A Study on Title Encoding Methods for e-Commerce Downstream Tasks

Cristian Cardellino and **Rafael Carrascosa**
Mercado Libre
cristian.cardellino@mercadolibre.com, rafael.carrascosa@mercadolibre.com

Abstract

In an e-Commerce marketplace there are usually many downstream tasks which have (relatively) less available resources than the few mainstream priority tasks, like recommendation or search. Examples of these tasks are product categorization, counterfeit detection, forbidden products detection, package size estimation, etc. Usually in these tasks the product titles are an appealing feature since they integrate key aspects of the product, are cheaply available and are easy to process. In this setting it makes sense to invest in a few high quality models that extract as much information as possible from the title and are shared among many downstream tasks. The present work explores the performance of different models to address different downstream tasks that are present in our marketplace. We also propose an adaptation of a deep network architecture from the Computer Vision field: "Bootstrap Your Own Latent" (BYOL), to learn product embeddings based on the title and compare it to several industrial baselines as well as some state-of-the-art supervised models. We found that although in some cases neural network based encoders can be very useful, in many scenarios the baselines given by shallower models are still hard to beat.

1 Introduction

The e-Commerce environment has grown at fast pace in recent years with new tasks and challenges to address. Some key tasks, like product search and recommendation, have large amounts of data available. However, some lesser known but still relevant tasks have relatively less resources available: smaller teams and annotated data. Examples are counterfeit product detection, package size estimation, product categorization, etc. In these scenarios, industrial applications started adopting organization-wide representations of business entities (e.g. customers, products, etc.) via mechanisms such as "Feature Stores" (Li et al. 2017).

Such representations tend to use features widely available across tasks and domains. In this work we explore the use of titles as the main feature for different supervised and self-supervised methods on 6 different downstream tasks from our marketplace. The product's title contains a lot of information in a limited space (e.g. brand, size, etc.).

We have 2 multiclass and 4 binary classification tasks from our marketplace. We compare several industry baselines as well as some state-of-the-art models, that can use the title as the main feature for each of the downstream tasks. In this study we explore FastText embeddings (Bojanowski et al. 2016) and FastText supervised classifier (Joulin et al. 2016), Meta-Prod2Vec (Vasile, Smirnova, and Conneau 2016) and a Bag-of-Words (with bigrams and trigrams) with a Logistic Regression classifier; we also explore BERT (Devlin et al. 2018) and a Text CNN (Kim 2014).

As part of our experiments we adapted "Bootstrap Your Own Latent" (BYOL) (Grill et al. 2020). In their work, the authors use a Siamese Network where one of the networks tries to predict the representation of the other using two augmentations of the same entity (in the case of BYOL, an image). In our work, the entity is an abstract product in which a user is interested. In this scenario we use pairs of products of the same browsing session of a user as the augmentation for BYOL. Our hypothesis is that products of the same sessions share similar properties useful in a transfer learning scenario in downstream tasks. We use this in two settings, as an encoder only later used with a linear classifier, and as a pre-trained encoder, finetuned for each task.

Our findings show that even though the state-of-the-art and deep models are usually the best in many tasks, the shallow and simpler models still are able to reach similar results with the added advantage of being cheaper to train, requiring less computational resources and even training data and easier to setup and maintain in a machine learning pipeline.

This paper is structured as follows: §2 presents other works in the area of product representation and also the works we take inspiration to design the encoder model. §3 describes all the components in our experimentation; the data, models, and architectures used for our experiments are thoroughly explained in this section. §4 showcases the results of our experimentation and discuss the findings and insights of it. Finally, in §5 we summarize our findings and delimit our line of future work.

2 Background

When working on e-Commerce, there are some tasks that usually receive more attention than others. Recommendation is perhaps the mainstream task in e-Commerce environments where there is a large body of work regarding the use of information such as the shopping or the browsing session of a user, like Prod2Vec (Grbovic et al. 2015) and Meta-

Prod2Vec (Vasile, Smirnova, and Conneau 2016). In particular, the use of metadata is also explored with the aid of deep learning models via neural networks (Hidasi et al. 2016; Zheng, Noroozi, and Yu 2017).

