LIDA: Lightweight Interactive Dialogue Annotator

Abstract

Dialogue systems are highly dependent on the quality of the data used to train them. It is therefore important to develop good dialogue annotation tools which can improve the speed and quality of dialogue data annotation. With this in mind, we introduce LIDA, an annotation tool designed specifically for conversation data. As far as we know, LIDA is the first dialogue annotation system that handles the entire dialogue annotation pipeline from raw text, as may be the output of transcription services, to structured conversation data. Furthermore it supports the integration of arbitrary machine learning models as annotation recommenders and also has a dedicated interface to resolve inter-annotator disagreements such as after crowdsourcing annotations for a dataset. LIDA is fully open source, documented and publicly available.

1 Introduction

Dialogue systems are becoming one of the most active research areas in Natural Language Processing (NLP) and Machine Learning (ML). Creating a high-quality dialogue dataset incurs a large annotation cost, which makes good dialogue annotation tools essential to ensure the highest possible quality. Many annotation tools exist for a range of NLP tasks but none are designed specifically for dialogue with modern usability principles in mind.

LIDA is a web application designed to make dialogue dataset creation and annotation as easy and fast as possible. In addition to following modern principles of usability, LIDA integrates best practices from other state-of-the-art annotation tools, most importantly by allowing arbitrary ML models to be integrated as annotation recommenders to suggest annotations for data. Any system with the correct API can be integrated into LIDA's back end, meaning LIDA can be used as a front end for researchers to interact with their dialogue systems and correct their responses, then save the interaction as a future test case.

When data is crowdsourced, it is good practice to have multiple annotators label each piece of data to reduce noise and mislabelling. Once you have multiple annotations, it is important to be able to resolve conflicts by highlighting where annotators disagreed so that an arbiter can decide on the correct annotation. To this end, LIDA provides a dedicated interface which automatically finds where annotators have disagreed and displays the labels alongside a percentage of how many annotators selected each label, with the majority annotated labels selected by default.

1.1 Main Contributions

Our main contributions with this tool are:

- A modern annotation tool designed specifically for task-oriented conversation data
- The first dialogue annotator capable of handling the full dialogue annotation pipeline from turn and dialogue segmentation through to labelling structured conversation data
- Easy integration of dialogue systems and recommenders to provide annotation suggestions
- A dedicated interface to resolve inter-annotator disagreements for dialogue data

2 Related Work

Various annotation tools have been developed for NLP tasks in recent years. Table 1 compares LIDA with other recent annotation tools. TWIST is a dialogue annotation tool which consists of two stages: turn segmentation and content feature annotation. Turn segmentation allows users to highlight and create new turn segments from raw text. After this, users can annotate sections of text in a segment by highlighting them and selecting from a predefined feature list. However, this tool doesn't allow users to specify custom annotations or labels and doesn't support classification or slot-value annotation.

INCEpTION is a semantic annotation platform for interactive tasks that require semantic resources like entity linking. It provides machine learning models to suggest annotations and allows users to collect and model knowledge directly in the tool. GATE is an

Table 1: Annotator Tool Comparison Table

Annotation Tool	Turn/Dialogue Segmentation	Classification Labels	Edit Dialogues/Turns	Recommenders	Inter-Annotator Disagreement Resolut
LIDA	YES	YES	YES	YES	YES
INCEpTION	NO	YES	NO	YES	YES/NO
GATE	NO	YES	NO	NO	YES/NO
TWIST	YES	NO	YES	NO	NO
BRAT	NO	YES	NO	YES	NO
DOCCANO	NO	YES	NO	NO	NO
DialogueView	YES	YES	YES	NO	NO

open source tool that provides predefined solutions for many text processing tasks. It is powerful because it allows annotators to enhance the provided annotation tools with their own Java code, making it easily extensible and provides an enormous number of predefined features. However, GATE is a large and complicated tool with a significant setup cost. Despite their large feature sets, INCEpTION and GATE are not designed for annotating dialogue and cannot display data as turns, an important feature for dialogue datasets.

BRAT and Doccano are web-based annotation tools for tasks such as text classification and sequence labeling. They have intuitive and user-friendly interfaces which aim to make the creation of certain types of dataset such as classification or sequence labelling datasets as fast as possible. BRAT also supports annotation suggestions by integrating ML models. However, like INCEpTION and GATE, they are not designed for annotating dialogues and do not support generation of formatted conversational data from a raw text file such as may be output by a transcription service. LIDA aims to fill these gaps by providing a lightweight, easy-to-setup annotation tool which displays data as a series of dialogues, supports integration of arbitrary ML models as recommenders and supports segmentation of raw text into dialogues and turns.

DialogueView is a tool for dialogue annotation. However, the main use-cases are not focused on building dialogue systems, rather it is focused on segmenting recorded conversations. It supports annotating audio files as well as discourse segmentation - hence, granular labelling of the dialogue, recommenders, inter-annotator agreement, and slot-value labelling is not possible with DialogueView.

3 System Overview

LIDA is built according to a client-server architecture with the front end written in standard web languages (HTML/CSS/JavaScript) that will run on any browser. The back end written in Python using the Flask web framework as a RESTful API.

The main screen which lists all available dialogues. The buttons below this list allow a user to add a blank or formatted dialogue file. Users can also drag and drop files in this screen to upload them. The user is then able to add, delete or edit any particular dialogue. There is also a button to download the whole dataset as a JSON file on this page. Clicking on a dialogue will take users to the individual dialogue annotation screen.

