Flipkart Extract-a-Thon

Text Problem Statement

Team: **Triodes**

Naman Jaswani
(*namanjaswani@iisc.ac.in*)

Introduction:

The task in hand for this hackathon (Extract-a-Thon) is to extract attributes from product description. i.e given a (text) product description, we need to tag various attributes from description(text).

Overall Approach:

Given that we would only be given product description in test data, my approach was simple. I decided to make it into a 'Supervised' learning task, by first annotating the given unsupervised data (product descriptions) using the sample values for each attribute in each category. Annotation was done completely algorithmically

Now that we have labeled dataset, I trained a simple sequential model (Bidirectional LSTM) on *TPU v3-8 (available freely on kaggle kernels)* [1], as a supervised NER task.

Environment Details:

Language : Python
Environment: Jupyter notebook (.ipynb)
Tensorflow : 2.4.1
Keras: 2.4.3
Hardware (*Kaggle Kernels*):
16GB RAM
TPU v3-8

Detailed Approach:

Overall approach can be divided in two sub-tasks.

1. *Dataset Generation*

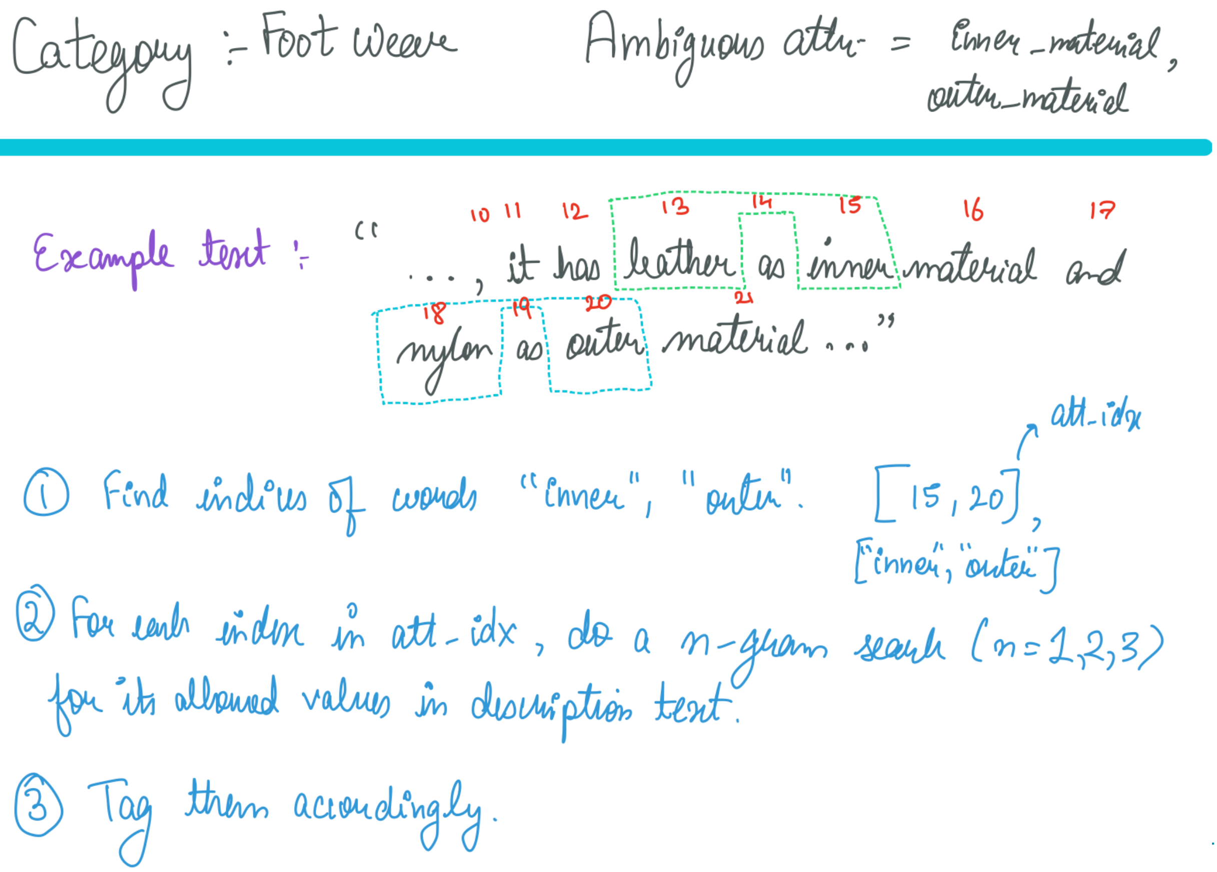
***a)* Tagging General attributes:**

Since manually labelling the entire unsupervised data is not practical. Hence we applied an algorithmic approach to annotate the entire dataset. The algorithm is as follows:

For each product description in *unsup.csv* , a word-by-word annotation is made with attribute values from the given *Attribute-values (5 csv files, one for each category was given).* Whenever the description word and attribute value are equal, the word is tagged with ′ B_* ′ and followed by the tag, indicating the beginning of this attribute. If the attribute value contains more than one word, then the subsequent words are tagged with ′ I_* ′ followed by tag. All other words which do not match any attribute value are tagged ‘O’.

b) **Tagging similar attributes:**

Now in order to tag similar attributes, eg. [outer_material and inner_material for foot wear category], n-gram analysis was used to find values for each of these *ambiguous* attributes. For better clarity, refer below handwritten note as an example.


