Applied Deep Learning Homework 1

Kaggle Due: 2022/03/20 23:59
Code/Report Due: 2022/03/22 23:59

UPDATES

Fed 23 10:15 PM - 開放 / 更新套件版本

Mar 1 10:49 PM - 開放 NLTK

Links

Homework Page

COOL Discussion

Sample Code

Data

Kaggle Intent Classification

Kaggle Sequence Tagging

adl-ta@csie.ntu.edu.tw

Submission FAQ

- 模型不見了怎麼辦？
  - 如果無法 reproduce 最終表現最好的模型，可以上傳其他的模型，如果有過 baseline會給分，但是『可能斟酌扣一點點分數』。
- Reproduce有一點誤差可以嗎？
  - Reproduce 成功 = Reproduced 分數 >= Kaggle 分數 * 0.99

Outline

Task Description

Part 1: Intent Classification

- Input: Text
  "i dont like my current insurance plan and want a new one",
  "when will my american express credit card expire",
  "how would i get to city hall via bus",

- Output: Intent
  "insurance_change",
  "expiration_date",
  "directions"

Part 2: Slot Tagging

- Input: Text
  "A table today for myself and 3 others"
  "My three children and i are in the party"

- Output: Intent
  "O O B-date O B-people I-people I-people O"
  "B-people I-people I-people I-people I-people O O O O"

Slot Tagging

- Slot Tagging: Inside-Outside-Beginning tagging problem
- Similar to NER task
- Classify each token in a sentence to a {O, B-xxx, I-xxx}
- After Preprocessing, this problem can be reduced to a multi-class classification problem

Metrics

- Intent Classification
    - Accuracy
- Slot Tagging
    - Joint Accuracy
    - A sample is correct only if all tokens are predicted correctly.

What to do

- Train an intent classification model and pass baselines:
  - Public Baseline: **0.87377**
  - Private Baseline: **Released after deadline**
- Train a slot tagging model and pass baselines:
  - Public Baseline: **0.71045**
  - Private Baseline: **Released after deadline**

Data

- Labeled data
  - train.json
  - eval.json
- Unlabeled data
  - test.json
- Download link

Data Format (json)

- Intent Classification
  - `id: str`
  - `text: str`
  - `Intent: str` ⇨ *Only in* `train.json` *and* `eval.json`
- `Slot Tagging`
  - `id: str`
  - `text: List[str]`
  - `tags: List[str]` ⇨ *Only in* `train.json` *and* `eval.json`

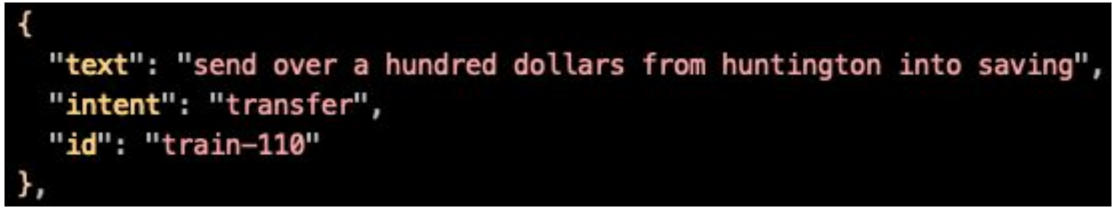
Field Description

- `Intent Classification`
  - `id:` Unique id
  - `text:` Input sentence
  - `intent:` A string that denotes the intent of the input sentence
- Slot Tagging
  - `id:` Unique id
  - `text:` A list of input tokens preprocessed from the input sentence
  - `tags:` A list of strings, each denotes the tag of its corresponding token in the input sentence