However, when it comes to other tasks, the body of work is smaller. The main examples are the use of embeddings inspired on Meta-Prod2Vec model for product categorization (Xu et al. 2021) or using complementary relationships in products for the same task (Xu et al. 2019). On the other hand, many of these tasks do not count with manually labeled data and most of them depend on user reports that are to be manually checked. Most importantly, in many cases, the metadata available might be limited. This is why we are interested in using a product's title as the main feature.

Recent years have seen a dramatic increase of latent representations, which are adequate to leverage information on the language itself to transfer knowledge to specific downstream tasks where the data availability is more limited. With the aid of architectures for training unsupervised language models (Merity, Keskar, and Socher 2017) or the attention mechanism (Vaswani et al. 2017), transfer learning has seen an explosion of applications in Natural Language Processing (Howard and Ruder 2018; Devlin et al. 2018; Radford et al. 2018).

Having this in mind we explored a novel architecture based on "Bootstrap Your Own Latent" (BYOL) by Grill et al. (Grill et al. 2020). In their work they propose a way to learn image embeddings for different downstream tasks that avoids using negative sampling (and thus avoid finding hard negatives) by learning over a Siamese network where only one of the heads is trained while the other is updated using a slow exponential moving average thus avoiding mode collapse. We adapted it to use products of the same browsing session as inputs.

3 Experimental Setup

3.1 Downstream tasks

We have 6 downstream tasks from our marketplace. Each task is comprised of a set of products and their labels, which depend on the task. Some tasks share the same set of products and only differ on the label. The main feature of each product, and the one that most of the evaluated methods use, is the title. The product's title contains a brief description of the product with some information such as brand, model, measures, etc. It is usually written with a narrowed vocabulary and simplified grammar. The same product can appear with variations of the title given by different vendors.

As the language of our marketplace is Spanish, the titles are normalized by stripping accents, removing stopwords and punctuation, and lowercasing the text. On the case of Meta-Prod2Vec only, and depending on the task, the product can have a unique universal ID (even if the product is sold by different vendors) and the category of the product.

Each task was split into train, test and validation subsets. The validation was used for hyperparameter tuning and the results reported in §4 are the ones of the test set.

In order to see the robustness of each of the models evaluated, we run experiments using different training sizes for each task: 1,000, 2,500, 5,000, 10,000, 15,000, 25,000 and the whole training dataset were used. We also run 5 experiments per each combination of task, training size and model, using different random samples and initialization of the data. We report the mean and standard deviation of our results.

In order to have a distribution of the data that better reflects the dynamics of the marketplace, the splits are calculated based on a specific date, this means that those products that are published before the date are part of the training set, and those that are after that date are part of the validation and test set (i.e. we have 2 split dates). This is because the products trends change over time and in some cases the downstream task distribution can change as well.

Product Categorization A multiclass classification task with the product category. Examples of categories are: "cars", "sneakers", "cellphones", etc. Usually this task is important for other downstream tasks because some of them use the label of the category as a feature in order to improve the results. Most of the times the label is given by the sellers when they publish the product. In some cases it is revised by a team, but in general term it is expected for a small proportion of the labels to be wrong, specially for very specific categories. The dataset consists of 120,000 products distributed in 986 categories. The distribution of the categories is exponential, where more than half of the products belong to one of the 100 most common categories. The dataset was filtered so that each of the categories has at least 10 products. This was mainly done for research purposes since we are not studying this downstream task only. There are 100,000 products for training, 10,000 for test and 10,000 for evaluation.

Product Identification A multiclass classification task where the label is a unique identifier in a catalog of products, regardless of the vendor that is selling it. Examples of a product are: "Galaxy S20", "Galaxy S20+", "iPhone 12", "iPhone 12 mini", etc. The task is needed in order to offer different options of the same product, by different vendors, in order to give the user a better experience when selecting. It is specially useful for very popular products such as phones, laptops and tablets. The labels of this data are carefully annotated since the catalog is used to group the products in order to show them to the buyer. It is a subset of the "Product Categorization" dataset since not all the products have a unique ID associated with them. It has 17,162 products with 1805 labels. It also has an exponential distribution, with half of the products belonging to the top 500 labels. Like with product categorization it was filtered so each label has at least 5 products associated with it. It has 10,000 elements for training, 3,581 for test and 3,581 for validation.

