e-Commerce product classification: our participation at cDiscount 2015 challenge

Ioannis Partalas
Viseo R&D, France
ioannis.partalas@viseo.com

Georgios Balikas
University of Grenoble Alpes, France
georgios.balikas@imag.fr

Abstract

This report describes our participation in the cDiscount 2015 challenge where the goal was to classify product items in a predefined taxonomy of products. Our best submission yielded an accuracy score of 64.20% in the private part of the leaderboard and we were ranked 10th out of 175 participating teams. We followed a text classification approach employing mainly linear models. The final solution was a weighted voting system which combined a variety of trained models.

1 Introduction

In this report we present our participation in the cDicount 2015 product classification challenge that was organised in the www.datascience.net platform. The organisers provided a large collection of product items containing mainly its textual description. We followed a text classification approach using mainly linear models (like SVMs) as our base models. Our final solution consisted of a weighted voting system which combined a variety of base models. In Section 2 we provide a brief description of the cDiscount task and data. Section 3 describes our implementations and the results we obtained. Finally, Section 4 concludes with a discussion and the lessons learnt.

2 e-Commerce Product Classification

Item categorization is fundamental to many aspects of an item life cycle for e-commerce sites such as search, recommendation, catalog building etc. It can be formulated as a supervised classification problem where the categories are the target classes and the features are the words composing some textual description of the items. In the context of the cDiscount 2015 challenge[1] the organisers provided the descriptions of e-commerce items and the goal was to develop a system that would perform automatic classification of those products. Table 1 presents some training instances. There were 15,786,886 product descriptions in the training test and 35,066 instances to be classified in the test set. The target categories (classes) were organised in a hierarchy that comprised 3 levels: the top level had 52 nodes, the middle 536 and the lowest level 5,789. The goal of the challenge was to predict the lowest level category for each test instance. It is to be noted that most of the classes were represented in the training set with only a few examples and there were only a few classes with many examples. For instance 40% of the training instances belong to the 10 most common classes and around 1,500 classes contain from 1 to 30 product items.

3 Our approach

We followed a text classification approach working with the textual information provided in the training data. In this context $x \in \mathbb{R}^d$ represents a document in a vector space and $y \in \mathcal{Y} = \{1 \dots K\}$ its associated class label where $|\mathcal{Y}| > 2$. The large number of classes in the problem treated as well as the scarcity of data for minority classes is a typical situation in large-scale systems. For example, we cite here challenges in the same line where in contrast the available text is much bigger like LSHTC (Partalas et al., 2015) and BioASQ (Balikas et al., 2014).

[1] https://www.datascience.net/fr/challenge/20/details#tab_brief55

Categorie3	Description	Libelle	Marque
1000015309	De Collectif aux éditions SOLESMES	Benedictions de l eglise	
1000015309	De Collectif aux éditions SOLESMES	Notice de st benoit lot de 10	
1000010100	or 750, poids : 3.45gr, diamants : 0.26carats	Bague or et diamants	AUCUNE
1000003407	Champagne Brut - Champagne-Vendu à l'unité-1 x 75cl	Mumm Brut	AUCUNE

Table 1: Part of the training data. We only present the fields "Description", "Libelle" and "Marque" that we used in our implementations. The "Categorie3" value was the class to be predicted.

3.1 Setup

Data Pre-processing. Since we used only the textual information that was provided for each training/test instance by the organisers our first task was to clean the data. The pipeline for cleaning the data included: removal of non-ascii characters, removal of non-printable characters, removal of html tags, accents removal, punctuation removal and lower-casing. We also split words that consisted of a text part and a numerical part in to two distinct parts. For instance, "12cm" would become "12" and "cm". In addition, we did not perform stemming, lemmatization and stop-word removal due to the fact that the text spans were small and such operations would result in loss of information (Shen et al., 2012). Finally, we tokenized the remaining text in words using the white space as delimiter.

Vectorization. Having cleaned the text, we generated the vectors using an one-hot-encoding approach. We experimented with binary one-hot-encoded vectors (i.e. a word exists or not), with term frequency vectors (how many times each word occurred in the instance) and with the $tf-idf$ (term frequency, inverse document frequency) weighting scheme. In the early stages of the challenge, we found that the latter performed the best in our experiments and we used it exclusively. To calculate $tf-idf$ vectors we smoothed idf and applied the sublinear tf scaling 1+log(tf). Finally, each vector has been normalized to a unit vector.

Classification of short documents can benefit from successful feature selection of n-grams. The size of the vocabulary of our cleaned dataset combined with the large number of training instances was prohibitively large for using all the n-grams with $n = 1, 2, 3$. On the other hand, selecting too many features may lead to over-fitting. As a result, selecting a representative part of those n-grams required careful tuning.

Apart from the "Description" and "Libelle" fields that we concatenated, we also used the "Marque" field. We examined two ways of integrating this information in our pipeline: by concatenating its value with the already existing text of "Description" and "Libelle" and by generating binary flags for each of the values of the field seen on the training set. Either way, compared to feature generation using only "Description" and "Libelle" benefited our models.

