Inn for your requested dates.	Hotels Match Search	replace	5.298
u.That is okay Thanks for trying. Goodbye.	Birth Hospital Goodbye	replace	5.893
a.Thank you and goodbye.	Thank Goodbye	replace	5.893

Table 6: Alignment of a conversation with a flow path. Intent names are generated with NGrams from the a cluster utterance set