Coticeru Dragos-Ionut -- Product extraction model

To summarize my process of thinking, at first, I wanted to implement a model that extracted all the text content from the URLs given and predicted all the products shown on each page. However, I believe this wasn't an efficient idea.
As Veridion's goal is to build comprehensive profiles from large amounts of data, I wanted to aggregate all the pages from the sites. Therefore, I started by taking the main domain, checking if the connection was successful, and after that, recursively accessing every link displayed in the raw HTML content. I chose to keep track of the links in a Set, as it has O(1) time complexity for searching and adding.

I came up with the idea of taking into consideration only the URLs that contain '/product/' or '/products/'. From these URLs, I have extracted the first 30 words (as they will contain the product name) and loaded them into the model for predicting.

After the model has predicted the products, they are extracted and put in a dictionary along with their count. This way, each time a product that has already been added will have the count increased by one. In the end, we can print the most popular products by count.

I liked this implementation because it limited the effort of the model, therefore the output will get better. Moreover, the code is structured in such a way that for each text input, if the model has found a product, it stops adding words to the dictionary (because there should be only one product per text input; this way, we ignore duplicates and false positives).

Nonetheless, by only taking into consideration the first 30 words from each page, it will also lower the time it takes to annotate the datasets for model training, as it will be only one product per page and will usually be located at the beginning of the text input.

In order to handle exceptions, I took these measures:

If the URL contains '.jpg', '.png', or '.jpeg', I do not load it into the model, as it is a photo with no text content. For each URL, I check if the connection is successful.

Even though a main domain will have a successful connection, its pages might not be working. Therefore, I added an error count each time a connection is not successful. When it reaches 20, it skips to the next URL from the CSV file.

In order to not count duplicate products from the same page but with a different URL (it ends with #Maincontent or similar endings), I normalized the URLs so they are different. For this, I implemented the function normalize_url(url).

All the datasets were created manually by me, using doccano at the beginning, then switching to Inception.

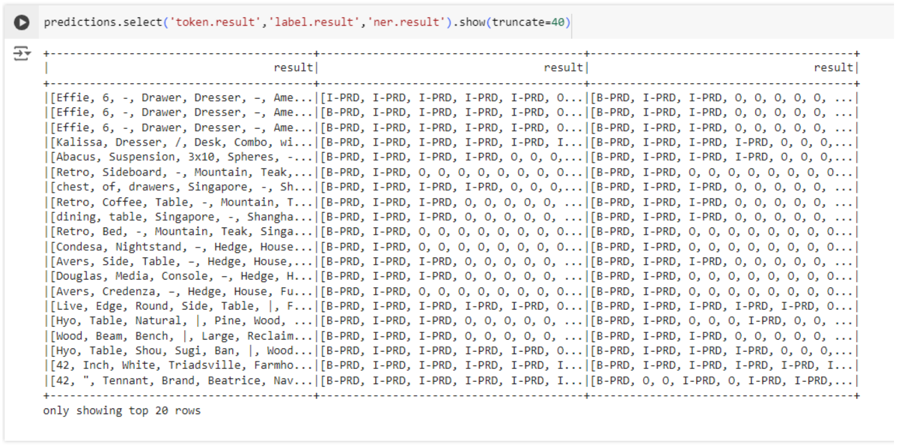
The model predictions are pretty good, considering the small dataset that I annotated for training:

```
Epoch 5/5 started, lr: 9.803922E-4, dataset size: 172


Epoch 5/5 - 5.52s - loss: 41.984108 - batches: 22
Quality on validation dataset (20.0%), validation examples = 34
time to finish evaluation: 0.38s
label    tp      fp      fn      prec     rec      f1
I-null   0       0       6       0.0      0.0      0.0
I-PRD    143     7       52      0.9533333        0.73333335       0.82898545
B-PRD    44      8       6       0.84615386       0.88     0.8627451
tp: 187 fp: 15 fn: 64 labels: 3
Macro-average    prec: 0.599829, rec: 0.5377778, f1: 0.567111
Micro-average    prec: 0.92574257, rec: 0.7450199, f1: 0.82560706
Quality on test dataset:
time to finish evaluation: 1.37s
label    tp      fp      fn      prec     rec      f1
I-PRD    281     26      176     0.9153094        0.61487967       0.73560214
B-PRD    109     13      14      0.89344263       0.88617885       0.8897959
tp: 390 fp: 39 fn: 190 labels: 2
Macro-average    prec: 0.90437603, rec: 0.7505293, f1: 0.82030153
Micro-average    prec: 0.90909094, rec: 0.67241377, f1: 0.7730426
```

