Attribute extraction from eCommerce product descriptions

Mikhail Sidorov
SUID: msidorov
Project category: Finance
Commerce/ NLP
msidorov@stanford.edu

Abstract

This project presents an implementation of named entity extraction for detecting attributes in the description of eCommerce products. This problem is very important for eCommerce search and catalog building systems. Effective named entity extraction could significantly improve quality of search results in eCommerce retail system and so the experience of customers. Because description of products is provided in plain text form without any structuring, this is also very challenging problem. Using as an example BestBuy eCommerce NER dataset we demonstrate the technology which includes feature extraction pipeline and trainig the model to recognize Brands, ModelNames, Price and other attributes from the product description. We provide a review of methods which are used for the information extraction. In our project we focused on three methids: SVM, Gradient Boosting Trees and Conditional Random Fields. Models we used were evaluated against the test set.

1 Introduction

Here we will determine some terms we will use in current report. We define *a product* as any commodity exposed by a retailer. Product contains set of attributes, where *attribute* is a named property of product which has some *attribute value* represented by one or several terms. The axmples of attributes are: Brand, Color, Proce, ModelName etc.

We define product desciption as a set of attributes with corresponding values.

As a short example:

```
apple watch series 2 grey
```

Has following attributes: Brand: apple; Category: watch; Color: grey; ModelName: series; ModelName: 2

Let us mark product as p, α_i as an anntribute and v_i it's value - the task will be to extract from text product description $p(\alpha_1 : v_1, ..., \alpha_m : v_m)$

Also we consider that each *product description* is represented as set of terms $(x_1, ..., x_n)$. We define our problem in the following way: for each attribute α_i we need to find a function E_{α_i} which will extract from product description attribute values v_i which belong to α_i. I.e.

$$v_i = (x_j | x_j \in E_{\alpha_i}(x_1, ..., x_n))$$

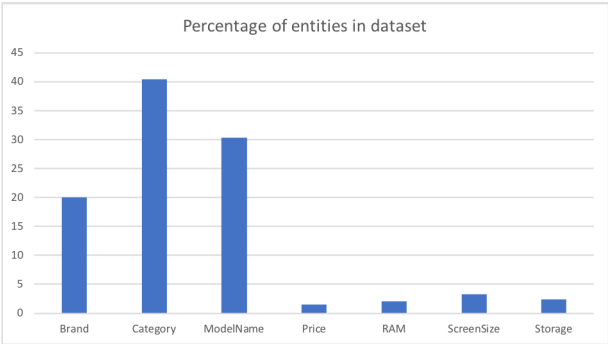
2 Data set description

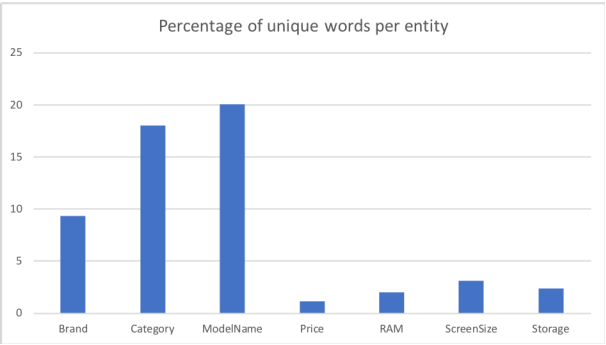
In our current project we use the data set provided:
`https://www.kaggle.com/datafurks/best-buy-ecommerce-ner-dataset/home`
and it's extension provided here:
`https://dataturks.com/projects/Dataturks/Demo%20Document%20Annotations`
Both data sets has the same format and we joined it and used it as one extended data set after deduplication. This joined data set has about 4000 records and 50% of these records are annotated (tagged) by experts. The structure of the annotated record is represented below. In this example we see that for the short description "Apple watch series 3 42mm from $339" expert annotated "Apple" as a Brand and "watch" as Category.


```
{"content": "Apple watch series 3 42mm from $339",
 "annotation":[
  {"label":["Brand"],"points":[{"start":0,"end":4,"text":"Apple"}]},
  {"label":["Category"],"points":[{"start":6,"end":10,"text":"watch"}]},
  {"label":["ModelName"],"points":[{"start":8,"end":14,"text":"series"}]},
  {"label":["ModelName"],"points":[{"start":19,"end":19,"text":"3"}]},
  {"label":["ScreenSize"],"points":[{"start":21,"end":24,"text":"42mm"}]},
  {"label":["None"],"points":[{"start":26,"end":29,"text":"from"}]},
  {"label":["Price"],"points":[{"start":31,"end":34,"text":"$339"}]}]
}
```


I.e. for annotated document the terms which expert marked as matched to some entity are mentioned in the annotation section of json.

Entities provided in the data set: Brand, Category, ModelName, ScreenSize, Storage, RAM. The frequency of the entities in the training set provided on graph below:





Because of the provided distribution on current phase we focused on the most frequent entities: Brand, Category and ModelName and train our algorithm to optimize metrics for these entities. Also we can see that, as we shopuld expect, Brand are represented by smaller subset of words, compare to Category and ModelName - it matched to the scenario when one company produce several caterories of product, and each category is represented by several models. So, we can expect different results of the extraction for these entities.