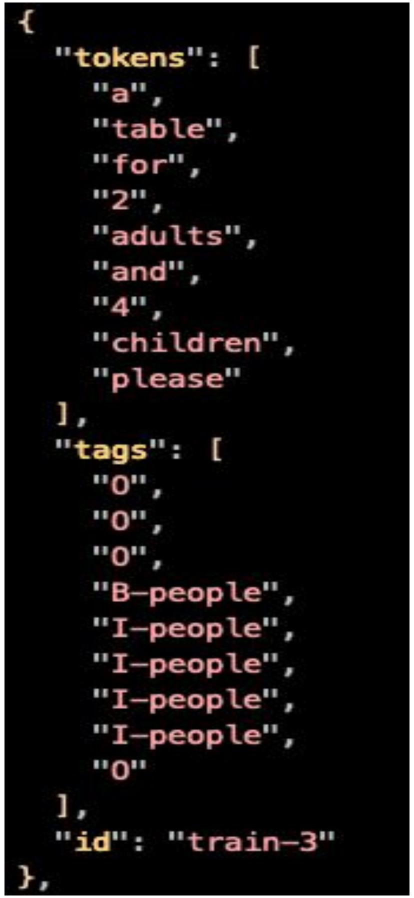
Data example in `train.json`

Slot Tagging

Intent Classification



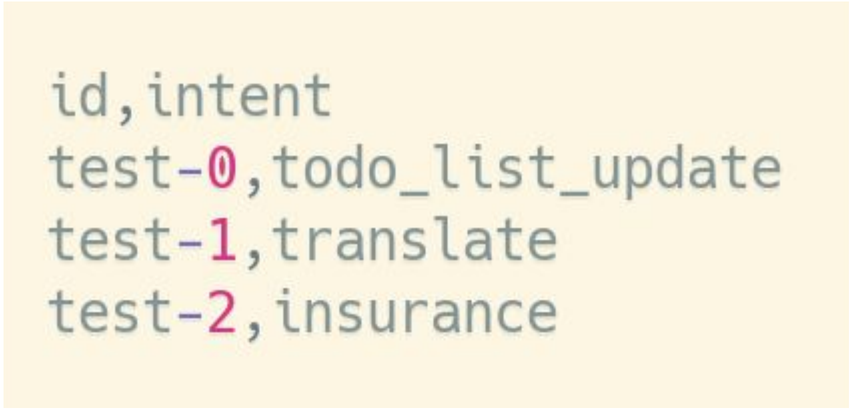
```
{
  "text": "send over a hundred dollars from huntington into saving",
  "intent": "transfer",
  "id": "train-110"
},
```



```
{
  "tokens": [
    "a",
    "table",
    "for",
    "2",
    "adults",
    "and",
    "4",
    "children",
    "please"
  ],
  "tags": [
    "O",
    "O",
    "O",
    "B-people",
    "I-people",
    "I-people",
    "I-people",
    "I-people",
    "O"
  ],
  "id": "train-3"
},
```

Submission Format - Intent Classification

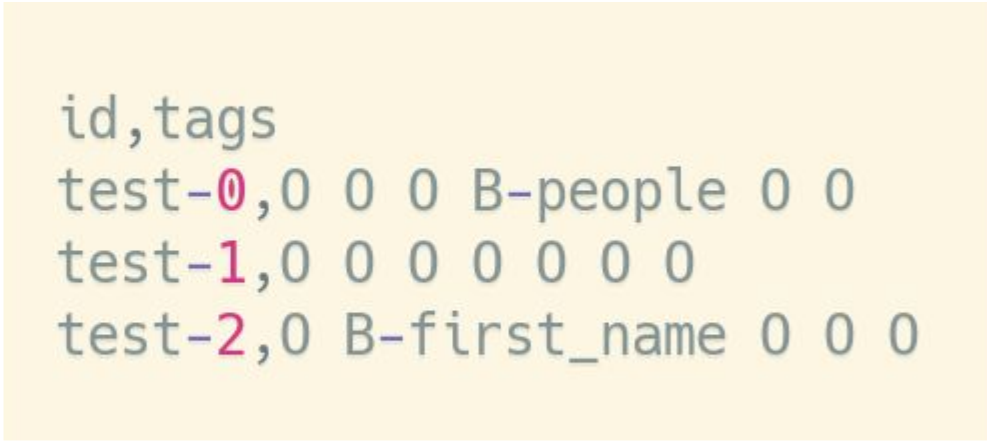
- CSV (Comma Separated Values) format with 2 columns:
  - id: Unique id for each sample
  - intent: Your prediction.



```
id,intent
test-0,todo_list_update
test-1,translate
test-2,insurance
```

Submission Format - Slot Tagging

- CSV (Comma Separated Values) format with 2 columns:
  - id: Unique id for each sample
  - tags: Your prediction. The tags should be separated with single space.



```
id,tags
test-0,O O O B-people O O
test-1,O O O O O O O
test-2,O B-first_name O O O
```

Logistics

Grading

- Model Performance (10%)
  - Your intent classification model passes the baseline on the public test set (2%) and the private test set (3%) on kaggle
  - Your slot tagging model passes the baseline on the baseline on the public test set (2%) and the private test set (3%) on kaggle
  - Only if you can reproduce your best submission in `intent_cls.sh` `slot_tag.sh`
- Format (1%)
  - TA can run the grading script without human intervention.
- Report (9% + 1% Bonus)
  - In PDF format!

