The task of open domain semantic parsing is to construct a **logical form** (e.g., lambda-calculus) for a natural language question. For simplicity, we will focus on two types of questions, **single-relation questions** and questions with **CVT** structures. Each question in the training set and development set is annotated with entities, the question type and the corresponding logical form. And each question in the test set is annotated with entities and the question type, except for **the logical form which should be predicted by your model**. (We will release the test dataset later.)

In the training dataset, there are 30804 questions in total, 26955 single-relation questions and 3849 cvt type questions. You could treat the two types of questions separately and your credits are mainly related to the performance of your model on single-relation questions.

Here, we give an example for each type of quesitons:

Single-relation:

<question id=2>what is famous novel joan blondell
<logical form id=2> (lambda ?x (mso:book.author.works_written joan_blondell ?x))
<parameters id=2> joan_blondell (entity) [4,5]
<question type id=2> single-relation

Since we have give you the entities of the question, you just need predict the relation *mso:book.author.works_written* for this logical form. (The pattern of lambda-calculus for each question type is the same.)

cvt:

<question id=23> what is marriage date for danielle steel
<logical form id=23> (lambda ?x exist ?y (and (mso:people.person.marriage danielle_steel ?y) (mso:people.marriage.type ?y marriage) (mso:time.event.start_date ?y ?x)))
<parameters id=23> marriage (entity) [2,2] ||| danielle_steel (entity) [5,6]
<question type id=23> cvt

For this question type, the pattern of lambda calculus is also constant, and you just need to fill the corresponding entity slots and relation slots. Specifically, the relation slots are usually appeared in a fixed combination, such as *mso:people.person.marriage*, *mso:people.marriage.type* and *mso:time.event.start_date* could be seen as a fixed combination. This observation maybe helps you build your model for cvt type questions.

In addition, we provide a simple python evaluation script, which gives the Exactly Matching (EM) score of your preditions. Note that, to use this script, you should make your predictions has the same order with gold data. The **EM score** is our mainly metrics, and you could also provide some other metrics on the dev dataset in your final report.

To use python script, you just need the following command:

```
python Evaluate_script.py PREDICTED_FILENAME GOLD_FILENAME
```

If you have any questions about this project, please contact me. (1801213762@pku.edu.cn)