After 5 epoch, It has a precision of 89% for correctly finding the beginning of a product, which will be the most important aspect for this algorithm.

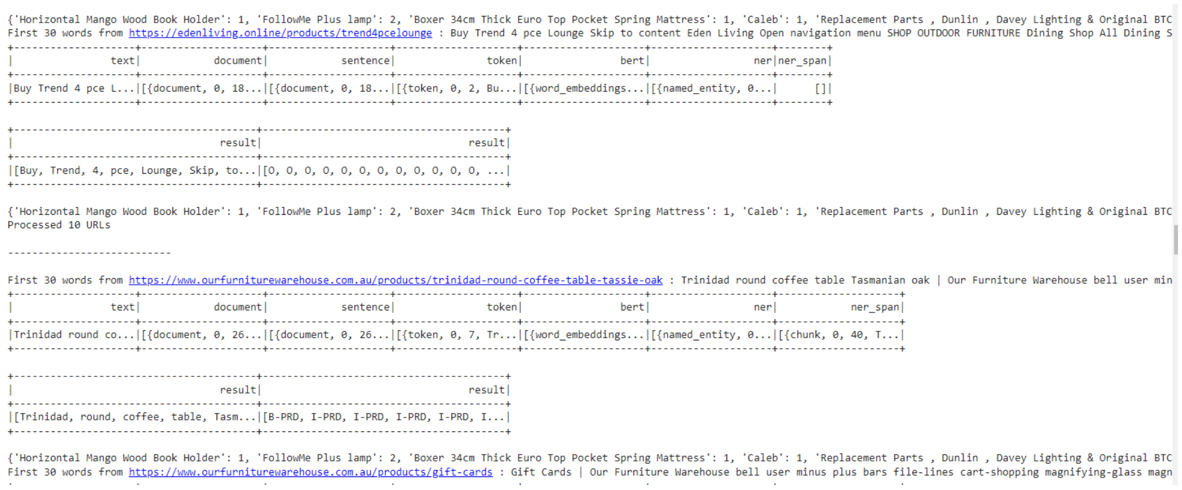
It performed good for the test dataset:



```
predictions.select('token.result','label.result','ner.result').show(truncate=40)
```

```
+----------------------------------------+----------------------------------------+----------------------------------------+
|                                  result|                                  result|                                  result|
+----------------------------------------+----------------------------------------+----------------------------------------+
|[Effie, 6, -, Drawer, Dresser, –, Ame...|[I-PRD, I-PRD, I-PRD, I-PRD, I-PRD, O...|[B-PRD, I-PRD, I-PRD, O, O, O, O, O, ...|
|[Effie, 6, -, Drawer, Dresser, –, Ame...|[B-PRD, I-PRD, I-PRD, I-PRD, I-PRD, O...|[B-PRD, I-PRD, I-PRD, O, O, O, O, O, ...|
|[Effie, 6, -, Drawer, Dresser, –, Ame...|[B-PRD, I-PRD, I-PRD, I-PRD, I-PRD, O...|[B-PRD, I-PRD, I-PRD, O, O, O, O, O, ...|
|[Kalissa, Dresser, /, Desk, Combo, wi...|[B-PRD, I-PRD, I-PRD, I-PRD, I-PRD, I...|[B-PRD, I-PRD, I-PRD, I-PRD, O, O, O,...|
|[Abacus, Suspension, 3x10, Spheres, -...|[B-PRD, I-PRD, I-PRD, I-PRD, O, O, O,...|[B-PRD, I-PRD, I-PRD, I-PRD, O, O, O,...|
|[Retro, Sideboard, -, Mountain, Teak,...|[B-PRD, I-PRD, O, O, O, O, O, O, O, O...|[B-PRD, I-PRD, O, O, O, O, O, O, O, O...|
|[chest, of, drawers, Singapore, -, Sh...|[B-PRD, I-PRD, I-PRD, I-PRD, O, O, O,...|[B-PRD, I-PRD, I-PRD, O, O, O, O, O, ...|
|[Retro, Coffee, Table, -, Mountain, T...|[B-PRD, I-PRD, I-PRD, O, O, O, O, O, ...|[B-PRD, I-PRD, I-PRD, O, O, O, O, O, ...|
