RNN for Concept Tagging

Aniruddha Tammewar
6.05.19

Resources

- Git Repository: https://github.com/fruttasecca/concept-tagging-with-neural-networks
- Implements and compares variety of architectures/models for the task of concept tagging
- We’ll focus on the Neural Network Architectures
- Framework: Pytorch 0.4
- MOVIES corpus: tagged with concepts like movie.name, actor.nationality, etc

Pytorch Introduction

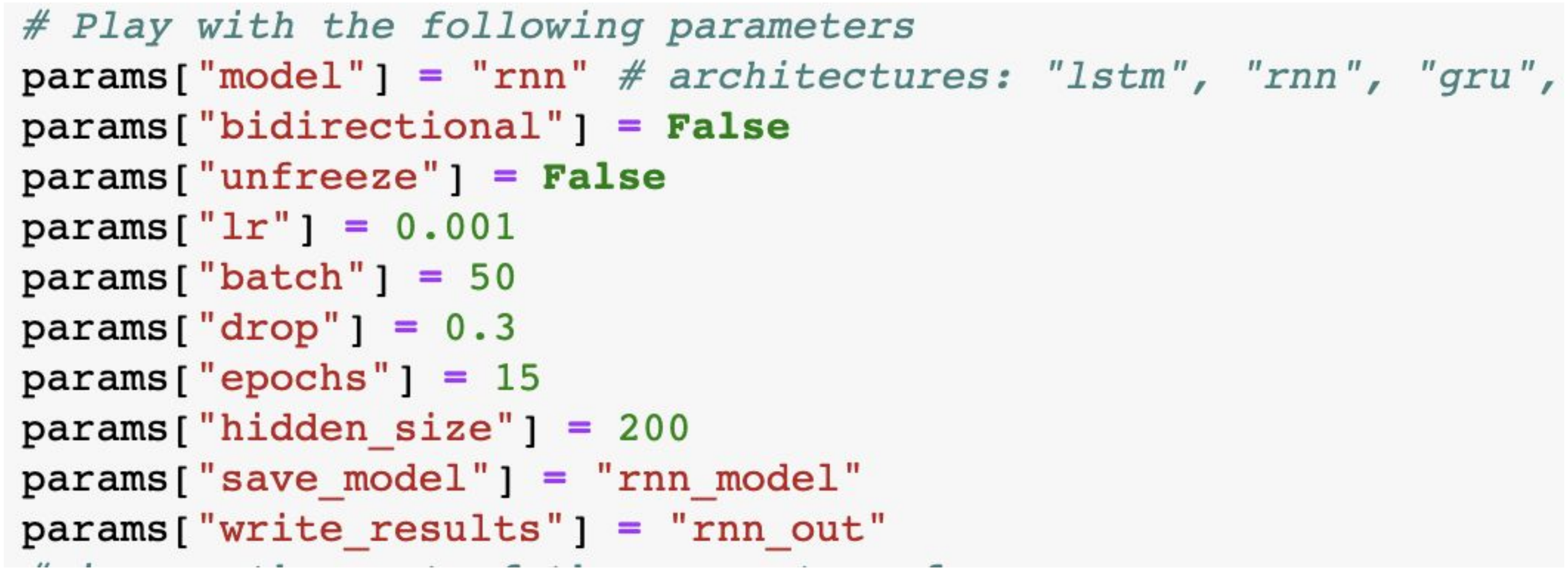
- https://pytorch.org/
- Latest stable version: 1.1
- Tutorials: https://pytorch.org/tutorials/
- Allows graph mode as well as eager mode
  - Graph mode: the complete computational graph is first defined, compiled then evaluate
  - Eager execution: allows to evaluate as you go without waiting for the graph to be built

Exercise 1: Examine the code, data

- Ipython Notebook: https://github.com/anutammewar/concept-tagging-with-neural-networks/blob/master/src/nn_lab.ipynb
- Keep it in the *src* directory
- How is the data stored in the pickle file?
  - What features are present
  - One row consists of one token or one sentence/sequence?
  - Size of train, test, dev sets
- # output labels
- Vocabulary size, embedding dimensions

Exercise 2: Play with architectures and parameters

- Explore Architectures: RNN, GRU, LSTM
- Parameters
  - Bidirectional: True/False
  - Unfreeze: True=embeddings are fine tuned
  - Lr: initial learning rate (0.1, 0.01, 0.001)
  - Drop: rate for dropout layers (0.3, 0.5, 0.7)
  - Hidden_size: size of the hidden layer (100, 200, 400)
  - Epochs: number of epochs (10, 15, 20, 25, ?)
- Try to replicate results from the paper: https://arxiv.org/pdf/1807.10661.pdf

