Computer Engineering Department
Faculty of Engineering
Cairo University

CMP 4019
Fall 2024

Natural Language Processing
Project Document

Project Description

With the growth of online food ordering, accurately understanding unique customer requests - such as a pizza with extra toppings or a specific type of crust - has become a complex challenge. Customers often use varied language to express these detailed preferences, so systems must go beyond basic responses to interpret each request accurately. This project addresses these needs by designing an NLP system to recognize the many nuances in pizza orders, transforming customer instructions into clear, structured information that can be reliably used.

The goal of this project is to develop sequence models that can accurately identify key items in complex food ordering requests, focusing specifically on pizza customization. By organizing natural language into actionable data, this project tackles real-world NLP challenges where specialized entities (such as "toppings," "pizza sizes," and "crust types") require precise extraction. This enables a more robust fulfillment of orders in online food applications.

Dataset Description

The dataset comprises a set of annotated commands related to pizza orders, with each entry labeled for entities such as ingredients, pizza types, sizes, and customization options. The dataset is split into training, development, and test sets, allowing us to build, tune, and evaluate the model's performance in a controlled environment.
The dataset portion sizes are as follows:

1. The training set contains 2,456,446 samples.
2. The dev set contains 348 samples.

Below, we give a high level idea of the main rules our target semantics follow:

- Each order can include any number of pizza and/or drink suborders. These suborders are labeled with the constructors **`PIZZAORDER`** and **`DRINKORDER`**, respectively.
- Each top-level order is always labeled with the root constructor **`ORDER`**.
- Both pizza and drink orders can have **`NUMBER`** and **`SIZE`** attributes.

- A pizza order can have any number of **TOPPING** attributes, each of which can be negated. Negative particles can have larger scope with the use of the **or** particle, e.g., **no peppers or onions** will **negate** both peppers and onions.
- Toppings can be modified by quantifiers such as **a lot** or **extra**, **a little**, etc.
- A pizza order can have a **STYLE** attribute (e.g., **thin crust** style or **chicago** style).
- Styles can be negated.
- Each drink order must have a **DRINKTYPE** (e.g. **coke**), and can also have a **CONTAINERTYPE** (e.g. **bottle**) and/or a **volume** modifier (e.g., **three 20 fl ounce coke cans**).

ORDER, PIZZAORDER, and **DRINKORDER** are viewed as intents, and the rest of the semantic constructors as composite slots, with the exception of the leaf constructors, which are viewed as entities (resolved slot values).

Input Example:

```
# Input
'''i want to order two medium pizzas with sausage and black olives and
two medium pizzas with pepperoni and extra
cheese and three
large pizzas with pepperoni and sausage '''
```

Output Example:

```
# Ouput
'''
(ORDER i want to order (PIZZAORDER (NUMBER two ) (SIZE medium )
pizzas with (TOPPING sausage ) and (TOPPING black olives ) )
and (PIZZAORDER (NUMBER two ) (SIZE medium ) pizzas with
(TOPPING pepperoni ) and (COMPLEX_TOPPING (QUANTITY extra ) (TOPPING cheese ) ) )
and (PIZZAORDER (NUMBER three ) (SIZE large )  pizzas with (TOPPING pepperoni ) and (TOPPING sausage ) ) ) '''
```

From the above example, each component of the order is a "**node**" within nested parentheses that represent relationships and categories. The root node, **(ORDER ...)**, serves as the main container for the full order. Within this, each sub-node like **(PIZZAORDER ...)** breaks down the details for individual pizzas, grouping related elements such as quantity **(NUMBER ...)**, size **(SIZE ...)**, and toppings **(TOPPING ...)**. Nested nodes, like **(COMPLEX_TOPPING ...)**, further specify complex attributes, such as adding extra or specific quantities to toppings, creating a structured output.

