The assignment

Create a new model that is able to extract products from Furniture Stores.

Inputs

List of URLs from furnitures stores sites.

Solution

This task consist of 3 main subtasks:

1. Data retrieval
2. Data Preprocessing
3. Model Training

Let’s talk on every task in detail

Data Retrieval

Data retrieval was the most challenging part of the task, as we have to work with unstructured and unlabeled data sources.

At first, I checked every site with `requests` library to check if it works to filter out ones that work to use in future. Next I retrieved visible text part from each working website with `BeautifulSoup` library for scraping. This library is a good start for scraping, but it should be used with Selenium to handle text that is loaded via `JavaScript`. For this test assignment I used only `BeautifulSoup`.

Next I had to retrieve some labeled data (product names) from these website. Here I had 3 approaches:

1. Get list of keywords from the Net and filter all the next base on keywords. This approach didn’t work as I did not get exact name of products

2. I noticed that a lot of sites from the list were based on Shopify platform, meaning that they had something in common. So I found out that Shopify websites had `/sitemapl.xml` page. This page contains all the exact product names and I successfully parsed them. This was still not enough for creating dataset as I only got named entities but not other text connected to the product.

3. And here we go with the latest approach that is used for dataset creation. I also parsed products page links, choose 4-5 links from each sitemap page and

got some product description. To obtain description or part of it, I selected html tags, that containing text with minimum of 15 words. This helped to exclude category names and made me more sure that descriptions did not contain our named entities. So in the end of task I had dataset with next target field: `product_name` and `product_description`.

Data preprocessing

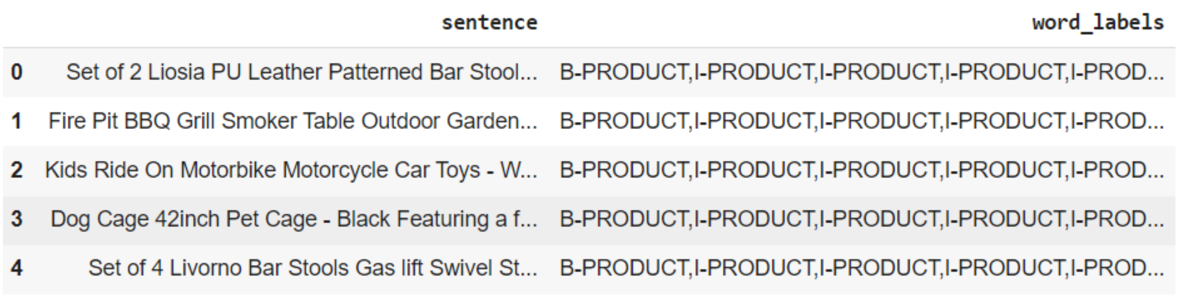
For current solution I concatenated product name on the first place and description on the second one for ease and speed of use, so now sentences have similar structure

In order to create dataset for NER model training we have to annotate each word in text with IOB labels. For this task we have three classes:

```
{'I-PRODUCT': 0, 'O': 1, 'B-PRODUCT': 2},
```

where B-Product class stands for the first word in the named entity, I-Product - rest of the words in entity, O for the rest of the words in the sentence.

After word annotation I returned everything back to sentence and created sequences of labels for each sentence



	sentence	word_labels
0	Set of 2 Liosia PU Leather Patterned Bar Stool...	B-PRODUCT,I-PRODUCT,I-PRODUCT,I-PRODUCT,I-PROD...
1	Fire Pit BBQ Grill Smoker Table Outdoor Garden...	B-PRODUCT,I-PRODUCT,I-PRODUCT,I-PRODUCT,I-PROD...
2	Kids Ride On Motorbike Motorcycle Car Toys - W...	B-PRODUCT,I-PRODUCT,I-PRODUCT,I-PRODUCT,I-PROD...
3	Dog Cage 42inch Pet Cage - Black Featuring a f...	B-PRODUCT,I-PRODUCT,I-PRODUCT,I-PRODUCT,I-PROD...
4	Set of 4 Livorno Bar Stools Gas lift Swivel St...	B-PRODUCT,I-PRODUCT,I-PRODUCT,I-PRODUCT,I-PROD...

Model training

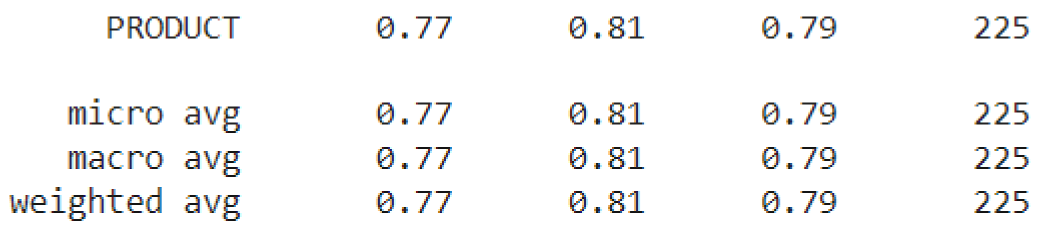
For current task I selected `BertForTokenClassification` and `BertTokenizerFast` with pretrained weights of `bert-base-uncased` from `transformers` library

I created custom `Dataset` and `DataLoader` classes from `pytorch` library. Inside dataset class there is a tokenizer than encodes sentences. Along with it there is also labels transformation, where we keep original label only for the first token from the word (as Bert can split word into several token and length of token should be equal to the length of labels).

Next step is defining the model. As I mentioned before I used `BertForTokenClassification` with pretrained weights of `bert-base-uncased`. The only thing that is need to be specified if number of classes of our model. We have 3 classes.

After that model is trained for 1 epoch with CrossEntropyLoss and Adam optimizer.

The results using seqeval library metrics are next:



```
     PRODUCT       0.77      0.81      0.79       225

   micro avg       0.77      0.81      0.79       225
   macro avg       0.77      0.81      0.79       225
weighted avg       0.77      0.81      0.79       225
```

Steps to improve the solution:

1. Improve scraping process to cover different types of websites
2. Use the whole visible text from website
3. Make deeper data preprocessing (e.x. work with punctuations)
4. Try other hyperparameters for model training
5. Try pretrained weights from different models

Resources:

1. Custom NER with BERT tutorial
2. Training Named Entity Recognition model with custom data using Huggingface Transformer
3. Beautiful Soup 4.12.0 documentation - Crummy