2.1 eCommerce data set for attribute extraction benchmark

We specially would like to mention a data set for eCommerce which has about 2 millions of tagged product descriptions which also contains images and which was created benchmark the task of the attribute extraction for eCommerce:

https://rloganiv.github.io/mae/

This data set contains annotated eCommerce descriptions as well as annotated images of products.

3 Metrics

For result esimation we use $Precission$, $Recall$ and F_1 metrics. The definition are:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F_1 = \frac{Precision \times Recall}{Precision + Recall}$$

We calculate per-entity metrics as well as total. Also we use accuracy to analyze classification of terms per entity.

4 Feature extraction pipeline

One of the important aspect of the project was arrangement of the feature extraction pipeline. We build feature extraction arround the concept of the extractor function. So, feature extraction pipeline is implemented as a collection of FeatureExtractors. Each feature extractor is a function which is applied to the term and check if it's possible to generate the feature value for this term. We follow the approach described in [1] and [5].

Feature extraction pipeline operates per product description, which is represented after normalization as $(x_1, ..., x_n)$.

Our pipeline could be represented as a function $f[(x_1, ..., x_n), position] \rightarrow (f_1, ..., f_D)$ which generate D dimensional feature vectir for term in $position = 1, .., n$. To descibe feature extraction we assume below $w_0 = x_{position}$ and index of w is a relative index to the $position$.

Below we provide a table with features we used with the short description. Note that we expect possibility of usage different set of features for the extraction of different entities.

So, in our machine learning model each term in product description is represented by the set of features, generated by feature extraction pipeline. One type of features could be specific for current term: for example is it numeric term or it consists only of letters, is term started from capital letter and length of term. Another type of features is contextual: value of feature depends on other terms in product description - bigramm is the simplest example.

Table 1: List of features for entity extraction

w_0	Represents term in current position
w_{-1}, w_0	Represents bigram with the previous term
w_0 is number	1 if w_0 iconsists only on digits otherwise 0
w_{-1} == and	1 if previous term is "and"
w_{-1} is uppercase	1 if previous term is uppercase
w_0 is uppercase	1 if current term is uppercase
i the position of w_0	Position of current term

5 Supervised classification approach

We use 2 classification methods for classification the entity for each token: SVM and Gradient Boosting Trees implemented scikit-learn package. Because we have a case of several classes (multinomial classification), we assign to the token an entity with the highest probability, if it

exceeded the threshold. We assign threshoild based on the ROC AUC curve which we build for SVM and GBT classifiers.

Also we tried Conditional random field to assigng labels for the tokens.

For SVM we found that the optimal recognition has been provided by SVM with RBF kernel and below we provid the results for different enttities. Paramaters γ and C for RBF were obtained via cross validation and are different for different entities.

Below we provided an example of multinomial classification for SVM classifier. Here Tag is the original Tag and TagPred is predicted Tag. For each category we explicitly provided the probability and the assigned category is calculated as category with maximum probability if it's exceed the threshold.

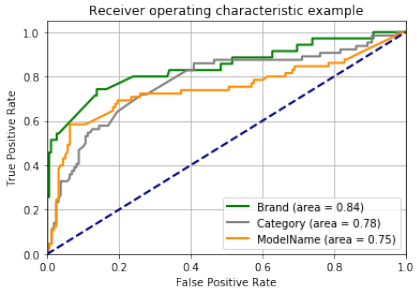
	TupleID	word	Tag	TagPred	Brand	Category	ModelName	None
0	1666	ipad	ModelName	ModelName	0.163666	9.191271e-02	0.779611	0.071780
1	1667	hp	Brand	Brand	0.776242	5.093781e-02	0.066650	0.071780
2	1667	27"	ScreenSize	None	0.021545	4.624442e-02	0.066650	0.161258
3	1667	ips	None	None	0.021545	1.535602e-01	0.041976	0.663453
4	1667	led	None	None	0.021545	1.298288e-01	0.041976	0.673207
5	1667	fhd	None	None	0.021545	1.298288e-01	0.041976	0.324960
6	1667	monitor	None	Category	0.021545	7.457812e-01	0.041976	0.182234
7	1668	apple	Brand	Brand	0.751999	9.191271e-02	0.066650	0.071780
8	1668	tv	Category	Category	0.021545	8.624945e-01	0.066650	0.088241
9	1669	asus	Brand	Brand	0.756061	9.191271e-02	0.066650	0.071780
10	1669	laptop	Category	Category	0.021545	8.033485e-01	0.066650	0.140245
11	1670	cell	Category	Category	0.163666	6.127541e-01	0.066650	0.071780

5.1 Classification results for SVM classifier

Below we provide results for SVM classifier (RBF kernel). We tried several other cores, but RBF is the optimal one. With cross-validation we defined the optimal parameters and provide final results for training and test set below.