After generating the vectors we applied the α-power normalization, where each vector $x = (x_1, x_2, \dots, x_d)$ is transformed to $x^{power} = (x_1^\alpha, x_2^\alpha, \dots, x_d^\alpha)$. This normalisation has been used in computer vision tasks (Jegou et al., 2012). The main intuition is that it reduces the effect of the most common words. After the transformation the vector is normalized to the unit vector. We found that taking the root ($a = 0.5$) of the values of the features consistently benefited the performance.

Subsampling As the training dataset was highly imbalanced the learned models would be biased towards the big classes. For this reason and also in order to speed up the vectorization process as well as the training of the systems we randomly sampled the data by downsampling the majority classes. This procedure helped to improve the performance of all the single systems (around +2.5% to our best single system) and also reduced the training time of the base models. The size of the vocabularies for the several sub-samples ranged from around 1 million unigrams for around half of the data to 1.6 millions of unigrams for the full dataset.

Tuning and Validation Strategy Tuning the hyper-parameters of our models was an important aspect of our approach. Performing k-fold cross validation in 15 million training instances with thousands of features (corresponding to the vocabulary size of the training set) was prohibitive given our cpu resources. We overcame this problem by performing hyper-parameter tuning locally, in a subsample of the training set. The subsample consisted of the instances of 1500 classes randomly selected. This approach allowed us to accelerate the calculations. Note that we also validated the applicability of our tuning strategy using the public part of the leaderboard; the decisions that improved the accuracy of our models locally had the same effect on our scores on the public leaderboard. A trick that we found useful was to use the current best submission on the public board as golden standard in order to validate the trained models. This helped us to avoid unnecessary submissions.

3.2 Training and Prediction

We relied on linear models as our base learning choice due to their efficiency on high-dimensional tasks like text classification. Support Vector Machines (SVMs) are well known for achieving state-of-the-art performance in such tasks. For learning the base models we used the Liblinear library which can support linear models learning in high dimensional datasets (Fan et al., 2008).

We tried two main strategies: a) flat classifiers which ignore the hierarchical structure among the classes and b) hierarchical top-down classification. For flat classification we followed the One-Versus-All (OVA) approach with a complexity $O(K)$ in the number of classes. For top-down classification we trained a multi-class classifier for each parent in the hierarchy of products and during prediction we started from the root and selected the best class according to the current multi-class classifiers. We iteratively proceeded until reaching a leaf node. Note that top-down hierarchical classification has a logarithmic complexity to the number of classes which accelerates the training and prediction processes significantly.

Trying to explore different learning methods that would also help to diversify the ensemble, we experimented with k-Nearest Neighbors classifiers, Rochio classifiers also known as Nearest-Centroid classifiers and online stochastic gradient-descent methods. Although widely studied we found that those methods did not give satisfactory results. For instance, our 3-Nearest Neighbors runs using 200 thousands unigram features, $tf - idf$ feature representation achieved accuracy score 41.62% in the public leaderboard. We also experimented with text embeddings using the word2vec tool (Mikolov et al., 2013); generating text representations in a low dimensional space (200 dimensions) and 3-NN as a category prediction approach improved over using the $tf - idf$ representation but was still away from performing competitively. For the above mentioned reasons we only report results in the rest of the report for SVMs. In the majority of the models we used L_2-regularized L_2-loss SVMs in the dual and we set $C = 1.0$ with bias ($-B$ 1).

3.3 Ensembling

Our final solutions were based on averaging the models in the ensemble which contained 103 models. We experimented with simple voting as well as with weighted voting schemes. Simple voting was always improving performance when it was calculated in a fraction of the whole ensemble containing only the best performing models. A simple approach was to order the models according to their performance (in terms of accuracy) with respect to the current best model on the leader board. Then a simple majority voting was applied on around 20%-30% of the ordered classifiers. This procedure would create a homogeneous sub-ensemble that would likely reduce the variance.

Also, we employed a weighted voting scheme weighting more few of our best single models. Our final best submission (64.20% on the private board) was a weighting voting ensemble giving bigger weights to the two best models. Weighted voting consistently improved accuracy about 1.2%-1.8%.

3.4 Software and Hardware

During the challenge we used scikit-learn (Pedregosa et al., 2011) as well as our own scripts to pre-process the raw data. For training the models we experimented with Liblinear, scikit-learn and Vowpal Wabbit. We had full access to a machine with 4 cores at 2,4Ghz and 16Gb of RAM and limited access to a shared machine with 24 cores at 3,3Ghz and 128 Gb of RAM. In the first machine we ran our experiments with up to 270 thousand features and in the second the experiments with more features that required more memory.