Code/Scripts/Report Submission

- Zip your folder into a single **.zip** file.
- Submit to NTU Cool.

File Layout

Your zip must contain files (case sensitive):

- `/[student id (lower-cased)]/`, ex. `/r12922000/`, no brackets
    - `intent_cls.sh`
    - `slot_tag.sh`
    - `README.md`
    - `report.pdf`
    - `download.sh`
    - Any other code/script.
- Do not upload training, validation, testing data and model to COOL.

Submission Files - download.sh

- `download.sh` to download your model.
    - Do not modify your file after deadline, or it will be seen as cheating.
    - Keep the URLs in `download.sh` valid for at least 2 weeks after deadline.
    - Do not do things more than downloading. Otherwise, your `download.sh` may be killed.
    - You can download at most 4G, and `download.sh` should finish within 1 hour.
- You can upload your model to Dropbox. (see tutorial)
- We will execute download.sh before predicting scripts.

Submission Files - Scripts

- `intent_cls.sh`, `slot_tag.sh`
- Corresponding to the intent classification model, slot tagging model
  - `"${1}"`: path to the testing file.
  - `"${2}"`: path to the output predictions.
- TA will predict testing data as follow:
  - `bash ./intent_cls.sh /path/to/test.json /path/to/pred.csv`
- Default python version would be 3.8
- **Make sure your code works!**

Submission Files - Reproducibility

- All the code you used to train, predict, plot figures for the report should should be upload.
- README.md
  - Write down how to train your model with your code/script **specifically**.
  - If necessary, you will be required to reproduce your results based on the `README.md`.
  - If you cannot reproduce your result, you may lose points.
- You will get at least - 2 penalty if you have no or empty README.md.

Execution Environment

- Will be run on computer with
  - Ubuntu 20.04
  - 32 GB RAM, GTX 3070 8G VRAM, 10G disk space available.
  - the packages we allow only.
  - python 3.8
- Time limit **<u>60</u>** min for `intent_cls.sh slot_tag.sh` in total
- No network access when predicting.
- You will lose (some or all) your model performance score if
  - your script is at wrong location, or cause any error.

Rules

Kaggle

- Displayed Team Name: [student_id]
  - e.g. r12345678
- For auditing, Displayed Team Name: audit_[anything]
  - E.g. audit_4fun
- You can submit your result 5 times a day for each task.
  - Any approaches to submit more than 5 times a day is prohibited!

What You Can Do

- Train with the data we give you.
- Use publicly available pre-trained word embeddings. (No contextualized word embedding.)
- Use the packages/tools we allow:
  - Python 3.8 and Python Standard Library
  - PyTorch 1.10.2, TensorFlow 2.8.0, pytorch-lightning 1.5.10
  - Gensim 4.1.2 for pre-processing functions only.
  - SpaCy 3.2.2 and NLTK 3.7 for non-model-based functions.
  - seqeval=1.2.2, tqdm, numpy, pandas, scikit-learn 1.0.2
  - Dependencies of above packages/tools.
- If you want to use other package, COOL/mail TA.

What You Can **NOT** Do

- Any means of cheating or plagiarism, including but not limited to:
  - Use others' code from anywhere (e.g. web, github, classmate, etc.).
  - Use the labels of the test data directly or indirectly. (Do not try to find them.)
  - Use package or tools not allowed.
  - Use model trained with other data.
  - Give/get model prediction to/from others.
  - Give/get trained model to/from others.
  - Publish your code before deadline.
- Violation may cause zero/negative score and punishment from school.

Submission Policy

- Submit to NTU Cool.
- Late submission of "code and report":
    - 0 day < late submission ≤ 1 day: original score * 0.95
    - 1 day < late submission ≤ 3 day: original score * 0.90
    - 3 day < late submission ≤ 4 day: original score * 0.75
    - 4 day < late submission ≤ 5 day: original score * 0.50
    - 5 day < late submission ≤ 6 day: original score * 0.25
    - 6 day < late submission: original score * 0.00
- Late submission is determined by the last submission.
    - Update your submission after deadline
      implies that you will get penalty.

Report

You may lose score if TA has difficulty understanding it.

Please write in a human-readable way.