Counterfeit Product This is a binary classification task that tells whether the product is a counterfeit, i.e. a copy of an original product. An example are clothing products where the product is sold as part of a brand when this is not the case. The dataset is constructed based on users that report a product as being "fake". This is then validated by a human annotator. The classification task consists in checking if a report is valid or not (i.e. if the report is valid, then the product is a counterfeit). The dataset has a total of 66,908 products. The positive class is when the product is a counterfeit. There are 48% of products in the dataset for the positive class (i.e.

counterfeit product) and 52% for the negative class (i.e. original product). There are 40,144 products for training, 13,382 for test and the same amount for validation.

Forbidden Product This is a binary classification task. The objective is to detect which products are prohibited to commercialize in our marketplace (e.g. firearms). Similar to the case of counterfeit, the dataset is constructed based on the users that report a product and is validated by a human annotator. The classification task checks if the report is valid or not. The dataset has a total of 65,575 products, 22% of them corresponding to a valid report (i.e. a forbidden product) which is the positive class. There are 50,000 products for training, 7,733 for test and 7,842 for validation.

Automated Logistics This is a binary task that predicts whether or not a product can physically fit into a logistics automation system. In the case of our marketplace, a product is eligible for automated logistics if all the product dimensions (length, height and width) are less than 70cm long. The dataset has 69,946 products, with only 3% of the products not being able to fit in the logistics automation system. This is considered the "positive" class. There are 50,000 products for training, 9,950 for test and 9,996 for validation.

Free Shipping A binary classification task that classifies whether a product is eligible for free shipping in our logistics network. This depends mainly on both the product and the category. One of the most important features is the size of the product, in terms of dimensions and weight, since larger or heavier products usually need to pay a fee for transportation. There are other attributes as well that have an impact in the decision of charging the transportation, e.g. if the product is more expensive it usually brings the extra benefit of a free shipping. Some products, like batteries, need to pay a fee as an insurance. As it is the case for Automated Logistics dataset, this dataset is unbalanced as well. It has 69,078 products, with only 3% not having free shipping. Similar to the other case, the positive class represents the products non eligible for free shipping. There are 50,000 products for training, 10,057 for test and 9,021 for validation.

3.2 Users' Browsing Session Data

There are 3 methods that require an unsupervised pre-training step: MP2V, FastText embeddings and the "Bootstrap Your Own Latent" embeddings. In order to pre-train these models we use browsing sessions of users in our marketplace. These browsing sessions are defined by the products that a user "visits" (i.e. that it clicks to check the details) within a time window. Formally speaking, given a sequence of products $s = (p_1, ..., p_n)$ where p_i is a product and $T(p_i)$ is the timestamp the user accessed the product, we have that $T(p_{i+1}) - T(p_i) \leq \mathbf{T}$ for a short fixed time window $\mathbf{T}$ in minutes. The products of a browsing session consist of the title, the category and, in some cases, the product unique ID (see "Product Identification" task). The product titles were normalized like the ones in the downstream tasks. There are 7.6 million sessions using a time window T of 5 minutes and a total of 2,132,039 different products.

3.3 Machine Learning Models

We explore 4 shallow industry baselines, 2 deep learning models, and 2 versions of the model pre-trained based on the architecture of "Bootstrap Your Own Latent" (BYOL), a linear evaluation and a finetuned evaluation.

Bag-of-Words + Logistic Regression (BoW) The learning rate, L2 regularization parameter and the size of n-grams used in BoW were obtained via the validation data.

FastText Supervised It is the text classification tool, with the autotune parameter over the validation data, which runs a grid search during 5 minutes per experiment.