10
11
12
13
14
15
16
17
18
19
20
21

Category :- Foot Weare   Ambiguous attr- = inner_material, outer_material

Example text :- "..., it has leather as inner material and nylon as outer material ..."

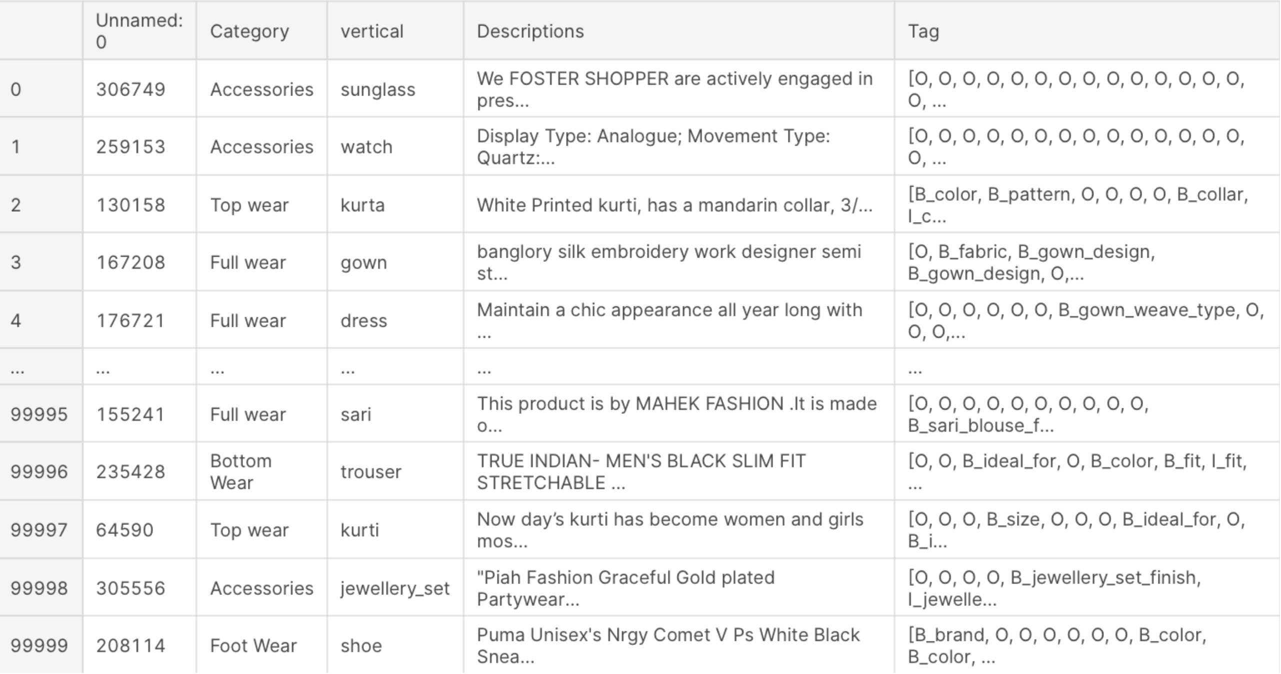
① Find indices of words "inner", "outer". [15, 20], ["inner", "outer"]
att_idx

② For each index in att_idx, do a n-gram search (n=1,2,3) for its allowed values in description text.

③ Tag them accordingly.

2. *Modeling*

The final labeled training dataset looks like below:

