An end to end approach for brand recognition in product titles with BI-LSTM-CRF

Mohamed Annis Souames, Larbi Abderrahmane Mohammedi

An end to end approach for brand recognition in product titles with BI-LSTM-CRF

Mohamed Annis Souames
Industrial Engineering Department
National Polytechnic School
Algiers, Algeria
mohamed_annis.souames@g.enp.edu.dz

Larbi Abderrahmane Mohammedi
Industrial Engineering Department
National Polytechnic School
Algiers, Algeria
larbi_abderrahmane.mohammedi@g.enp.edu.dz

***Abstract*—The aim of this paper is to describe an end to end approach using different deep learning architectures to detect brand names in product titles from online stores and online retailers such as Amazon, Ebay, etc. In this paper we developed a named entity recognition model based on a Bi-LSTM architecture with a conditional random field layer using Flair framework, we also explain how the dataset was curated, cleaned and augmented to improve the model performances. Finally, we compare the trained model against few other models trained using the Spacy framework. Our model gave a relatively high f1-score of 0.83 with a good generalisation to real world cases.**

***Index Terms*—Named entity recognition, Bi-LSTM, CRF, Flair, Spacy.**

I. Introduction

Tagging brands in user generated content and more specifically in product titles is important for several business applications and is mainly used in e-commerce to understand customers shopping behaviour, search queries and to analyse the performance of different brands in a shop.

However, this task is tedious and requires either manual tagging by the user or the online store owner, or searching in large databases for the brand name that appears in a product title. This task can be automated to a large extent using named entity recognition models that can tag entities in a given document. For example, given the input : *New Jean **Zara** Blue Extra Slim*, the model should be able to tag **Zara** as a brand from that product title. This can be quite beneficial for different use cases as described above but it is also quite challenging to build a robust NER model given several challenges which will be detailed in the following section.

II. Current challenges

There is a number of challenges related to brand tagging in product titles which we will detail in this section :

- **Unstructured syntactic structure :** Product titles are highly unstructured pieces of information, there is no universal structure for writing a product title, hence, the same product might have different titles : "Skinny Jean Zara Brand New" is equivalent to "Zara Brand New Skinny Jeans".
- **Brand names are complex** : Brands do not share the same word shape, some brands contain numbers, other brands contain more than 3 words, and several brands do not exist in the English dictionary as words, so we are faced with another sub problem : brands are rare words.
- **Data scarcity** : There are several datasets for named entity recognition but all of these datasets are either related to news, literature or Wikipedia articles. Even though some e-commerce datasets exists such as the "Amazon Product Reviews" dataset curated by Jianmo Ni [6] or few other datasets found on Kaggle website, but all of them are not suitable for direct usage in named entity recognition tasks.

In this paper we explain how we tackled these challenges through data augmentation and deep neural networks.

III. Definition & concepts

Before diving into our method, we will begin by defining what is named entity recognition, bidirectional LSTM and conditional random fields.

A. Named Entity Recognition

Sequence tagging or named entity recognition is an important natural language processing task where the goal is to predict the class of each token as an entity, for instance, given the input sentence : Ahmed loves Algiers, the sequence tagger (also known as the named entity recognition model) should tag Ahmed as a *person* and Algiers as a *location*.

It is essentially a classification problem where given a sequential input and it's corresponding annotation, the NER model will learn and output a sequence of labels for each token w_i in a sentence of length N.

NER systems expect an annotated input with IOB scheme,where *B-ENTITY* is the beginning of an entity, *I-ENTITY* is used when the token is inside an entity, *O* is used to tag a token as others. For instance : *Annis Lives in New York* would have the following annotation : B-PER O O B-LOC I-LOC.

In our case, we are only interested in the *"BRAND"* entity, so we will have 3 possible labels for each token : *"B-BRAND", "I-BRAND"* and *"O"* to denote when a token is not a brand. Given that most of product titles have only one brand and other tokens are not interesting, we will have more *"O"* labels than other labels, therefore this classification task is highly imbalanced and the general metric for evaluation in this case is the **f1-score**.

B. Bidirectional LSTMs

Recurrent neural networks have been used extensively in natural language processing tasks because they can keep a memory of historical informations (for instance : previous words) at each step t.

