Comparison of Chatito & Fountain

Overview

Fountain and Chatito are both useful in generating dataset for training and validating chatbot models. Comparison should be made between both tools to see which will fit into the RASA models better.

Goal

PoC to compare Fountain and Chatito in data augmentation.

The comparison between Chatito and Fountain is conducted as per the following aspects:

* DSL (Domain Specific Language) Syntax
* Data Generation Probabilities
* Slots Recognition
* Overfitting Problem

Chatito

Chatito is a domain specific language designed to simplify the process of creating, extending and maintaining datasets for training natural language processing (NLP) models for text classification, named entity recognition, slot filling or equivalent tasks.

Chatito design principles:

* Simplicity: should be understandable by someone looking at it for the first time
* Speed: generate samples by pulling them from a cloud of probabilities on demand
* Practicality: this tool is meant to help people who use it, the design should be guided by the community needs

Fountain

Fountain is a natural language data augmentation tool that helps developers create and expand domain specific chatbot training datasets for machine learning algorithms.

Most of NLU system requires entering thousands of possible queries that future users would -- most possibly -- use, and annotate every sentence segment that can identify user's intentions. Fountain aims to help developers smooth away this manual hectic process and generate a volume of training examples to make it easier to train and build a robust chatbot systems.

DSL (Domain Specific Language) Syntax

The domain specific language (DSL) of Chatito and Fountain satisfies different goals. Chatito, as indicated by its design principles, aims at building generic, simple and practical DSL, while Fountain targets at building a more complex DSL to generate sufficient training data to manage large projects.

Chatito aims at a generic but powerful DSL, that should stay simple. While it is perfectly fine for small projects, when projects get larger, this simplicity may become a burden: your template file becomes overwhelmingly large, to the point you get lost inside it.

“The Chatito tool produces training examples by running through all the combinations specified in the file, producing hundreds and thousands of variations in each phrase. We limit the number of training examples that Chatito should generate for each intent to 5,000”.

Fountain defines a more complex DSL to be able to manage larger projects.

As an example, the Chatito data:

// This template defines different ways to ask for the location of toilets (Chatito version)

%[ask\_toilet]('training': '3')

~[sorry?] ~[tell me] where the @[toilet#singular] is ~[please?]?

~[sorry?] ~[tell me] where the @[toilet#plural] are ~[please?]?

~[sorry]

sorry

Sorry

excuse me

Excuse me

~[tell me]

~[can you?] tell me

~[can you?] show me

~[can you]

can you

could you

would you

~[please]

please

@[toilet#singular]

toilet

loo

@[toilet#plural]

toilets

As an example, the Fountain data:

// This template defines different ways to ask for the location of toilets (Fountain version)

%[&ask\_toilet](3)

~[sorry?] ~[tell me] where the {@[toilet#singular] is/@[toilet#plural] are} [please?]?

~[sorry]

sorry

excuse me

~[tell me]

~[can you?] {tell/show} me

~[can you]

{can/could/would} you

@[toilet#singular]

toilet

loo

@[toilet#plural]

toilets

The Chatito version is arguably easier to read, but the Fountain version is shorter, which may be very useful when dealing with lots of templates and potential repetition.

Data Generation Probabilities

In data generation process, Fountain mechanically takes all slots and make a full cartesian product of all the synonyms and slot sample values, while the way that Chatito works, is like pulling samples from a cloud of possible combinations and avoiding duplicates. Chatito outperforms Fountain in the way that it controls the data generation probabilities using semantics, thus, generating a more important (relevant) training data into the RASA model.

Once the sentences definitions gain complexity, the max possible combinations increments exponentially, causing a problem where the generator will most likely pick sentences that have more possible combinations, and omit some sentences that may be more important at the dataset. To overcome this problem, semantics for controlling the data generation probabilities are provided.