LIDA uses the concept of a “turn” to organise how a dialogue is displayed and recorded. A turn consists of a query by the user followed by a response from the system, with an unlimited number of labels allowed for each user query. The user query and system response are displayed in the large area on the left of the interface, while the labels for each turn are shown in the scrollable box on the right. There are two forms that these labels can currently take which are particularly relevant for dialogue: multilabel classification and slot-value pair.

An example of multilabel classification is whether the user was informing the system or requesting a piece of information. An example of a slot-value pair is whether the user mentioned the type of restaurant they’d like to eat at (slot: restaurant-type) and if so what it was (value: italian, for example). The front-end code is written in a modular form so that it is easy for researchers

3.0.1 Experimenting with Dialogue Systems

LIDA is designed with this in mind - a dialogue system can be integrated into the back end so that it will run whenever the user enters a new query in the front end. The user will then be able to evaluate whether the system gave the correct answer and correct the labels it gets wrong using the front end. LIDA will record these corrections and allow the user to download the interaction with their dialogue system with the corrected labels so that it can be used as a test case in future versions of the system.

3.0.2 Creating a New Dialogue Dataset

Users can create a blank dialogue on LIDA’s home screen, then enter queries in the box shown at the bottom of the screen. Along with whole dialogue systems, arbitrary ML models can be added as recommenders in the back end. Once the user hits "Enter", the query is run through the recommender models in the back end and the suggested annotations displayed for the label. If no recommender is specified in the back end, the label will be left blank. Users can delete turns and navigate between them using

"Enter" or the arrow keys. The name of the dialogue being annotated can be seen next to the "Back" button at the top left of the screen and can be edited by clicking on it.

3.0.3 Annotating An Existing Dataset

Datasets can be uploaded via drag-and-drop to the home screen of the system, or paths can be specified in the back end if the system were being used for crowdsourcing. Datasets can be in one of two forms, either a ".txt" file such as may be produced by a transcription service, or a formatted ".json" file, a common format for dialogue data. Once the user has uploaded their data, their dialogue(s) will appear on the home screen. The user can click on each dialogue and will be taken to the single dialogue annotation screen to annotate it. If the user uploaded a text file, they will be taken to a dialogue and turn segmentation screen. Following the same constraints imposed in previous works, this turn segmenter assumes that there are only two participants in the dialogue: the user and the system, and that the user asks the first query. The user separates each utterance in the dialogue by a blank line, and separates dialogues with a triple equals sign ("==="). Once the user clicks "Done", the text file will automatically be parsed into the correct JSON format and each query run through the recommenders in the back-end to obtain annotation suggestions.

3.0.4 Resolving Annotator Disagreement

Researchers could use LIDA's main interface to crowdsource annotations for a dialogue dataset. Once they have several annotations for each dialogue, they can upload these to the inter-annotator resolution interface of LIDA. The disagreements between annotators will be detected, with a percentage shown beside each label to show how many annotators selected it. The label with the highest percentage of selections is checked by default. The arbiter can accept the majority label simply by pressing "Enter" and can change errors with the arrow keys to facilitate fast resolution. This interface also displays an averaged (over turns) version of Cohen's Kappa, the total number of annotations, the total number of errors, and the averaged (over turns) accuracy.

3.1 Features

Specifying Custom Labels LIDA's configuration is controlled by a single script in the back end. This script defines which labels will be displayed in the UI and is easy to extend. Users can define their own labels by altering this configuration script. If a user wishes to add a new label, all they need to do is specify the label's name, its type (classification or slot-value pair, currently) and the possible values the classification can take. Alongside the label specification, they can also specify a recommender to use for the label values. The label will then automatically be displayed in the front end. Note that labels in uploaded datasets will only be displayed if the label has an entry in the configuration file.

Custom Recommenders When creating a dialogue dataset from scratch, LIDA is most powerful when used in conjunction with recommenders which can suggest annotations for user queries to be corrected by the annotator. State-of-the-art tools emphasize the importance of being able to use recommenders in annotation systems. Users can specify arbitrary ML models to use for each label in LIDA's back end. The back end is written in Python, the de facto language for machine learning, so researchers can directly integrate models written in Python to the back end. This is in contrast to tools such as INCEpTION and GATE which are written in Java and so require extra steps to integrate a Python-based model. To integrate a recommender, the user simply provides an instantiated Python object in the configuration file that has a method called "transform" that takes a single string and returns a predicted label.

Dialogue and Turn Segmentation from Raw Data When uploading a .txt file, users can segment each utterance and each dialogue with a simple interface. This means that raw dialogue data with no labels, such as obtained from a transcription service, can be uploaded and processed into a labelled dialogue. Segmented dialogues and turns are automatically run through every recommender to give suggested labels for each utterance.

4 Evaluation

To test LIDA's capabilities, we designed a simple experiment: we took a bespoke dataset of 154 dialogues with an average of 3.5 turns per dialogue and a standard deviation of 1.55. The task was to assign three classification labels to each user utterance in each dialogue. Each annotator was given a time limit of 1 hour and told to annotate as many dialogues as they could in that time. We had six annotators perform this task, three of whom were familiar with the system and three of whom had never seen it before.

These annotators annotated an average of 79 dialogues in one hour with a standard deviation of 30, which corresponds to an average of 816.5 individual annotations. The annotators who had never seen the system before annotated an average of 60 dialogues corresponding to an average of 617 individual annotations.

Once we had these six annotations, we performed a second experiment whereby a single arbiter resolved inter-annotator disagreements. In one hour, the arbiter resolved 350 disagreements and noted that resolution.