Long Short Term Memory cells are a special type of gated recurrent neural networks that work well with finding dependencies in a long sentence, the hidden state h at a given step t, denoted h_t is computed in this manner :

$$\mathbf{h_t} = f(\mathbf{W_x}x_t + \mathbf{W_h}h_{t-1}) \quad (1)$$

$$\mathbf{y_t} = g(\mathbf{W_y}h_t) \quad (2)$$

Where W_x, W_h, W_y are the weight matrices learned by the model and f, g are the activation functions.

By stacking two layes of LSTMs : one receiving information from past input features h_{t-1} (the forward LSTM) and the second getting information from the future input features h_{t+1} we get a bidirectional LSTM network that can learn current features h_t based on historic and future informations. The bidirectional LSTMs have been used in the recent years for sequence tagging tasks and yielded good results [4].

C. Conditional Random Fields

Conditional random fields are a special type of Markov random fields that can take into account neighboring tags to predict a given tag and give some consistency, for instance, a CRF model can learn that it's more likely that *B-BRAND* will be followed either by *I-BRAND* tag or by *O* tag rather than a second *B-BRAND* tag. CRF combined with word features such as the word shape, N gram features, previous tags, next tags, etc, have been used several times in the past for named entity recognition and resulted in higher accuracy compared to other classical classification approaches such as SVM, Decision trees or other models.

Mathematically, the conditional random field compute a conditional probability distribution :

$$P(y|x) = \frac{exp(score(x, y))}{\sum_{y'} exp(score(x, y'))} \quad (3)$$

Where score is the log of potentials ϕ_i in the CRF :

$$score(x, y) = \sum_i log\phi_i \quad (4)$$

D. BI-LSTM-CRF

An important model in sequence tagging that has been used recently is the combination of a BI-LSTM network with a final CRF layer, the BI-LSTM part is used to learn features for each token using historic informations and future informations, while the CRF layer is trained to learn a good tagging strategy by taking into context neighboring tags, this combination outperforms simple BI-LSTM architectures or vanilla CRF models in named entity recognition tasks scoring an f1 score of 88.83 on the CoNLL 2003 dataset (used for named entity recognition benchmarks) [4].

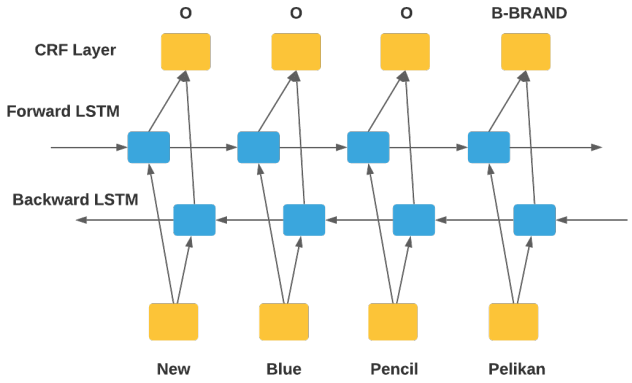

Fig. 1. An example of a BI-LSTM Network with a final CRF Layer

IV. Our Method

In this section, we will discuss the data set used to train the model, the data cleaning process, the main transformations applied to augment the data and the models that were trained.

A. Data curation

We tried multiple data sets before finding the "Amazon Product Reviews" data set curated by Jianmo Ni [6] which has over 1 million products. However, we only downloaded data for office supplies, computers, food & drinks and industrial tools. We ended up with over 150k products after joining all the products for each category.

B. Data Cleaning

We cleaned the data by removing the products with brands having special characters such as "+, @, #, etc." We, also replaced the HTML encoding codes found in the title by their textual representation such as "& to &".

Initially, using this dataset, the f1 score was over 80%, but the models we tested were over-fitting on titles having brands as their first word. We found that our dataset was imbalanced, more than 80% of the products included had the brand at the beginning of the title, thus, we decided to augment the data by shifting brands positions strategically, the data augmentation process is discussed below.

C. Data augmentation

We implemented a logic to sample the data, which shifts the position of the brand from the beginning to the end or the middle.