Table 2: Entity recognition metrics for SVM (RBF kernel)

Training Set	Brand	Category	ModelName
Precision	0.900	0.900	0.930
Recall	0.750	0.800	0.780
F1-score	0.820	0.840	0.840
Dev Set	Brand	Category	ModelName
Precision	0.895	0.667	0.800
Recall	0.486	0.531	0.431
F1-score	0.630	0.591	0.560

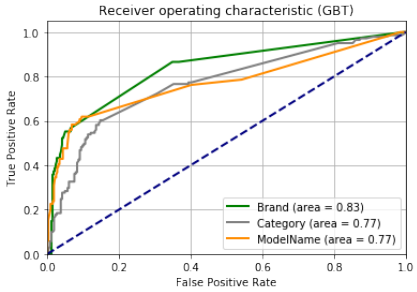


5.2 Classification results for GBT classifier

Next we used the same approach, but another classifier - Gradient Boosting Decision trees, also from scikit-learn package. Again, we determine parameters using cross-validation on training set and provide the results below for training and test set.

Table 3: Entity recognition metrics for GBT

Training Set	Brand	Category	ModelName
Precision	0.870	0.830	0.860
Recall	0.830	0.840	0.800
F1-score	0.850	0.840	0.830
Dev Set	**Brand**	**Category**	**ModelName**
Precision	0.793	0.603	0.727
Recall	0.697	0.620	0.480
F1-score	0.742	0.611	0.578



5.3 Conditional random fields

In CRF approach we estimate the probability of tags using:

$$p(y|x) = \frac{1}{Z(x)} \prod_{t=1}^{T} exp\left(\sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, x_t)\right)$$

where $\{f_k\}$ are feature functions, $\{\theta_k\}$ - parameters adjusted to model the observed statistics and$Z(x)$ is a normalization constant. . The most probable label sequnce y^* for input sequence x is:

$$y^* = argmax_y p(y|x)$$

Table 4: Conditional random field

Training Set	Brand	Category	ModelName
Precision	0.925	0.869	0.908
Recall	0.928	0.964	0.940
F1-score	0.927	0.914	0.924
Dev Set	**Brand**	**Category**	**ModelName**
Precision	0.818	0.621	0.631
Recall	0.614	0.711	0.746
F1-score	0.701	0.663	0.684

The same model we applied for attribute extraction of CoNLL2002 data set (`https://www.kaggle.com/nltkdata/conll-corpora`) as a base line (actually we use data set conll2002 provided by nltk) and got with the same approach the following results:

	precision	recall	f1-score	support
B-LOC	0.808	0.786	0.797	1084
I-LOC	0.685	0.643	0.663	325
B-MISC	0.735	0.563	0.638	339
I-MISC	0.709	0.594	0.646	557
B-ORG	0.813	0.834	0.823	1400
I-ORG	0.853	0.793	0.822	1104
B-PER	0.842	0.886	0.863	735
I-PER	0.887	0.942	0.914	634
avg / total	0.810	0.789	0.798	6178

For implementation we used CRF library `https://sklearn-crfsuite.readthedocs.io/en/latest/`

6 Discussion

In current project we applied supervised classification: SVM, GBT and Conditional random fields approaches to assign entities to the tokens from eCommerce product description. In general we can see based on $F1 - score$ that CRF demonstrated better results. We didn't use gazetteers as a source for additional features, but we assume that it will be very strong signal which can significantly improve the results.

Also we had very limited data set and we were not able to use other models like pretrained word embedding, which could compensate small data set, because we have ModelNames which are very specific and most likely are not a part of pretraing set like GloVe. Because our data set is relatively small, we were not able to solve the problem of overfitting, but even on this data set we can see good recognition of Brands.

It happened that SVM training demonstrated the longest computation time and CRF was the fastest one.

For future work we consider to apply approach described in [4] (Draft available on Stanford site: `https://web.stanford.edu/~jurafsky/slp3/17.pdf`) based on deep learning approach.

7 Code for the project

We used github as a code repository for the project `https://github.com/masidorov/cs229-project`

 All calculation were implemented on Amazon AWS SageMaker platform `https://aws.amazon.com/sagemaker/`.

References

[1] Ajinkya More (2016) Attribute Extraction from Product Titles in eCommerce, WalmartLabs, Sunnyvale CA 94089.

[2] Asif Ekbal Sivaji Bandyopadhyay (2010) Named Entity Recognition using Support Vector Machine: A Language Independent Approach. World Academy of Science, Engineering and Technology .

[3] Jesus Gimenez Llus Marquez (2004) Fast and Accurate Part of Speech Tagging: The SVM Approach Revisited. TALP Research Center, LSI Department, Universitat Politecnica de Catalunya Jordi Girona Salgado, Barcelona.

[4] Dan Jurafsky and James H. Martin (2019). Speech and Language Processing (3rd ed. draft). To be published.

[5] Marie-Francine Moens (2006). Information Extraction: Algorithms and Prospects in a Retrieval Context (The Information Retrieval Series).