COMP5423 NATURAL LANGUAGE PROCESSING

Group Project Instruction (Draft[1])

Tuesday, March 9, 2021

Topic: Document-based Question Answering

Group Size: 4-5 Students per Group

Due date: April 20, 2021 (Tuesday, 23:59)

Objectives:

Over past decades, there has been a growing interest in making the machine understand human languages. And recently, great progress has been made in machine reading comprehension (MRC). In one view, the recent tasks titled MRC can also be seen as the extended tasks of question answering (QA). The fast development of the MRC field is driven by various large and realistic datasets released in recent years. Each dataset is usually composed of documents and questions for testing the document understanding ability. The answers for the raised questions can be obtained through seeking from the given documents and even from the external knowledge bases.

Extractive QA vs Abstractive QA vs Multiple-Choice QA:

According to the formats of answers, MRC datasets can be classified into three types, namely datasets with extractive answers, with abstractive answers, and with multiple-choice answers. In this project, we focus on the last one. Multiple-choice question has long been used for testing students reading comprehension ability, and can be objectively gradable. Generally, this kind of questions can extensively examine one's reasoning skills, including simple pattern recognition, clausal inference and multiple sentence reasoning, of a given passage.

Conventional Features vs Vector Representations vs Deep Neural Networks:

According to implementation approaches, QA systems can also be classified into conventional feature based, vector representation based, and deep neural network based. (1) The simplest way is to extract indicative features from both documents and questions based on syntax, frame semantics, co-reference. These conventional features are then combined into a classifier to produce the final decision. (2) Another way is to select the option answers based on vector representations, e.g., pre-trained word embeddings. (3) With the development of deep neural networks, the attention-based architecture becomes a widely adopted learning paradigm. This paradigm allows an end-to-end way to learn representations and then select the answers based on the learned representations and attention scores.

---

[1] The project description may have minor changes according to the feedback from you and our teaching progress. Please keep close contact with TA.

MCTest, RACE and DREAM Datasets:

We provide three multiple-choice document-based question answering dataset to evaluate your approaches, i.e., MCTest, RACE, and DREAM. The datasets are available at /Content/Group Project/Project Description/dataset.zip. You can also download these datasets from the following links:

- MCTest: https://mattr1.github.io/mctest/data.html
- RACE: https://www.cs.cmu.edu/~glai1/data/race/
- DREAM: https://dataset.org/dream/

Basic statistics and required QA skills are presented in the table below.

	#Docs	#Questions	#Words/Doc	#Words/Q	Required Skill
MCTest	500	2,000	205	8.0	Matching
RACE	27,933	97,687	323	10.0	Matching Multi-Sentence Reasoning
DREAM	6,444	10,197	86	8.6	Matching Multi-Sentence Reasoning Commonsense Reasoning

According to the table, we can see:

1. MCTest is the smallest benchmark, while DREAM is the second large dataset, and RACE is the largest. It will require a relatively long time to run your system on RACE and DREAM datasets.
2. Besides the dataset size, these three datasets require different skills for QA systems to answer correctly. MCTest is the simplest dataset where the majority of the questions can be handled using word matching techniques. RACE is more difficult since answering 60% of its questions require summarizing multiple evidences across the documents, the so-called multi-hop reasoning skill. Besides the textual information in the dialogue, 30% questions in DREAM require additional commonsense knowledge that cannot be obtained from the dialogue.
3. Hence, DREAM is the most difficult one.

Remarks:

1. You are required to focus on developing document-based question answering systems.
2. You can implement the document-based question answering system based on auxiliary approaches.
3. You are required to evaluate your system on at least one of the three datasets,
4. i.e., MCTest, RACE, DREAM.
5. You are strongly recommended to write the program in **Python**.

Project Requirements:

You are required to develop a multiple-choice document-based question answering system to select the answer from several candidates.

Input: a document, and a question (query)
Output: an answer (select from options)

Basic Requirements (50-70)

1. Your system should be able to test on MCTest.
2. Your system supports basic word matching techniques using at least conventional features.
3. Your system selects an answer based on well-encapsulated feature extractors and classifiers provided by existing platforms (e.g., nltk, gensim, etc).

Advanced Requirements (70-90)

1. Your system at least meets the basic requirements.
2. Your system should be able to test on both MCTest and RACE/DREAM datasets.
3. Your system should be equipped with reasoning skills more than simple word matching.
4. Your system selects an answer based on vector representations, e.g., pretrained embeddings, pretraining+finetuning approaches, etc.

Superior Requirements (90+)

1. Your system at least meets the basic and advanced requirements.
2. You will receive more marks if your system is tested on both RACE and DREAM benchmarks.
3. Your system is designed to handle multiple-hop reasoning skill and even commonsense reasoning.
4. Your system selects an answer using deep neural network based approaches, e.g., attention mechanisms, etc.

References

Required Datasets:

MCTest: A Challenge Dataset for the Open-Domain Machine Comprehension of Text
RACE: Large-scale ReAding Comprehension Dataset From Examinations
DREAM: A Challenge Dataset and Models for Dialogue-Based Reading Comprehension

Existing Approaches (Codes and Pre-trained Models):

Linguistic Features for Document-based QA
Using Pre-trained Word Embeddings
Using BERT-based Embeddings for Document-based QA
RACE Baselines
BERT for RACE
Baidu Machine Reading Comprehension Platform PALM

Related Work (Reference Papers):

Machine Reading Comprehension: a Literature Review
A Survey on Neural Machine Reading Comprehension
A Survey on Machine Reading Comprehension Systems

Machine comprehension with syntax, frames, and semantics

Free GPU Resources:

Baidu PaddlePaddle: Course Registration Tutorial Usage
Google Colab: Tutorial Usage
Kaggle Kernel: Tutorial Usage
FolydHub: Tutorial Usage

What to Hand In:

1. The Source Code of Your Program.
   - The reported performance can be achieved using the code step by step.
   - A readme file describes: the structure of your program, the steps to run the code to achieve the reported performance and the dependence environment (the used packages and their versions).
   - Add annotations in the front of each Python file to specify its usage and necessary comments.
   - Put all the files into a directory named "code".
   - TAs will run the code. If there are some problems, TAs will contact your group to demonstrate online. And if the submitted code cannot achieve the reported performance finally, there will be some deductions of your grade.
2. Written Report (At Least 6 Pages with 12-Point Font Size and Single Line Spacing, Not Including the Cover Page. In the form of pdf.)
   - A cover page with project topic and group member' names and IDs;
   - Introduction of the topic (e.g., the significance and practical value of the topic, etc.);
   - Functions of the (expected) system (including those you have implemented and those you can only design);
   - Flowchart diagram of the system;
   - Approaches and tools used to implement the functions;
   - The accuracy of your system on the test set of the dataset. And some results analysis is necessary. (The reported accuracy must be achieved using the submitted code.)
   - The role and the contribution of each group member;
   - Anything else you would like to share with us.

Pack all the submissions in one zipped file and submit to Blackboard. As described above, the file contains the "code" directory and the report pdf.

Remark: Please do remember to click the "Submit" button after you upload the file

Grading Scheme:

We will grade the project according to the following schemes:

1. Written Report 20%
2. System Implementation 80%

Note that we will assess your system implementation based on the written report and the code. Please refer to the COMP5423 Group Project Grading Criteria file in /Content/Group Project/Project Description/ for more details. In general, the members in the same group will received the same marks unless there are strong arguments.

Contact Information:

- Yongqi Li (csyqli@comp.polyu.edu.hk)
- Jiashuo Wang (csjwang@comp.polyu.edu.hk
- Maggie (cswjli@comp.polyu.edu.hk)