CSE 556: Natural Langugae Processing Assignment 2

Deadline - 16 March 2025, Sunday (11:59 pm IST)

Instructions

1. **Plagiarism** will be very strictly dealt with, and institute policy will apply.
2. The assignment is time consuming so please start timely. Each member is expected to contribute to a task individually and should be aware of all other tasks.
3. You have to submit a single .zip file named **A2-{Group No.}.zip**. Example: **A2-4.zip**.
4. **Viva:** During the demo, some questions related to the concepts covered in this task and your implementation will be asked. The marks for the viva will not be based on group they will be assigned individually.
5. **End Note:** Training an NLP model is like teaching a parrot to summarize Shakespeare—it takes patience, data, and a lot of trial and error. But fear not! With enough epochs and the right embeddings, even the noisiest model can learn to speak sense.

1 Task - Aspect Term Extraction — 35 Marks

1.1 Task Description

The goal of this assignment is to perform **Aspect Term Extraction (ATE)**. Given a dataset containing sentences annotated with aspect terms and their corresponding sentiment polarities, students will preprocess the data using BIO encoding and train deep learning models to extract aspect terms.

1.2 Dataset Description

You will be provided with `train.json` and `val.json` files containing examples in the following format:

```
{
    'sentence_id': '3534',
    'sentence': 'All the money went into the interior decoration, none of it
        went to the chefs.',
    'aspect_terms': [
        {
            'term': 'interior decoration',
            'polarity': 'positive',
            'from': '28',
            'to': '47'
        },
        {
            'term': 'chefs',
            'polarity': 'negative',
            'from': '72',
            'to': '77'
        }
    ],
    'aspect_categories': [
        {
            'category': 'ambience',
            'polarity': 'positive'
        },
        {
            'category': 'food',
            'polarity': 'negative'
        }
    ]
}
```

1.3 Preprocessing

You need to preprocess the data into the following format using BIO encoding:

```
{
    'sentence': 'All the money went into the interior decoration, none of it
        went to the chefs.',
    'tokens': [
        'All', 'the', 'money', 'went', 'into', 'the', 'interior', 'decoration
            ,', 'none', 'of', 'it', 'went', 'to', 'the', 'chefs.'
    ],
    'labels': [
        'O', 'O', 'O', 'O', 'O', 'O', 'B', 'I', 'O', 'O', 'O', 'O', 'O', 'O',
            'B'
    ],
    'aspect_terms': [
        'interior decoration',
        'chefs'
    ]
}
```

The preprocessed files should be saved as `train_task_1.json` and `val_task_1.json`.

1.4 Model Training

You are required to train the following models:

- RNN and GRU with pre-trained GloVe and fastText embeddings. (In total 4 models).

The training and validation loss plots for all models must be included in the report.

1.5 Evaluation Criteria

The models will be evaluated based on the **F1-score**. If padding is applied during training, it must be ignored while computing the F1-score. Failure to do so will result in a score of zero. You must **save your best performing model.** You need to use the following function for F1-calculation https://github.com/sighsmile/conlleval. Report both "chunk-level" and "tag-level" performance.

1.6 Testing and Model Inference

A `test.json` file will be provided during the demo. You must implement a function that:

- Loads the trained model.
- Processes the `test.json` file.
- Returns the stated f1-scores using conlleval evaluate function.

1.7 Deliverables

You must submit the following:

- Preprocessed datasets: `train_task_1.json` and `val_task_1.json`.
- Implementation code for preprocessing, model training, and inference in a single file named `task1.py/task1.ipynb`.
- Best performing saved model.
- A detailed report including:
  - Explanation of preprocessing steps.
  - Model architectures and hyperparameters used.
  - Training and validation loss plots.
  - Performance comparison of all models.
  - Best-performing model and its evaluation.
- A function that loads a trained model and computes the Macro F1-score for `test.json`.

2 Task - Aspect Based Sentiment Analysis 35 Marks

The goal of this task is to perform **Aspect-Based Sentiment Analysis (ABSA)**. Students will use the same `train.json` and `val.json` files as in Task 1 but will preprocess them differently.

2.1 Dataset Description

Students will be provided with `train.json` and `val.json` files containing examples in the following format:

```
{
    'sentence_id': '1634',
    'sentence': 'The food is uniformly exceptional, with a very capable
       kitchen which will proudly whip up whatever you feel like eating,
       whether it's on the menu or not.',
    'aspect_terms': [
        {
            'term': 'food',
            'polarity': 'positive',
            'from': '4',
            'to': '8'
        },
        {
            'term': 'kitchen',
            'polarity': 'positive',
            'from': '55',
            'to': '62'
        },
        {
            ‘term’: ‘menu’,
            ‘polarity’: ‘neutral’,
            ‘from’: ‘141’,
            ‘to’: ‘145’
        }
    ],
    ‘aspect_categories’: [
        {
            ‘category’: ‘food’,
            ‘polarity’: ‘positive’
        }
    ]
}
```