First, we split our data into three subsets A_1, A_2, A_3 : A_1 contains products having brand in the beginning, A_2 in the middle and A_3 at the end. As shown in the pseudo-code below :

$N = length(products)$
for $k \leftarrow 1$ to N **do**
  if `products[k] has brandFirst` **then**

$productsFirst \leftarrow products[k]$
else
$remainingProducts \leftarrow products[k]$
end if
end for

Then, we applied a transformation to the products having the brand in the beginning according to this logic :

$groups = productsFirst.groupby('brand')$
for `group in groups` **do**
$counter \leftarrow 0$
$halfLen \leftarrow (int)group.shape[0]/2$
for `idx,row in group.iterrows()` **do**
$counter \leftarrow counter + 1$
if $counter \leq halfLen$ **then**
$newStartBrand \leftarrow row$
else
$toSample \leftarrow row$
end if
end for
end for

Now that we have our dataframe **toSample**, we just need to shift the brand from the beginning to the middle and the end. We transformed our product title into a list then we shift it randomly to the middle or the end.

D. Modeling

In this project we used two well-known NLP frameworks : spaCy and Flair to design and train NER models easily along with Gensim to train custom word embeddings.

1) Word embeddings : Word2Vec was used as an embedding algorithm with the Gensim library to train custom embeddings on the product titles. We trained the model for 10 epochs and used an embedding dimension of 200, we trained the embeddings for words that repeated at least two times (min_count=2 in Gensim). Unfortunately, over 70% of the tokens were unique and were replaced by a null vector instead. The custom embeddings gave interesting results :

$$\mathbf{similarity}(Dell, Lenovo) = 0.89$$
$$\mathbf{similarity}(Staedtler, Pencil) = 0.71$$

The model learned linear relationships between products, brands and different attributes. In this paper we used both GloVe embeddings with 6 billion tokens and 300 dimension as well as the custom embeddings we trained.

2) spaCy: spaCy is an industrial level NLP framework offering several tools and pretrained models to achieve different tasks such as named entity recognition. In this project, we did not use a pretrained model due to the challenges with the unstructured nature of product titles described above, we decided to train a blank model using the standard architecture with few adjustments to the model parameters. The spaCy model is composed of different trainable components, the tokenization and embedding component is : 1) tok2vec which uses hash embeddings for prefix, suffix, normalized token and the word shape features, it also accepts static pretrained word embeddings as an additional input. 2) transformer : In this case the embedding is done using a transformer architecture such as **BERT**, or other transformer architectures, we did not use these architectures due to the large model size and the limited computation power available in our case.

The second component in spaCy is the ner component which is based on a variant of LSTMs known as Stack-LSTM and are used for dependency parsing [3]. The stack LSTM learns transitions from different states (for instance : [NULL] → [BRAND]) and is well suited for named entity tasks.

It is worth noting that for spaCy we use a different format for data than the IOB scheme, we use an offset format as the following :

```
(
    "Computer Dell Inspiron Intel I7",
    "entities" : [(8,11,BRAND)]
)
```

We generate a tuple for each title in our dataset, we then convert the data in the offset format to a JSON file and a spaCy binary file.

We trained two different spaCy models for 10 epochs with a batch size of 32, one using only custom embeddings we trained with Word2Vec as input, and the second using both our custom word embeddings and the hash embedding network in spaCy.

3) Flair: The Flair framework is a new NLP framework built as a research project [1]. It can be used for text classification, embeddings, sequence tagging (named entity recognition) and different other tasks. The sequence tagger model in Flair is based on a BI-LSTM architecture that can be supplemented with a crf layer at the end of the network. The framework is specifically good with named entity recognition and has a nice high level API. Flair also accepts different embeddings such as GloVe [5], FastText [2] and transformer based embeddings, it also has custom pretrained embeddings known as **Flair Embeddings**.

In our project we used a stacked embedding composed of GloVe 6B tokens pretrained embeddings and two pretrained Flair embeddings : one trained on news in a forward way and the second trained also on news articles but in backward. The three embeddings are then summed into one embedding that is passed to the sequence tagger model (BI-LSTM + CRF) with a batch size of 32 and a decaying learning rate.

One problem we faced with Flair is that we had to train it for 40 epochs and since it uses PyTorch, the resulting model has a size of around 300 MB. Besides, we can't use our own Word2Vec embeddings trained on product titles as opposed to spaCy, we can however train our own Flair embeddings but we did not in our case.