FastText Embeddings A FastText model trained with the titles of the products in the users' browsing session data (§3.2). For each task we use the trained model to encode the products' titles, and use these embeddings to train a logistic regression classifier similar to the BoW one.

Meta-Prod2Vec (MP2V) A model trained with the users' browsing session data (§3.2). The metadata available is the product ID, the category and the title. Depending on the task, the embeddings were trained with all the metadata, or part of it. For "Product Categorization", we removed the category and only train with the product ID and title, and for "Product Identification" the model was trained using only the category and the title, without the product ID. We use the trained model to encode the products of each task and use a Logistic Regression classifier with the embeddings, similar to FastText embeddings described before.

BERT We finetune BETO (Cañete et al. 2020), a BERT model pre-trained for Spanish. We use the Adam optimizer where the number of epochs, learning rate and regularization parameters were obtained with the aid of the validation data. The validation data is also used for early stopping the training of the model before it reaches the final epoch if there is no improvement after 5 epochs.

Text Convolutional Neural Network (CNN) It is used in combination with a word-piece tokenizer (Schuster and Nakajima 2012). The tokenizer, which has 32,768 subtokens, was trained on the products' titles from the users' browsing sessions data. The model was trained with an Adam optimizer and the parameters were set via validation similarly to BERT. This model is the base encoder for the BYOL model (§3.4). The architecture has 4 kernels of size 2, 3, 4, 5. Each kernel has 1024 filters and a global max pooling operation that is concatenated to a vector of size 4096 and then projected to a vector of dimension 256.

3.4 BYOL for Pre-training

We use "Bootstrap Your Own Latent" (BYOL) (Grill et al. 2020) as a pre-training task for the Text CNN encoder. The objective is to minimize the distance between encoded products' titles in the same session. Given an encoding function f_θ we use the browsing session data to calculate the parameters θ. After pre-training f_θ, we can add a linear layer for classification on each downstream task: Given the task data $S = \{(x_1, y_1), ..., (x_n, y_n)\}$, where x_i is a product and y_i is the label, we either train that layer for linear evaluation, or finetune the linear layer plus the encoder f_θ.

Figure 1 shows the model architecture. We switch the use of image augmentations with pairs of products in a session.

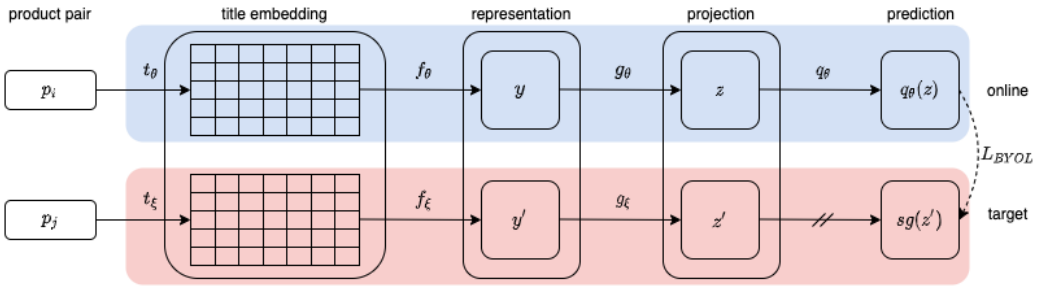

Figure 1: The model's architecture is based on BYOL (Grill et al. 2020). The model receives as input a pair of products p_i and p_j (represented by their titles). One product is projected through the *online* network until the projector value $q_\theta(z)$ is obtained. The other product is projected through the *target* network until the value z' is obtained. The model is trained on the weights θ by minimizing the objective function L_{BYOL}, which is the distance between q_θ and $sg(z')$ (where sg means stop-gradient operation).

The pairs are obtained combining all the pairs in each browsing session. The products are represented by the title. The encoder, f_θ, is the same Text CNN defined in §3.3.

