Independent Study Report

Cheng Ding

I cooperate with my partners and complete an intelligent web crawler, which is based on reinforcement learning. This software is aimed at helping user to obtain target webs with a topic more efficiently. It can predefine the relevance value of web link candidates and only try to crawl the web links with high relevance value. In addition, the software can improve itself by learning the rewards of crawled webs.

I focus on predicting the relevance value of web link candidates that is called link_evaluation. Firstly, it obtains the url of web link candidate and the corresponding text that owns the hyperlink. Also with the crawling topic and a set of relevant web links, link_evaluation converts these attributes to numerical features, which are the edit distance and 2,3,4,5,6,7-gram appearance rate of web link and given relevant links, tf value, word similarity and word synonym of the text that owns the hyperlink and topic.

Based on the numerical features, link_evaluation applies a neural network with three fully connected layers, by Pytorch, to train the weights of features. With the trained weights, the model can be used to predict the relevance value of a web link before seeing the real content of this link. Besides, the neural network structure can be modified simply by the users.

link_evaluation

To calculate the relevance between a web link and a given topic.

get_feature(links, topic, relevant_urls)

To parse the web link and the text that owns the hyperlink to 10 numerical features, which are the edit distance and 2,3,4,5,6,7-gram appearance rate of web link and given relevant links, tf value, word similarity and word synonym of the text that owns the hyperlink and topic.

parameters: links: *dictionary*
    The key is the web link.
    The value is the text that owns the hyperlink.
  topic: *string*
    The given topic.
  relevant_urls: *list of tuples*
    The first element of tuple is the web link.
    The second element of tuple is the relevance score between this web link and the given topic
return: output: *dictionary*
    The key is the web link.
    The value are the numerical features.

generate_train_test(links, topic, relevant_urls, labels)

To separate the original data to train and test.

parameters: links: *dictionary*
The key is the web link.
The value is the text that owns the hyperlink.
topic: *string*
The given topic.
relevant_urls: *list of tuples*
The first element of tuple is the web link.
The second element of tuple is the relevance score between this web link and the given topic
Labels: *list*
Label the web link with 4 categories, 0, 1, 2, 3, which number is proportional to the relevance.
return: output: *None*
Generate the train set and test set for both features and labels.

train(epochs = 1000, learning_rate = 1e-3)
To train the weight of features using a neural network with three fully connected layers.

parameters: epochs: *number, optional*
The number is the epochs for training
learning_rate: *number, optional*
The number is learning rate for training
return: output: *None*
Generate a neural network model for web link evaluation.

test()
To test the neural network model.

parameters: None
return: output: *None*
Test the trained neural network model for web link evaluation and print the test accuracy

get_link_score(links, topic, relevant_urls)
To get the relevance score of web links.

parameters: links: *dictionary*
The key is the web link.
The value is the text that owns the hyperlink.
topic: *string*
The given topic.
relevant_urls: *list of tuples*
The first element of tuple is the web link.

The second element of tuple is the relevance score between this web link and the given topic

return: output: *dictionary*

The key is the web link.

The value is the relevance score, (0, 0.333, 0.666, 1)

Example:

```
>>> eva = link_evaluation()
>>> eva.generate_train_test(links, topic, relevant_urls, label)

>>> eva.train()
Epoch: 50 129 Loss: 135.996930539608
Epoch: 100 129 Loss: 131.46723574399948
Epoch: 150 129 Loss: 126.40633052587509
…

>>> eva.test()
The test accuracy is: 0.7878787878787878

>>> eva.get_link_score({"www.hbo.com": "series"}, "news", [("www.cnn.com", 1)])
{'www.hbo.com': 0.3333333333333333}

>>> eva.get_link_score({"www.nytimes.com": "new"}, "news", [("www.cnn.com", 1)])
{'www.nytimes.com': 1.0}
```