2.2 Preprocessing

Students need to format the data such that each aspect term is treated as a separate instance. The formatted data should be structured as follows:

```
{
    ‘tokens’: [
        ‘The’, ‘food’, ‘is’, ‘uniformly’, ‘exceptional’, ‘with’, ‘a’, ‘very’,
            ‘capable’, ‘kitchen’, ‘which’, ‘will’, ‘proudly’, ‘whip’, ‘up’, ‘
            whatever’, ‘you’, ‘feel’, ‘like’, ‘eating’, ‘whether’, ‘its’, ‘on
            ’, ‘the’, ‘menu’, ‘or’, ‘not’
    ],
    ‘polarity’: ‘positive’,
    ‘aspect_term’: [‘food’],
    ‘index’: 1
}

{
    ‘tokens’: [
        ‘The’, ‘food’, ‘is’, ‘uniformly’, ‘exceptional’, ‘with’, ‘a’, ‘very’,
            ‘capable’, ‘kitchen’, ‘which’, ‘will’, ‘proudly’, ‘whip’, ‘up’, ‘
            whatever’, ‘you’, ‘feel’, ‘like’, ‘eating’, ‘whether’, ‘its’, ‘on
            ’, ‘the’, ‘menu’, ‘or’, ‘not’
    ],
    ‘polarity’: ‘positive’,
    ‘aspect_term’: [‘kitchen’],
    ‘index’: 9
}

{
    ‘tokens’: [
        ‘The’, ‘food’, ‘is’, ‘uniformly’, ‘exceptional’, ‘with’, ‘a’, ‘very’,
            ‘capable’, ‘kitchen’, ‘which’, ‘will’, ‘proudly’, ‘whip’, ‘up’, ‘
            whatever’, ‘you’, ‘feel’, ‘like’, ‘eating’, ‘whether’, ‘its’, ‘on
            ’, ‘the’, ‘menu’, ‘or’, ‘not’
    ],
    ‘polarity’: ‘neutral’,
    ‘aspect_term’: [‘menu’],
    ‘index’: 24
}
```

Each aspect term is treated as an independent example, with its corresponding sentiment polarity and index in the sentence. The files should be saved as `train_task_2.json` and `val_task_2.json`.

2.3 Model Training

You are required to train a model using RNN, GRU, or LSTM with word embeddings from FastText, GloVe, or BERT embeddings. The architecture is flexible, allowing for creative design choices, which must be justified in the report. **(Simple model choices like using just RNN/LSTM/GRU are discouraged you will be judged on your creativity as well)**. Additionally, the report should include training and validation plots to support your analysis. You have to submit one model only.

2.4 Evaluation

The evaluation metric for this task is **accuracy**. The best-performing model should be saved for inference.

2.5 Testing and Model Inference

A `test.json` file will be provided during the demo. Students must implement a function that:

- Loads the trained model.
- Processes the `test.json` file.
- Returns the accuracy score.

2.6 Additional Task

After completing the above part, fine-tune BERT, BART, and RoBERTa for this task, and evaluate their performance on both the training and validation sets. Present the accuracy metrics for each model and include the corresponding training and validation loss plots. Ensure that the final report contains these results along with a comprehensive analysis.

2.7 Deliverables

You must submit the following:

- Preprocessed datasets: `train_task_2.json` and `val_task_2.json`.
- Implementation code for pre-processing, model training, and inference in a single file named `task2.py/task2.ipynb`.
- Best performing saved model.
- A detailed report including:
  - Explanation of preprocessing steps.
  - Model architectures and hyperparameters used.
  - Training and validation loss plots for your best performing model, BERT, BART and RoBERTa.
  - Evaluation metric on validation set for all the four models.
- A function that loads a trained model and computes the accuracy for `test.json`.

3 Task - Fine-tuning SpanBERT and SpanBERT-CRF 30 Marks

3.1 Task Description

In this task, you are required to fine-tune SpanBERT and SpanBERT-CRF for the question-answering task on the SQuAD v2 dataset. The objective is to extract the answer span from the given context as per the question asked. An example:

```
{
'context': 'Beyonc  Giselle Knowles-Carter (born September 4, 1981) is an
    American singer, songwriter, record producer and actress. Born and raised
     in Houston, Texas, she performed in various singing and dancing
    competitions as a child, and rose to fame in the late 1990s as lead
    singer of R&B girl-group Destiny\'s Child. Managed by her father, Mathew
    Knowles, the group became one of the world\'s best-selling girl groups of
     all time. Their hiatus saw the release of Beyonc \'s debut album,
    Dangerously in Love (2003), which established her as a solo artist
    worldwide, earned five Grammy Awards and featured the Billboard Hot 100
    number-one singles "Crazy in Love" and "Baby Boy".',

'question': 'When did Beyonce start becoming popular?',

'answers': {'text': ['in the late 1990s'], }
```

3.2 Dataset Description

The dataset used for this task is the Stanford Question Answering Dataset v2 (SQuAD v2). This dataset consists of question-answer pairs, including unanswerable questions where no valid answer exists in the given passage. Due to the large size of the dataset, you may use a subset of at least 15,000 samples for training.

3.3 Model Training

Both SpanBERT and SpanBERT-CRF should be fine-tuned using an appropriate subset of SQuAD v2. The training process must be well-documented, including hyperparameters, optimization techniques, and any preprocessing steps applied. Training and validation loss plots should be included in the report to illustrate model performance over time. Minimum epochs for training is 6.

3.4 Evaluation Criteria

The models will be evaluated using the *exact-match (EM)* metric which measures the percentage of predictions that exactly match the ground truth. This metric should be reported on the validation set to assess the effectiveness of each model. Use the following code for the metric

```
def exact_match_score(predictions, references):
    assert len(predictions) == len(references), "Lists must have the same length"
    matches = sum(p == r for p, r in zip(predictions, references))
    return matches / len(references) * 100  # Convert to percentage
```

3.5 Deliverables

You are required to submit the following:

- A detailed report including:
  - A description of the dataset and preprocessing steps.

    - Justification of model choices and hyperparameters.
    - Training and validation plots.
    - Comparative analysis of SpanBERT-CRF and SpanBERT.
    - Exact-match scores on the validation set. (Screenshot should be taken from code's output and the same output cell should be there in gc's sumbission)
- The code used for training and evaluation in a single file named `task3.py/task3.ipynb`.

Important Note

Evaluation scores contribute to the final grading of all tasks. Therefore, students are encouraged to optimize their models and training strategies to achieve the best possible performance.