The BYOL model consists on two paths: the *online* network and the *target* network. The online network is defined by a set of weights θ and is comprised of 4 stages: an *embedding lookup* t_θ, a *representation encoder* f_θ, a *projector* g_θ, and a *predictor* q_θ. The target network has a very similar architecture to the online network, except that it doesn't have the predictor q_θ. It also has a different set of weights ξ. The weights of the target network are obtained by a slow exponential moving average of the online parameters θ, given a target decay rate $\tau \in [0, 1]$. After each training step the weights ξ are updated: $\xi \leftarrow \tau\xi + (1 - \tau)\theta$. Both θ and ξ are initialized to the same values when creating the networks. After training the model, we are interested in the pre-trained encoder $f_\theta \circ t_\theta$. The rest of the model is discarded.

BYOL trains the online network by minimizing the distance between the prediction q_θ and the projection z' of the target network. The stop-gradient *sg* is important to avoid optimization over the weights ξ which would cause mode collapse (Chen and He 2020). The online network's *prediction* $q_\theta(z)$ is compared to the target network's *projection* z' and the error is the L_2-normalized loss between the vectors:

$$L_{BYOL} = 2 - 2 \cdot \frac{\langle q_\theta(z), z' \rangle}{\|q_\theta(z)\|_2 \cdot \|z'\|_2} \quad (1)$$

Equation 1 only refers to one way of the product pair p_i, p_j. The symmetric loss is obtained by swapping the inputs of the network. The pre-training was done using 315 million pairs that were obtained from the 7.6 million users' browsing sessions. The architecture of the encoder is the same as the Text CNN model. The projection encoder g_θ is a *multi-layer perceptron* that takes the 256 dimensions and expands it to a hidden layer of size 1024, followed by batch normalization, rectified linear unit and a final linear layer of size 1024. The predictor q_θ has this same architecture. The model was pre-trained with the parameters of Grill et al.

We use the encoder in two settings: linear evaluation and fine-tuning of data. In the former, BYOL is an encoder only, we train a Logistic Regression classifier in the same way it was done for FastText Embeddings and MP2V. For the latter, we fine-tune the model training it like for Text CNN.

4 Results and Discussion

Table 1 shows the results of our experiments. Each horizontal section of the table corresponds to a task, with the task name and metric reported in the first column. The second column shows the names of the models. There are 7 columns corresponding to the different sizes of training sample. The last column represents training with all the training data for that task (§3.1). For the case of Product Identification, the maximum training data available was 10,000 (represented in the "All Data" column for the task). For the multiclass classification tasks (Categorization and Identification) we report the *F1-Score Macro Average* over all the classes. For the binary classification tasks we report the *Average Precision Score*, which is an approximation of the area under the precision recall curve.

The table shows us that: 1) BERT has the best performance overall across different tasks, leading 3 of the 4 binary tasks when all data is used; 2) Bag-of-Words (BoW) has a surprising performance being a close top-3 in all tasks, regardless of training size; 3) BYOL performance's excels in the tasks of product categorization and identification; 4) MP2V relies on the metadata, particularly the category, and when it is not available, e.g. in the categorization task, the method collapses; the extra information is not enough to boost its performance over other methods, except for very specific scenarios; 5) FastText supervised is somewhat irregular in its performance since for some experiments it collapses; 6) this is not the case for FastText embeddings which show an overall good and uniform performance across tasks; 7) the pre-training of the Text CNN encoder via BYOL is clearly superior over training it from scratch.

The performance of BYOL in the first two tasks (Categorization and Identification) hints that pairs of products seen by a user in the same browsing session are either the same product or a slight variation of the product within the same