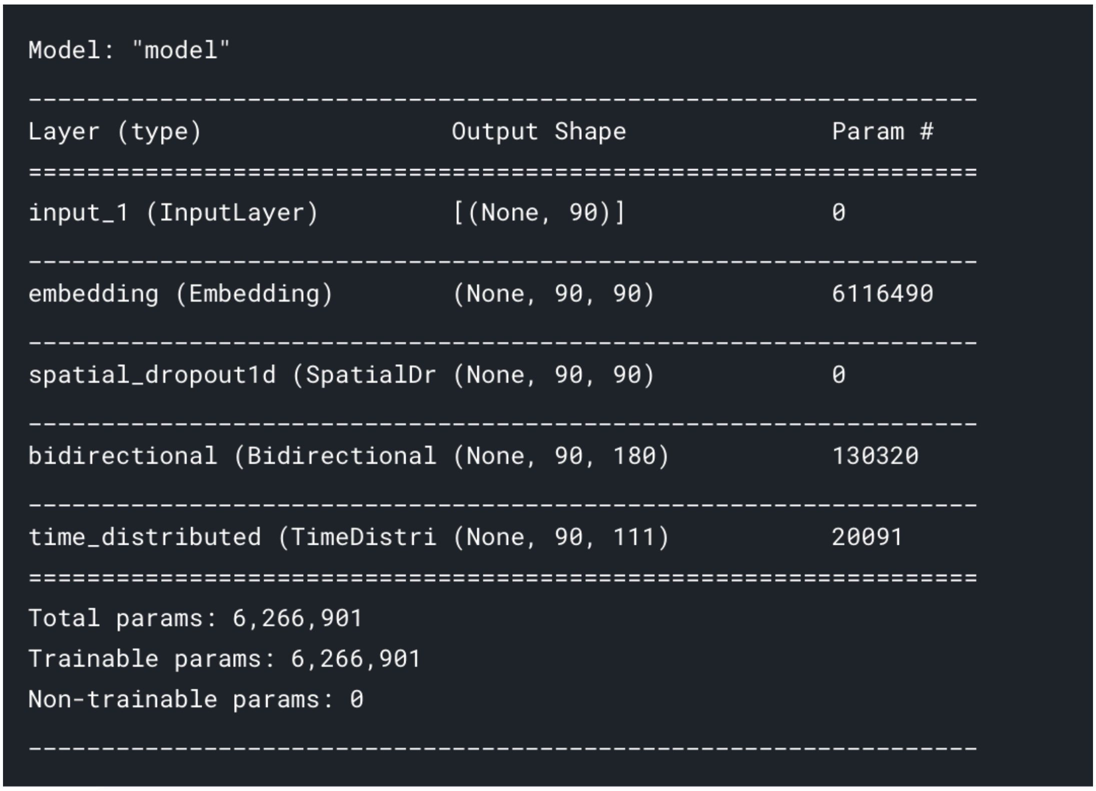


	Unnamed: 0	Category	vertical	Descriptions	Tag
0	306749	Accessories	sunglass	We FOSTER SHOPPER are actively engaged in pres...	[O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, ...
1	259153	Accessories	watch	Display Type: Analogue; Movement Type: Quartz:...	[O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, ...
2	130158	Top wear	kurta	White Printed kurti, has a mandarin collar, 3/...	[B_color, B_pattern, O, O, O, O, B_collar, I_c...
3	167208	Full wear	gown	banglory silk embroidery work designer semi st...	[O, B_fabric, B_gown_design, B_gown_design, O,...
4	176721	Full wear	dress	Maintain a chic appearance all year long with ...	[O, O, O, O, O, O, B_gown_weave_type, O, O, O,...
...	...	...	...	...	...
99995	155241	Full wear	sari	This product is by MAHEK FASHION .It is made o...	[O, O, O, O, O, O, O, O, O, O, B_sari_blouse_f...
99996	235428	Bottom Wear	trouser	TRUE INDIAN- MEN'S BLACK SLIM FIT STRETCHABLE ...	[O, O, B_ideal_for, O, B_color, B_fit, I_fit, ...
99997	64590	Top wear	kurti	Now day's kurti has become women and girls mos...	[O, O, O, B_size, O, O, O, B_ideal_for, O, B_i...
99998	305556	Accessories	jewellery_set	"Piah Fashion Graceful Gold plated Partywear...	[O, O, O, O, B_jewellery_set_finish, I_jewelle...
99999	208114	Foot Wear	shoe	Puma Unisex's Nrgy Comet V Ps White Black Snea...	[B_brand, O, O, O, O, O, O, B_color, B_color, ...

Next task is to train a sequential model for NER (Named Entity recognition) task.

For that, the entire model training algorithm is as follows:
Note: A single Model was trained on 20000 samples from each category.

1. Preprocessed the annotated data to get a csv file with each row containing one word in 'Word" column with one tag in 'Tag' Column, all these word-tag rows are further grouped by description number.
2. Vocabulary of unique words is made using all train + test data.
3. This data is further processed to get list of (word,tag) tuples as one description.
4. These words and tags from each description number are then encoded into separate lists of indexes (integers) so that they can be given as an input to embedding layer in model. Now we have a list of list of indexes representing collection of descriptions encoded into integers, same with tags.
5. These encoded lists are then padded to make each list of same length, since BiLSTM takes inputs of fixed lengths.
6. After that these encoded inputs and corresponding outputs are divided into train-validation sets and then given as inputs to below model:



```
Model: "model"
-----------------------------------------------------------------
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 90)]              0
-----------------------------------------------------------------
embedding (Embedding)        (None, 90, 90)            6116490
-----------------------------------------------------------------
spatial_dropout1d (SpatialDr (None, 90, 90)            0
-----------------------------------------------------------------
bidirectional (Bidirectional (None, 90, 180)           130320
-----------------------------------------------------------------
time_distributed (TimeDistri (None, 90, 111)           20091
=================================================================
Total params: 6,266,901
Trainable params: 6,266,901
Non-trainable params: 0
-----------------------------------------------------------------
```

And thats is !

References:

[1] Kaggle kernels https://www.kaggle.com/code