Project Pipeline

Your task is to build a system that takes only the order text and produces both the entities and the parsing tree of the order. Here is the pipeline that you will follow:

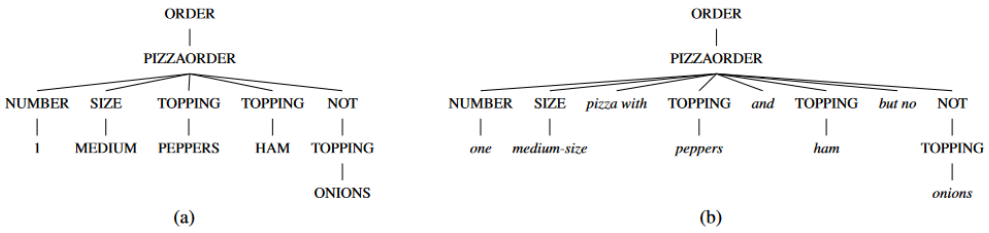
1. **Data Preprocessing:**

   Data cleaning: eliminate undesired words or symbols from the input strings, such as special characters (e.g., @, #), unnecessary punctuation, and irrelevant terms.

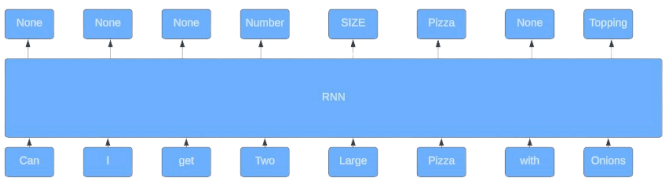
   Tokenization: Use the tokenizer that you think is more suitable for the data.

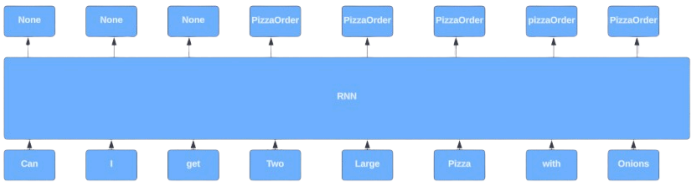
   Lemmatization: You can try if using lemmatizers will be useful for your case or not.

2. **Feature Extraction:** you are required to try **at least three different** features (e.g. Bag of Words, TF-IDF, One hot encoding,Word Embeddings, Trainable embeddings etc.) It would also be great if you tried other features (e.g. contextual embeddings).
3. **Model Building:** In this phase, you are required to build **at least two different** machine learning models that **at least one** of the models has to be a deep learning model(e.g. RNN, LSTM). Optimizing the model weights will be done using the training set. You will need to use the dev set to pick the model that performs the best. You can have multiple models in your system based on your approach.For example, you could have one model focused on identifying **entities** at the word level and another model dedicated to understanding relationships between entities and creating boundaries to build a structured representation, such as a **parsing tree** as it is shown in the following example :



The following diagrams show the proposed solution that you may follow :





4. **Model Testing:** In this phase, you will use your best-performing model to produce the entities and parsing tree of the given test set orders.

Grading Criteria

This is a competitive project. The teams will be ranked based on the scores they get in the two tasks: entity extraction and constructing the parsing tree. You will be provided with the test set orders only **ONE DAY** before the final delivery. We will use **Exact Match Accuracy (EM) with Modulo Sibling Order** as the metric for the ranking process. We will use Kaggle for the ranking process.

The overall grading will depend on the following:

1. The team rank in the two tasks
2. The approach you followed. This includes the following:
    a. The preprocessing techniques
    b. The features you used (at least three different features)
    c. The models you trained (at least two different models).
3. The workload division.

Project Schedule

- **Project Document Release**: Week 6 (Saturday 2nd Nov. 2024)
- **Final Delivery**: Week 12.

Project Instructions

- You will work in teams of 4.
- Your final submission only will be considered for the ranking process.
- There is a penalty for late submissions.
- Any sign of cheating or plagiarism will not be tolerated and will be graded **ZERO** in the project.

Final Deliverables

1. **Final Project Document** containing the following:
    a. Project Pipeline
    b. A detailed description of each phase in your pipeline
        i. Data preprocessing
        ii. Feature extraction
        iii. Model training
    c. Evaluation: Report the accuracy (and all other metrics you tried) for all trials you did.
    d. Specify what model you used for the test set submission on Kaggle and the reason for choosing it.
2. **Codes**: All scripts you used.

3. **The final Model**: the weights of the model you used for submission. Use the framework you used for training the model default format when saving.
4. **Presentation**: you will use it for the final project discussion.
5. **Demo**: Simple demo where we can enter pizza orders and your system generates a structured output after parsing (e.g. in JSON format)