4 Results

We provide in Table 2 a subset of our submissions along with the public and private scores we obtained. Comparing the pairs of submissions (1),(2) and (9), (10) it is clear that the α-power transformation helps the performance with respect to accuracy. From submissions (1), (3) and (4) one can see that by increasing the number of the unigram features the performance increases. However, at around 300 thousand features the improvements becomes negligible. The pairs of submissions (3) , (5) and (2), (6) demonstrate the advantage of adding bigrams apart from unigrams to the feature set. Going further and adding, apart from unigrams and bigrams, trigrams also improves the performance as indicated by comparing submissions (8), (9) and (10) with the rest of the Table. Note that in each of the above cases the features were selected with criterion their frequency. For instance in submission (1) we selected the 200k most common unigrams, and in submission (9) the most common features between all unigrams, bigrams and trigrams.

We tested also several hierarchical models using 200,000 unigrams for each parent node in the hierarchy. The best submission was one with the first level pruned achieving 60.61% on the private board while the fully hierarchical model got only 58.88%. Note that in the case of the pruned hierarchy we remove a step during prediction and thus reduce the propagation errors. Several other hierarchical models were used to increase the variability of the ensemble.

Table 3 presents the best single system which was trained in approximately half of the data where the “Marque” was concatenated directly in the description. Also we present the results on the two best weighted voting systems using a total of 103 base models. Additionally, we report the coverage of each system which shows how many classes were detected during the prediction phase.

5 Conclusion and Discussion

Participating in the cDiscount 2015 was a nice and interesting experience with several lessons learnt. We shortly discuss below the most important ones:

- Data Preparation. Although short text classification for e-Commerce product is not a new task the cDiscount problems had two particularities: the huge number of training instances and the big vocabulary even after carefully cleaning the dataset. We found that feature selection and feature engineering performed by studying the statistics of the dataset, the predictions of base models and by trying to identify patterns for the classes played an important role. As an example we would like to highlight the “Marque” field. We found in the late stages of the challenge that by using this information our models could benefit significantly.

- Learning tools. From the early stages of the challenge we decided to use SVMs which are known to perform well in such problems. It it the case, however, that the more tools one can use efficiently the better since there is no strategy that works in every setting. Unfortunately, we did not perform an exhaustive hyper-tuning of Vowpal Wabbit on-line learning system which had the potential to provide a high performing system.

- Ensemble methods. Base models such as SVMs can obtain satisfactory performance in classification tasks. In the framework of a challenge, however, ensemble methods have the potential of deciding the winner. Ensemble methods (stacking, feature weighted linear stacking etc.) with a pool of strong and diversified models can improve performance by several units of accuracy. Here we relied on weighted voting processes but we believe that using more sophisticated techniques would have helped us.

Acknowledgements

We would like to thank the AMA team of the University of Grenoble-Alpes for providing us the machines where we ran our algorithms.

References

[Balikas et al.2014] George Balikas, Ioannis Partalas, Axel-Cyrille Ngonga Ngomo, Anastasia Krithara,

	Description	Public	Private
1	200K unigrams	61.09	60.77
2	200K unigrams, α=0.5	61.60	61.25
3	250K unigrams	61.142	60.77
4	300K unigrams	61.148	60.87
5	250K unigrams, 250K bigrams	61.79	61.37
6	200K unigrams, 400K bigrams, α=0.5	62.09	61.76
7	200K unigrams, 400K bigrams, α=0.5, "Marque" as binary feature	62.64	62.15
8	1,2 M unigrams, bigrams, trigrams, α=0.5	62.28	61.99
9	2M unigrams, bigrams, trigrams	62.35	61.83
10	2M unigrams, bigrams, trigrams, α=0.5	63.30	62.99

Table 2: A subset of our base model submissions with their scores in the public and private leaderboard

	Description	Public	Private	Coverage
1	270K unigrams, $\alpha = 0.5$, half data	63.56	63.11	3,208
2	Weighted voting 1	64.55	64.20	3,128
3	Weighted voting 2	64.57	64.14	3,116

Table 3: Our best submissions with their scores in the public and private leaderboard

Eric Gaussier, and George Paliouras. 2014. Results of the bioasq track of the question answering lab at clef 2014. *Results of the BioASQ Track of the Question Answering Lab at CLEF*, 2014:1181–93.

[Fan et al.2008] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874.

[Jegou et al.2012] H. Jegou, F. Perronnin, M. Douze, J. Sanchez, P. Perez, and C. Schmid. 2012. Aggregating local image descriptors into compact codes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(9):1704–1716, Sept.

[Mikolov et al.2013] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[Partalas et al.2015] Ioannis Partalas, Aris Kosmopoulos, Nicolas Baskiotis, Thierry Artieres, George Paliouras, Eric Gaussier, Ion Androutsopoulos, Massih-Reza Amini, and Patrick Galinari. 2015. Lshtc: A benchmark for large-scale text classification. *arXiv preprint arXiv:1503.08581*.

[Pedregosa et al.2011] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[Shen et al.2012] Dan Shen, Jean-David Ruvini, and Badrul Sarwar. 2012. Large-scale item categorization for e-commerce. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 595–604. ACM.