Task	Train Size / Model	1000	2500	5000	10000	15000	25000	All Data
Categorization	BoW	9.8±0.26	17.5±0.75	25.7±0.55	35.4±0.66	41.8±0.41	49.5±0.46	66.7±0.0
	FastText Sup	3.0±1.02	3.0±1.34	3.1±1.58	3.5±1.56	3.8±1.77	1.3±0.04	1.5±0.0
F1-Score	FastText Emb	12.8±0.5	20.7±0.59	28.6±0.17	37.8±0.59	43.8±0.5	51.0±0.81	64.8±0.0
Macro	MP2V	8.2±4.61	3.3±7.38	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
Average	BYOL Emb	**15.0±1.43**	**25.6±0.87**	**36.0±0.56**	**46.0±0.63**	**50.9±0.86**	**56.7±1.28**	**68.2±0.08**
	BYOL	12.5±0.68	22.6±0.68	33.6±0.86	45.3±1.07	48.9±3.24	52.6±2.65	63.9±0.8
	BERT	13.0±0.71	24.4±1.19	32.0±1.88	40.9±4.02	46.4±2.27	52.8±1.21	65.4±0.65
	Text CNN	6.1±0.35	11.2±0.46	11.5±5.31	18.9±2.37	23.6±3.23	31.9±1.88	58.8±1.48
Identification	BoW	18.2±0.35	39.9±0.5	59.5±0.61				76.8±0.0
	FastText Sup	6.7±2.45	11.1±2.23	15.1±3.75				23.0±1.07
F1-Score	FastText Emb	16.8±0.52	36.0±0.29	54.4±0.7				73.5±0.0
Macro	MP2V	11.3±0.33	25.1±0.24	32.6±9.47				40.5±2.21
Average	BYOL Emb	2.8±1.13	38.7±0.56	57.6±0.97				74.3±0.19
	BYOL	**19.5±0.49**	**43.1±0.68**	**64.5±0.5**				**80.4±0.34**
	BERT	7.0±9.44	7.8±17.53	12.2±27.28				15.7±35.11
	Text CNN	17.3±0.16	26.4±14.6	51.6±1.61				66.5±0.11
Counterfeit	BoW	88.0±0.4	90.1±0.22	91.6±0.12	93.1±0.15	**93.9±0.11**	95.1±0.11	96.1±0.0
	FastText Sup	85.0±3.81	89.9±0.14	91.3±0.18	92.7±0.15	93.6±0.09	94.9±0.21	96.4±0.04
Average	FastText Emb	86.3±0.22	87.4±0.13	87.9±0.19	88.4±0.1	88.5±0.05	88.7±0.04	88.8±0.0
Precision	MP2V	84.3±0.48	85.4±0.22	86.2±0.18	86.7±0.17	86.8±0.1	87.0±0.09	87.0±0.0
Score	BYOL Emb	81.9±0.97	87.9±0.37	89.3±0.05	89.9±0.05	90.1±0.1	90.3±0.07	90.4±0.0
	BYOL	**88.5±0.32**	**90.9±0.16**	**91.8±0.21**	92.8±0.17	93.6±0.23	94.3±0.17	95.4±0.09
	BERT	88.1±0.88	89.6±1.44	91.2±0.47	**93.2±0.15**	93.9±0.34	**95.3±0.17**	**96.5±0.0**
	Text CNN	87.2±0.46	88.9±0.79	90.7±0.46	91.8±0.23	93.2±0.73	94.0±0.28	95.2±0.04
Forbidden	BoW	27.3±1.45	32.7±0.61	36.7±1.12	39.1±1.46	40.6±1.23	42.8±1.31	45.4±0.0