|[dining, table, Singapore, -, Shangha...|[B-PRD, I-PRD, I-PRD, O, O, O, O, O, ...|[B-PRD, I-PRD, I-PRD, O, O, O, O, O, ...|
|[Retro, Bed, -, Mountain, Teak, Singa...|[B-PRD, I-PRD, O, O, O, O, O, O, O, O...|[B-PRD, I-PRD, O, O, O, O, O, O, O, O...|
|[Condesa, Nightstand, –, Hedge, House...|[B-PRD, I-PRD, O, O, O, O, O, O, O, O...|[B-PRD, I-PRD, O, O, O, O, O, O, O, O...|
|[Avers, Side, Table, –, Hedge, House,...|[B-PRD, I-PRD, I-PRD, O, O, O, O, O, ...|[B-PRD, I-PRD, I-PRD, O, O, O, O, O, ...|
|[Douglas, Media, Console, –, Hedge, H...|[B-PRD, I-PRD, I-PRD, O, O, O, O, O, ...|[B-PRD, I-PRD, O, O, O, O, O, O, O, O...|
|[Avers, Credenza, –, Hedge, House, Fu...|[B-PRD, I-PRD, O, O, O, O, O, O, O, O...|[B-PRD, I-PRD, O, O, O, O, O, O, O, O...|
|[Live, Edge, Round, Side, Table, |, F...|[B-PRD, I-PRD, I-PRD, I-PRD, I-PRD, O...|[B-PRD, I-PRD, I-PRD, I-PRD, I-PRD, O...|
|[Hyo, Table, Natural, |, Pine, Wood, ...|[B-PRD, I-PRD, I-PRD, O, O, O, O, O, ...|[B-PRD, I-PRD, O, O, O, I-PRD, O, O, ...|
|[Wood, Beam, Bench, |, Large, Reclaim...|[B-PRD, I-PRD, I-PRD, O, O, O, O, O, ...|[B-PRD, I-PRD, I-PRD, O, O, O, O, O, ...|
|[Hyo, Table, Shou, Sugi, Ban, |, Wood...|[B-PRD, I-PRD, I-PRD, I-PRD, I-PRD, O...|[B-PRD, I-PRD, I-PRD, I-PRD, O, O, O,...|
|[42, Inch, White, Triadsville, Farmho...|[B-PRD, I-PRD, I-PRD, I-PRD, I-PRD, I...|[B-PRD, I-PRD, I-PRD, I-PRD, I-PRD, I...|
|[42, ", Tennant, Brand, Beatrice, Nav...|[B-PRD, I-PRD, I-PRD, I-PRD, I-PRD, I...|[B-PRD, O, O, I-PRD, O, I-PRD, I-PRD,...|
+----------------------------------------+----------------------------------------+----------------------------------------+
only showing top 20 rows
```