Two frequency distribution strategies for controlling probabilities in Chatito are specified in the link below: <https://github.com/rodrigopivi/Chatito/blob/master/spec.md>

When the generator generates samples for an entity, the training data that one wants to feed into the RASA model will influence which data generator, as well as, which frequency distribution strategies, to choose. If the training data incorporates a large volume of important short sentences, which are meaningful to feed into the RASA, Chatito is a better choice.

Slots Recognition

Chatito, with a number of predefined built-in slot types (entities), performs better at slots recognition than Fountain, which supports only several entities to simplify and standardize how data in the slots is recognized. Users can enrich the built-in datasets by adding more files.

Fountain Built-in Libraries

The library supports several predefined slot types (entities) to simplify and standardize how data in the slot is recognized.

These entities have been collected from different open-source data sources.

* Dates, and Times
  + FOUNTAIN:DATE
  + FOUNTAIN:WEEKDAYS
  + FOUNTAIN:MONTH\_DAYS
  + FOUNTAIN:MONTHS
  + FOUNTAIN:HOLIDAYS
  + FOUNTAIN:TIME
  + FOUNTAIN:NUMBER
* Location
  + FOUNTAIN:COUNTRY
  + FOUNTAIN:CITY
* People
  + FOUNTAIN:FAMOUSPEOPLE

Fountain Data Sources

In order to build Fountain's builtin data types, we processed data from the following data sources:

* [Wikidata](https://www.wikidata.org/wiki/Wikidata:Main_Page) WikiData
* [GeoNames](http://www.geonames.org/) Geographical database
* [Pantheon dataset](http://pantheon.media.mit.edu/rankings/people/US/all/-4000/2010/H15)

Overfitting Problem

As always with machine learning, having too much data may cause your models to perform less well because of overfitting.

To prevent overfitting, Chatito helps create an intersection between data augmentation and a probabilistic description of possible sentences combinations. It is not intended to generate deterministic datasets and avoids generating all possible combinations. Fountain does not have this functionality, thus, user should be aware of the accuracy of the model to detect overfitting and compare against previous accuracies.

By default, intent definitions can expect the training and testing argument keys, when defined, are used to declare the maximum number of unique examples to generate for the given intent, and splitting them in two datasets, the training dataset is to be used to train the NLU model, and the testing dataset should be used to evaluate the accuracy of the model with examples it never trained with.

In Chatito, users can specify the number of training and testing examples to per intent in order to prevent overfitting during training. The generator will first populate the training dataset and testing dataset until reaching the sum of both values.

%[greet]('training': '2', 'testing': '1')

hello

hi

hola

salute

The greet intent could generate a maximum of 4 examples, but the declaration only requests 3. The training dataset will contain 2 utterances for greet intent and the testing dataset 1.

Summary

This documentation specifies the comparison of two powerful data augmentation tools, Chatito and Fountain, for RASA model, in four aspects, DSL (Domain Specific Language) Syntax, data generation probabilities, slots recognition and overfitting problem. Chatito, a generic and simple data augmentation tools, lost its competitiveness to Fountain in its DSL language syntax, which becomes a burden in large projects. However, Chatito has an edge over Fountain in generating more important and balanced training data, recognizing slots entities as well as avoiding overfitting problem in data augmentation stage. Overall, Chatito seems to be a more powerful data augmentation tool for RASA model at this stage.

Reference

Chatito Projects Github

<https://github.com/rodrigopivi/Chatito>

Fountain Projects Github

<https://github.com/tzano/fountain>

Two Controlling Data Generation Probabilities Method in Chatito

<https://github.com/rodrigopivi/Chatito/blob/master/spec.md>

Data Generator Scripts for Fountain

<https://github.com/tzano/fountain/blob/master/fountain/data_generator.py>

Fountain Predefined Built-in Libraries for Slots Recognition

<https://github.com/tzano/fountain/blob/master/fountain/resources/builtin.py>

How Chatito works in Building Chatbots

[AI Blueprints: How to build and deploy AI business projects](https://books.google.com.pe/books?id=sR2CDwAAQBAJ) implements practical full chatbot examples using chatito at chapter 7.