	FastText Sup	10.7±2.1	22.9±10.39	30.2±9.72	27.3±13.26	28.4±16.23	24.1±16.58	10.7±1.16
Average	FastText Emb	34.2±1.12	36.3±1.58	38.5±0.82	39.6±0.71	39.6±0.53	40.0±0.22	40.1±0.0
Precision	MP2V	27.0±0.72	30.9±1.11	30.9±0.53	29.1±1.76	26.6±7.3	30.7±1.56	30.8±0.0
Score	BYOL Emb	29.0±1.38	33.2±0.36	34.6±0.65	35.2±0.46	35.7±0.39	36.0±0.35	36.0±0.0
	BYOL	**34.7±0.82**	38.1±1.14	41.3±0.94	42.2±1.35	43.9±1.47	47.0±2.55	50.9±0.08
	BERT	33.6±2.4	**39.7±1.64**	**43.3±1.25**	**47.7±1.59**	**50.3±1.3**	**52.5±1.61**	**51.2±0.0**
	Text CNN	27.7±3.38	32.3±2.58	36.7±1.48	40.0±1.3	38.5±1.27	43.1±0.75	46.4±0.0
Logistics	BoW	5.8±1.05	8.0±1.47	11.5±2.4	15.4±3.22	**18.9±1.6**	23.2±1.8	27.1±0.0
	FastText Sup	3.1±0.13	3.2±0.04	6.8±3.4	10.8±5.14	12.6±5.83	21.0±2.1	27.1±0.5
Average	FastText Emb	7.4±0.92	8.8±0.29	11.4±0.91	13.9±0.43	16.3±0.33	17.8±0.54	19.8±0.0
Precision	MP2V	**8.4±1.57**	**11.5±1.26**	**14.1±1.11**	**15.6±0.77**	17.1±0.62	18.1±0.77	19.0±0.0
Score	BYOL Emb	5.0±0.74	6.7±0.88	9.4±0.52	11.8±1.0	12.7±0.83	13.9±0.42	14.8±0.0
	BYOL	3.7±0.11	7.9±1.49	10.5±2.25	13.9±3.67	16.5±1.04	17.8±1.77	25.7±1.21
	BERT	5.0±0.77	8.7±1.56	11.3±1.96	14.6±3.35	17.7±1.09	**23.5±1.82**	**28.2±0.0**
	Text CNN	4.6±0.72	5.6±0.81	7.1±0.96	8.4±1.57	9.2±0.79	12.0±1.88	15.3±0.49
Shipping	BoW	11.3±2.61	21.4±2.04	30.4±2.36	**37.5±2.04**	**44.6±1.99**	**51.8±1.84**	**61.9±0.0**
	FastText Sup	3.5±0.24	11.2±7.25	16.1±11.13	26.5±12.4	38.6±2.85	43.2±3.72	53.5±1.59
Average	FastText Emb	15.5±3.66	22.0±2.86	26.3±1.84	29.9±0.89	32.4±1.19	35.3±0.65	37.3±0.0
Precision	MP2V	**18.0±4.86**	**27.5±1.42**	**32.0±2.54**	34.2±1.34	35.9±1.14	36.6±0.94	37.6±0.0
Score	BYOL Emb	6.8±1.1	8.0±1.77	16.4±2.09	21.2±2.45	24.0±2.13	26.5±1.32	28.5±0.0
	BYOL	3.1±0.26	17.5±2.97	25.2±3.95	33.7±2.71	39.4±2.4	45.1±2.25	56.9±1.89
	BERT	7.9±1.79	16.2±1.98	25.8±3.34	36.3±4.82	42.7±1.95	51.4±2.13	60.5±0.0
	Text CNN	4.9±0.92	8.9±1.2	14.6±2.62	20.8±3.34	25.0±4.38	32.0±1.84	40.0±0.0