Nonetheless, the full pipeline works, extracting products from every page from each site (I limited it to 5 pages per each main domain). It successfully extracts products and adds them to the dictionary, along with their count.



```
{'Horizontal Mango Wood Book Holder': 1, 'FollowMe Plus lamp': 2, 'Boxer 34cm Thick Euro Top Pocket Spring Mattress': 1, 'Caleb': 1, 'Replacement Parts , Dunlin , Davey Lighting & Original BTC
First 30 words from https://edenliving.online/products/trend4pcelounge : Buy Trend 4 pce Lounge Skip to content Eden Living Open navigation menu SHOP OUTDOOR FURNITURE Dining Shop All Dining S
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------+
|                text|            document|            sentence|               token|                bert|                 ner|ner_span|
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------+
|Buy Trend 4 pce L...|[{document, 0, 18...|[{document, 0, 18...|[{token, 0, 2, Bu...|[{word_embeddings...|[{named_entity, 0...|      []|
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------+

+----------------------------------------+----------------------------------------+
|                                  result|                                  result|
+----------------------------------------+----------------------------------------+
|[Buy, Trend, 4, pce, Lounge, Skip, to...|[O, O, O, O, O, O, O, O, O, O, O, O, ...|
+----------------------------------------+----------------------------------------+

{'Horizontal Mango Wood Book Holder': 1, 'FollowMe Plus lamp': 2, 'Boxer 34cm Thick Euro Top Pocket Spring Mattress': 1, 'Caleb': 1, 'Replacement Parts , Dunlin , Davey Lighting & Original BTC
Processed 10 URLs

--------------------------

First 30 words from https://www.ourfurniturewarehouse.com.au/products/trinidad-round-coffee-table-tassie-oak : Trinidad round coffee table Tasmanian oak | Our Furniture Warehouse bell user min
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+
|                text|            document|            sentence|               token|                bert|                 ner|            ner_span|
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+
|Trinidad round co...|[{document, 0, 26...|[{document, 0, 26...|[{token, 0, 7, Tr...|[{word_embeddings...|[{named_entity, 0...|[{chunk, 0, 40, T...|
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+

+----------------------------------------+----------------------------------------+
|                                  result|                                  result|
+----------------------------------------+----------------------------------------+
|[Trinidad, round, coffee, table, Tasm...|[B-PRD, I-PRD, I-PRD, I-PRD, I-PRD, I...|
+----------------------------------------+----------------------------------------+

{'Horizontal Mango Wood Book Holder': 1, 'FollowMe Plus lamp': 2, 'Boxer 34cm Thick Euro Top Pocket Spring Mattress': 1, 'Caleb': 1, 'Replacement Parts , Dunlin , Davey Lighting & Original BTC
First 30 words from https://www.ourfurniturewarehouse.com.au/products/gift-cards : Gift Cards | Our Furniture Warehouse bell user minus plus bars file-lines cart-shopping magnifying-glass magn
```

This was the summary that illustrated the most significant facts of the creation of the model and algorithm. Moreover, I have shown my thoughts and ideas in chronological order. Please keep in mind that this is my first time interacting the NLP and datasets annotation, therefore I had to look up a lot of documentation. I have marked only the significant documentation below. Moreover, I have mainly used the documentation presented in the challenge

https://towardsdatascience.com/named-entity-recognition-ner-with-bert-in-spark-nlp-874df20d1d77

and the free annotation tools:

https://medium.com/dida-machine-learning/the-best-free-labeling-tools-for-text-annotation-in-nlp-844525c5c65b

My ideas in chronological order:

After reading the assignment, I opened some of the URLs to get a better idea of how they look. As suggested in the assignment, some of them contained furniture products, some of them were unavailable and some of the site didn't even work at all.

Before I start thinking about a solution, I like to deeply analyze my data set until I feel comfortable that I know what I am dealing with.

I created short lists with urls that work and contain products, the urls that work but do show any product and with urls that do not work at all.

Since I am not very familiar with NER, I started watching a basic video to get a better understanding of how it works:

https://www.youtube.com/watch?v=2XUhKpH0p4M

https://www.youtube.com/watch?v=h4rI8v6UjV0

https://www.youtube.com/watch?v=fEU37G70SFc

1st method: Simple Lookup (add words in the vocabulary and just check if the words appear in a new search). I do not like this idea since it implies hard coding.

2nd method: Rule based NER: Find a rule for the entities that you are searching for

3rd method: Machine Learning: Use Conditional Random Fields (BERT)

First approach:

Extract the HTML of 100 sites and store it in a dataset.

Find a way to locate the Products in these examples

Train the model on these sites.