When Describing Model

- Please limit the use of imprecise words.
- Use equation whenever possible.
- Descriptions which is imprecise or hard to understand may cause loss of points.
- Ex.
  - bad: Feed the embedding of the sentence into a LSTM.
  - good: h_t, c_t = LSTM(w_t, h_{t-1}, c_{t-1}), where w_t is the word embedding of the t-th token.

Q1: Data processing (2%)

1. Describe how do you use the data for `intent_cls.sh`, `slot_tag.sh`:
    a. How do you tokenize the data.
    b. The pre-trained embedding you used.
2. If you use the sample code, you will need to explain what it does in your own ways to answer Q1.

Q2: Describe your intent classification model. (2%)

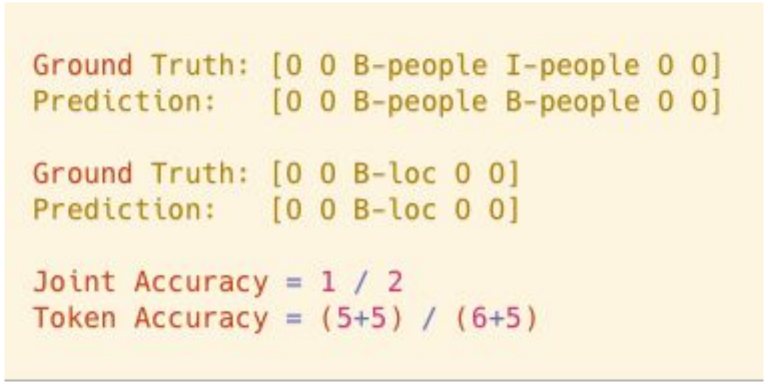
1. Describe
   a. your model
   b. performance of your model.
      (public score on kaggle)
   c. the loss function you used.
   d. The optimization algorithm (e.g. Adam), learning rate and batch size.

Q3: Describe your slot tagging model. (2%)

1. Describe
   a. your model
   b. performance of your model.
      (public score on kaggle)
   c. the loss function you used.
   d. The optimization algorithm (e.g. Adam), learning rate and batch size.

Q4: Sequence Tagging Evaluation (2%)

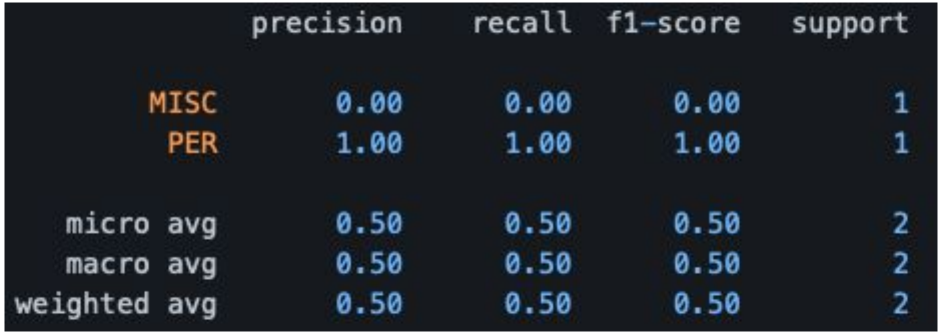
- Please use seqeval to evaluate your model in Q3 on validation set and report *classification_report(scheme=IOB2, mode='strict')*.
- Explain the differences between the evaluation method in seqeval, token accuracy, and joint accuracy.



```
Ground Truth: [O O B-people I-people O O]
Prediction:   [O O B-people B-people O O]

Ground Truth: [O O B-loc O O]
Prediction:   [O O B-loc O O]

Joint Accuracy = 1 / 2
Token Accuracy = (5+5) / (6+5)
```



```
              precision    recall  f1-score   support

        MISC       0.00      0.00      0.00         1
         PER       1.00      1.00      1.00         1

   micro avg       0.50      0.50      0.50         2
   macro avg       0.50      0.50      0.50         2
weighted avg       0.50      0.50      0.50         2
```

Q5: Compare with different configurations (1% + Bonus 1%)

- Please try to improve your baseline method (in Q2 or Q3) with different configuration (includes but not limited to different number of layers, hidden dimension, GRU/LSTM/RNN) and EXPLAIN how does this affects your performance / speed of convergence / ...
- Some possible BONUS tricks that you can try: multi-tasking, few-shot learning, zero-shot learning, CRF, CNN-BiLSTM
- This question will be grade by the completeness of your experiments and your findings.