V. Results

Table 1 shows the different models trained with spaCy and Flair with their corresponding parameters and f1 score.

Framework	Word embeddings	Architecture	Batch size	Epochs	Other parameters	**f1 score**
spaCy	Custom WE (Word2Vec)	transition based BI-LSTM	32	10	W = 128 Lr = 0.001	0.82
spaCy	Custom WE (Word2Vec)	Multi Hash Embedding + Maxout Window Encoder + Transition based BI-LSTM	32	10	W = 128 Lr = 0.001	0.84
Flair	GloVe + Flair embedding (news-forward) + Flair embedding (news-backward)	BI-LSTM + CRF layer	32	40	W = 256 Lr = 0.1 (decaying*)	0.84

TABLE I

TABLE 1. RESULTS OF THE DIFFERENT MODELS TRAINED WITH SPACY AND FLAIR (W IS THE WIDTH OF THE HIDDEN LSTM LAYER, LR IS THE LEARNING RATE USED), *IN THE FLAIR MODEL, THE LEARNING RATE DECAYS AFTER 4 EPOCHS WITH NO IMPORVEMENT IN THE MODEL PERFORMANCE BY AN ANNEALING FACTOR OF 0.5, DURING THE LAST 5 EPOCHS, THE LEARNING RATE WAS 0.015.

We can see that the model built with Flair has the same f1-score as the second one built with spaCy, however when testing the model we noticed that Flair had a slightly better generalization to real world examples. All models were shared on the Github repository of the project.

Below are some examples tested with the trained flair model (model N° 3) and the spaCy model (model N°2) (Figure 3 and 4):

```
Laptop Dell <B-BRAND> Inspiron X546
Black Pelikan <B-BRAND> Pencil 16mm
Battery Smart Energy by Energizer <B-BRAND>
Fijutsu <B-BRAND> DSLR Camera 156p
Genuine Paul <B-BRAND> Smith <I-BRAND> Men 's Belt-Leather Woven Plait Belt / BNWT / Sz : 36 '/ RRP : 110.00
Computer HP X80 Intel <B-BRAND> Xeon
Smart <B-BRAND> Watch <I-BRAND> Apple
Computer Big Hewelett-Packard <B-BRAND> Intel Xeon
24 Buttermilk oz Oroweat <B-BRAND> Bread ,
Black pencil Vertex <B-BRAND>
Wireless mouse MacTech <B-BRAND>
Smart Mouse Logitech <B-BRAND>
Brother <B-BRAND> Printer <I-BRAND> V167 Black Ink
```

Fig. 2. Some example titles not included in the dataset tested with the Flair BI-LSTM+CRF model

Laptop Dell BRAND Inspiron X546

Black Pelikan BRAND Pencil 16mm

Battery Smart Energy by Energizer BRAND

Fijutsu BRAND DSLR Camera 156p

Genuine Paul Smith BRAND Men's Belt-Leather Woven Plait Belt/BNWT/Sz: 36'/RRP:110.00

Laptop HP X80 Intel Xeon

Smart Watch BRAND Apple

Computer Big Hewelett-Packard BRAND Intel Xeon

24 Buttermilk oz Oroweat BRAND Bread,

Black pencil Vertex BRAND

Wireless mouse MacTech

Smart Mouse Logitech BRAND

Brother BRAND Printer Inkjet

Fig. 3. Some example titles not included in the dataset tested with the spaCy model (2nd model in Table 1)

A. *Discussion*

The Flair model, although trained for 40 epochs and has a larger size than the spaCy model can generalise better than the models trained with spaCy.

Alternatively, spaCy was faster to train and produced good models with a lower memory overhead. It is worth noting that both models (flair & spaCy) could detect unseen brands in the dataset such as Vertex, Mactech, etc.

What's interesting is that both models fail at tagging "Smart watch Apple" because in our case, there are very few examples of Apple (brand) products but there several food beverage products which contain the word Apple (the fruit) such as Apple juice or Apple pie, therefore, the model learned "Apple" as a fruit instead of a brand.