Use the model to test new sites.

Evaluate the results.

Create a dictionary of products that contain name and counter. For each new product found, check whether it already exists in the list (thus increase the counter) or if it doesn't exist, create a new element with that name and counter 1.

When searching in a site, do not count the name of 1 product multiple times (for example here https://4-chairs.com/products/mason-chair the product 'Mason Chair' appears in the title and in the description of the item). Use a list of items every time you search an URL and add every item only once.

My first idea of how to search for products is to use the raw html and look for titles and headers like:

<title> PRODUCT </title>

<h1>

<h2>

As I have seen many sites use this kind of approach to tag their products

My second idea is to extract the text from the html and just tag the products and let the model get the right products

Trying to find a rule for entity recognition:

https://www.youtube.com/watch?v=uj_bbI3Ao-s

Basic operations (annotation, tokens and pipelines):

https://www.youtube.com/watch?v=fsS057SNFtg

BIO training:
https://www.youtube.com/watch?v=7CRyqwCZFY0

https://towardsdatascience.com/named-entity-recognition-ner-with-bert-in-spark-nlp-874df20d1d77

I looked up free annotation tools to tag the product entities:

https://medium.com/dida-machine-learning/the-best-free-labeling-tools-for-text-annotation-in-nlp-844525c5c65b

For my case it seems that doccano would be the best choice.

Web scraping with Python BeautifulSoup

https://www.youtube.com/watch?v=QhD015WUMxE

I have tried to scrape all the content from the pages, however there was too much information to tag entities.

Another idea I had: for each site, look how the main product is displayed (until which class). Click all the links and see if the link that u clicked contains that class. If not, stop. If yes, add the link to a list and search for the title of the same product. If all the sites have already been searched, stop.

For test data: Only use the main page and tag the title entity. Add the code of recursion searching after you implement the model.

Take in consideration only the links that contain a relative path. Therefore, search for

```
href="/
```

This idea limits the effort of the model, because it does not have to search for multiple products on the same page. It limits it to finding only 1 product per page. It does not matter if the main page does or does not contain a main product (if it does not contain a main product, the model will look for recommendations and will take a product from there)

This also limits the work of entity tagging, as you only have to tag 1 entity per website, thus making the process quicker.

A new idea: All the URLs that go into the model should contain '/product/' or '/products/'

https://www.youtube.com/watch?v=teHDlOzfN-A

I need a function that looks for all the links in a URL.

Installing sparknlp:
https://github.com/JohnSnowLabs/spark-nlp/discussions/1022%20GitHubGitHub%20How%20to%20correctly%20install%20Spark%20NLP%20on%20Windows%208%20and%2010%20%C2%B7%20Discussion%20#1022%20%C2%B7%20JohnSnowLabs/spark-nlp%20State%20of%20the%20Art%20Natural%20Language%20Processing.%20Contribute%20to%20JohnSnowLabs/spark-nlp%20development%20by%20creating%20an%20account%20on%20GitHub.%20Veysel%20%2010%20minutes%20ago%20And%20this%20is%20how%20it%20is%20being%20done%20in%20Docker%20https://github.com/JohnSnowLabs/spark-nlp/discussions/1714

Found this tutorial and started the sparknlp session in colab

https://www.youtube.com/watch?v=F2ph02HWWAo

aggregate of the urls found on the main domain

some are the same pages with # different at the end

we normalize the urls:

normalized_url = normalize_url(url)

put a dictionary with all the main languages for "product" and "products"

Use recursion to look for all the pages on the site. Aren't there too many pages that are unused? The amount of pages that do not show a main product is very limited compared to the amount of products displayed, so we can afford to search to include them in the set (the time search does not increase significantly). The urls not containing a main product is less than 10% from my tests.

Sets have O(1) time complexity for addition and search.

In the ml only include the text from urls that have "/products/"

Also added urls that contain /product/

If the url contains ".png" or ".jpg" or ".jpeg" remove it, because it will lead to a picture with no text.

Idea to add: also include the price and make a search engine to scan for the cheapest site from where you can buy a specific product

Idea to add: also include different languages