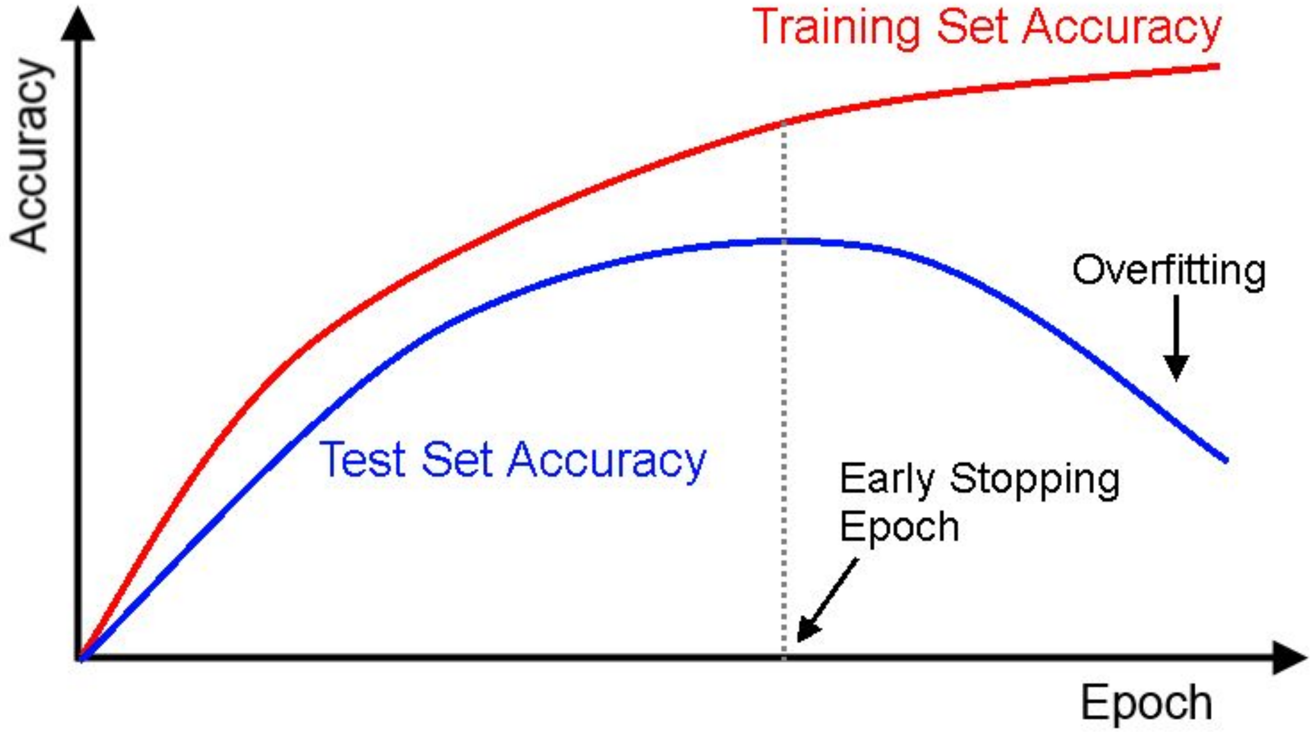


```python
# Play with the following parameters
params["model"] = "rnn" # architectures: "lstm", "rnn", "gru",
params["bidirectional"] = False
params["unfreeze"] = False
params["lr"] = 0.001
params["batch"] = 50
params["drop"] = 0.3
params["epochs"] = 15
params["hidden_size"] = 200
params["save_model"] = "rnn_model"
params["write_results"] = "rnn_out"
```

Expected effect of parameters

- Bidirectional: “True” should perform better as it considers context on both the sides while predicting a label for a particular word
- Unfreeze: “True” should perform better
    - The pre-trained embeddings are usually trained on a large dataset of various domains for the task of Language Modeling
    - Fine tuning helps embeddings to be altered to perform better on the target dataset and task
- Learning rate: a good fit has to be identified by experimenting (learning rate decay usually helps)
- Drop rate: a good fit has to be identified by experimenting (You can get a hint from bias-variance curves to see if a model is overfitting)
- Hidden_size: a good fit has to be identified by experimenting (again be sure not to overfit)
- Epochs: one way to identify a good value is by plotting error/accuracy curves for training and dev data with the number of epochs

Early Stopping



Exercise 3: Modifications

- Word Embeddings:
    - If w2v = None initialize random embedding matrix (torch.nn.Embedding,)
    - Vary dimensions
    - Results with the pre-trained embeddings should be better
    - Embeddings trained on a small dataset cannot capture semantics of the words very well

Additional Exercise

- Additional features (Advanced) :
  - POS tags, Named Entities tags, syntactic tags
  - NLP library like Spacy, NLTK if features are not provided
  - Create vocab (tag to index) for pos tags
  - Embeddings for POS
  - Concatenate word vector, POS vector
- Optimizers: sgd, adagrad, adam, etc. (torch.optim)
- Explore other architectures from the library: "lstm2ch", "encoder", "attention", "conv", "fcinit", "lstmcrf"

Project Possibilities

- Multi task learning for jointly predicting multiple NLU tasks (eg. concept tagging and Dialogue Act tagging)
- Experiment with different combination of features for a specific task
- Transfer learning (repurpose a model trained for one task to be applied for another task)
- Response generation using seq2seq architectures
- Implementing NN architectures for the various modules involved in the dialogue systems (i.e. besides NLU, Dialogue Manager, Natural Language Generation)
- Explore newer architectures like Transformers
- Explore various Word Embeddings like BERT, ELMo