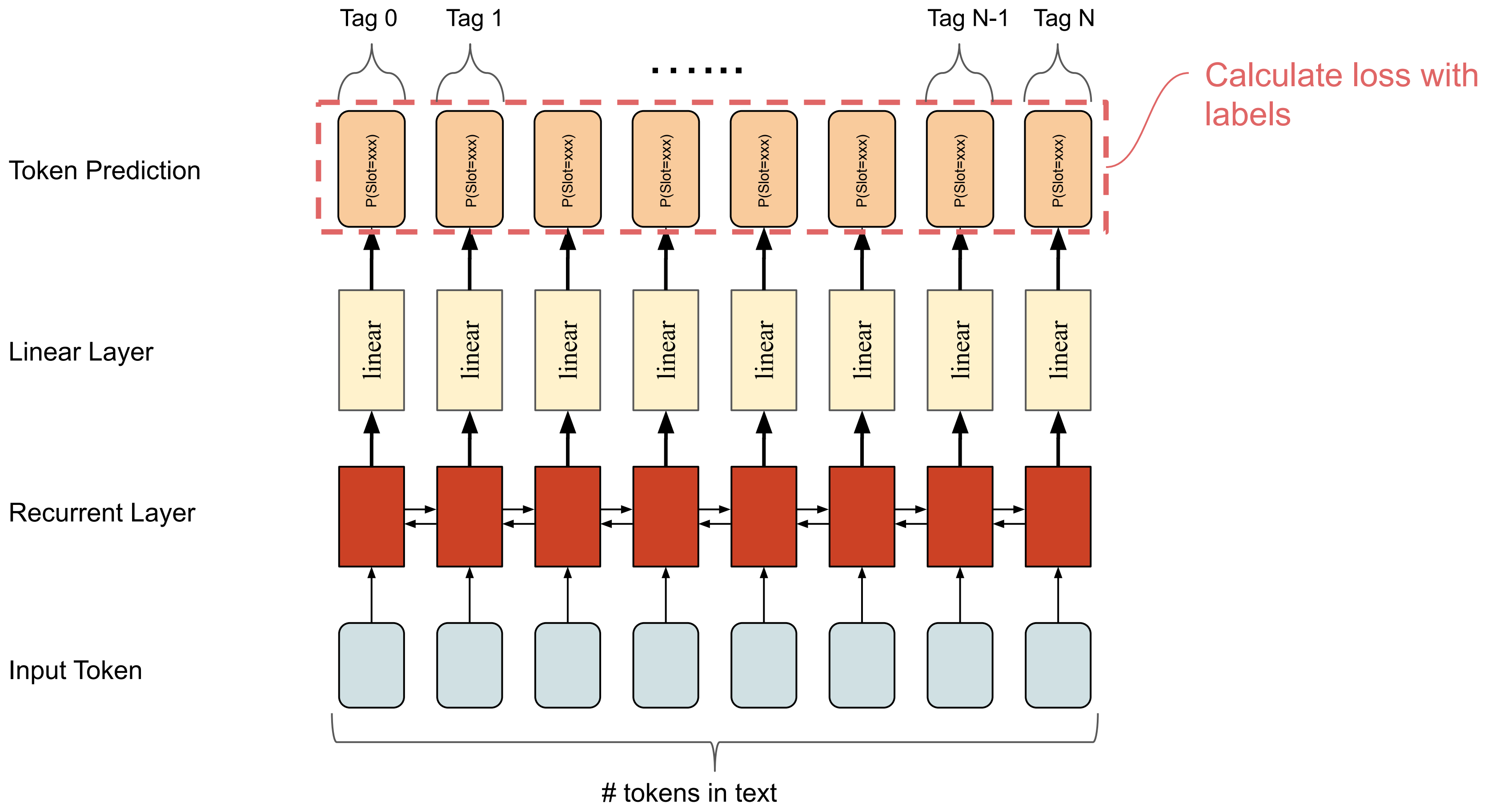
Guides

Pipeline for (Deep) NLP

- Load pre-trained embedding (GloVe, ...).
- Preprocess the dataset
  - Tokenize the sentences (SpaCy).
  - Convert token to word indices.
- Prepare batch
  - Sample batch
  - Pad samples to the same length.
- Train, check metrics on validation.
- Predict!

Already written in the sample code

Slot Tagging Pipeline



Sample Code

- Link
- TA will not explain the sample code for you.
- You can also write from scratch.
- Any bug report is welcome!
- See READMD.md for instructions