VI. IMPROVEMENTS

The models presented in this paper give satisfying results, however they can be improved, we believe that using and fine tuning a transformer model to generate the embeddings would be more beneficial and could result in a higher f1 score, for instance, it would solve the problem of detecting Apple as a fruit instead of a brand, since embeddings generated by transformer models are contextual so the embedding for Apple as a fruit and as a brand will be different.

A second way to improve the model accuracy is to get cleaner data, even though we did extensive data cleaning, there are some titles with different symbols, spelling errors, etc. Using a higher quality data set could improve the model performances.

Finally, the model can be supplemented with rule based algorithms that check if the brand already exist in the data set or in some database or both, there would be no need to use the model in this case unless the brand can't be found, in this case the model will be used to infer the brand. Figure 4 explains how this solution could work conceptually.

VII. RELATED WORK

Several works are related to using BI-LSTM + CRF in sequence tagging, [4] used several LSTM, BI-LSTM and BI-LSTM-CRF based models for sequence tagging and compared their performances. The problem of brand tagging in products has been tackled either using CRF [8], BI-LSTM CRF and deep word embeddings [7] combined with an LSTM neural

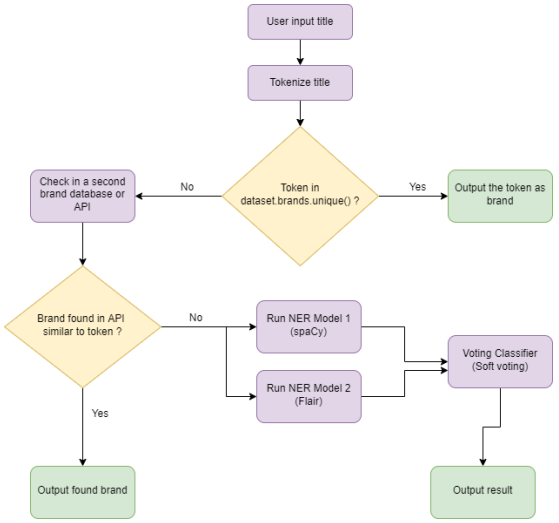

Fig. 4. First we search the brand in the brand column in the dataset, if we fail to find the brand, we search for it in a second API or database, if this fails, we run two models in parallel and give the results to a voting system that output the final result.

network used for classification. Furthermore, this task can be considered as a subtask of product-attribute extraction, [9] introduced several techniques such as BI-LSTM+CRF to tag different attributes in product profiles.

VIII. Conclusion

In this paper we suggested an end to end approach for brand tagging in product titles, from data curation to improving the final solution, we hope that this work will help others gain important insights in tagging brand names from unstructured text and could lead to developing better and more accurate techniques.

References

[1] Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, Roland Vollgraf, *FLAIR: An Easy-to-Use Framework for State-of-the-Art NLP*

[2] Piotr Bojanowski, Edouard Grave, Armand Joulin, Tomas Mikolov, Enriching Word Vectors with Subword Informations, 2015

[3] Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, Noah A. Smith *Transition-Based Dependency Parsing with Stack Long Short-Term Memory* 2015

[4] Zhiheng Huang, Wei Xu, Kai Yu, *Bidirectional LSTM-CRF Models for Sequence Tagging*, arXiv, 2015

[5] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013 *Distributed representations of words and phrases and their compositionality*. In Proc. NIPS

[6] Jianmo Ni, Jiacheng Li, Julian McAuley, *Justifying recommendations using distantly-labeled reviews and fined-grained aspects*, Empirical Methods in Natural Language Processing (EMNLP), 2019

[7] Andrey KulaginEmail authorYuriy GavrilinYaroslav Kholodov, *Deep Embeddings for Brand Detection in Product Titles*, AIST 2019

[8] Sen Wu, Zhanpeng Fang, Jie Tang,*Accurate Product Name Recognition from User Generated Content*, ICDM 2012 CPROD1

[9] Guineng Zheng, Subhabrata Mukherjee, Xin Luna Dong, Feifei Li, *OpenTag: Open Attribute Value Extraction from Product Profiles*, Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, London, UK, August 19-23, 2018

[10] spaCy, Industrial-Strength Natural Language Processing library (*https://github.com/explosion/spaCy*)