Table 1: Experiments Results: The table is divided horizontally and vertically by lines. Each group between horizontal lines represents a group of experiments for a task. Each row within that group has the results of each model. The first column has the name of the task on top and the metric used to evaluate that task in the middle. The second column has the name of the model that corresponds to the results in that row. Each column with a number on top has the results for that training sample.

category. This also indicates that the chosen "augmentation" is good for this type of task, but not necessarily good for others. The finetuned evaluation still lags behind BERT. These limitations could be addressed with other kinds of augmentations or larger encoders with attention mechanism.

The results show that BoW with Logistic Regression is an excellent candidate in a text classification setup, much better than FastText supervised model which showed extremely poor performance in some scenarios, even using its autotune mechanics. BoW rivals BERT's performance in some cases. FastText embeddings with logistic regression show good results (and more consistency than the supervised version), although not as good as the ones shown by BoW. The problem with BoW is the sparse nature of the encoding making it much harder to be complemented with other features.

There are only 2 scenarios where the size of training data has impact on the performance of a model making it much better than the others: logistics and shipping with small

amount of training data. In these cases MP2V is better. This hints that MP2V embeddings better captures the product's size with limited data.

5 Conclusions

This study explores different ways to encode a title and use it for downstream tasks. We evaluate several models for many text classification tasks. In addition, we adapted an architecture for pre-training an encoder for downstream tasks that appear in the daily challenges of an e-commerce scenario, based on the work by Grill et al. (Grill et al. 2020).

The proposed model, BYOL, shows very good results for two of the tasks, which are related, and this is a clear consequence of the selected method for "augmentation". This opens the possibility for future research on what type of augmentations are worth exploring for certain types of tasks. This could also be explored by adding extra parameters in the objective function, e.g. the downstream tasks objectives, making it a multi-task learning model. Also, the comparison with a Text CNN trained from scratch showed us that the BYOL architecture is feasible for pre-training.

Depending on the task, different models can have different advantages. Even if BERT was the leading model in most of the tasks, it comes at a cost of pre-training, finetuning and even inference, that none of the other models have. BoW with logistic regression is a hard to beat baseline that we should always check, and the use of FastText embeddings for representation shows very good and balanced results. Meta-Prod2Vec can vary greatly depending on the task, and may need a non linear model for better performance. While BYOL, with the correct choice for augmentation and encoder can be a good alternative.

References

Bojanowski, P.; Grave, E.; Joulin, A.; and Mikolov, T. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Cañete, J.; Chaperon, G.; Fuentes, R.; and Pérez, J. 2020. Spanish pre-trained bert model and evaluation data. In *to appear in PML4DC at ICLR 2020*.

Chen, X., and He, K. 2020. Exploring simple siamese representation learning.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Grbovic, M.; Radosavljevic, V.; Djuric, N.; Bhamidipati, N.; Savla, J.; Bhagwan, V.; and Sharp, D. 2015. E-commerce in your inbox: Product recommendations at scale. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, 1809–1818. New York, NY, USA: Association for Computing Machinery.

Grill, J.-B.; Strub, F.; Altché, F.; Tallec, C.; Richemond, P. H.; Buchatskaya, E.; Doersch, C.; Pires, B. A.; Guo, Z. D.; Azar, M. G.; Piot, B.; Kavukcuoglu, K.; Munos, R.; and Valko, M. 2020. Bootstrap your own latent: A new approach to self-supervised learning.

Hidasi, B.; Quadrana, M.; Karatzoglou, A.; and Tikk, D. 2016. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, 241–248. New York, NY, USA: Association for Computing Machinery.

Howard, J., and Ruder, S. 2018. Universal language model fine-tuning for text classification.

Joulin, A.; Grave, E.; Bojanowski, P.; and Mikolov, T. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Kim, Y. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1746–1751. Doha, Qatar: Association for Computational Linguistics.

Li, L. E.; Chen, E.; Hermann, J.; Zhang, P.; and Wang, L. 2017. Scaling machine learning as a service. volume 67 of *Proceedings of Machine Learning Research*, 14–29. Microsoft NERD, Boston, USA: PMLR.

Merity, S.; Keskar, N. S.; and Socher, R. 2017. Regularizing and optimizing lstm language models.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2018. Language models are unsupervised multitask learners.

Schuster, M., and Nakajima, K. 2012. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5149–5152.

Vasile, F.; Smirnova, E.; and Conneau, A. 2016. Meta-prod2vec: Product embeddings using side-information for recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, 225–232. New York, NY, USA: Association for Computing Machinery.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is all you need. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc. 5998–6008.

Xu, D.; Ruan, C.; Körpeoglu, E.; Kumar, S.; and Achan, K. 2019. Modeling complementary products and customer preferences with context knowledge for online recommendation. *CoRR* abs/1904.12574.

Xu, D.; Ruan, C.; Korpeoglu, E.; Kumar, S.; and Achan, K. 2021. Theoretical understandings of product embedding for e-commerce machine learning. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, WSDM '21, 256–264. New York, NY, USA: Association for Computing Machinery.

Zheng, L.; Noroozi, V.; and Yu, P. S. 2017. Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, WSDM '17, 425–434. New York, NY, USA: Association for Computing Machinery.