CIKM AnalytiCup 2017 – Lazada Product Title Quality Challenge: Deep and wide learning for clarity and conciseness classification

Thanh Lam Pham
Data Science Group, Sentifi
Ho Chi Minh City, Vietnam
lam.pham@sentifi.com

Quang Hieu Vu
Data Science Group, Zalora
Ho Chi Minh City, Vietnam
quanghieu.vu@zalora.com

ABSTRACT

This paper introduces an ensemble model which combines deep and wide learning to perform binary classification. This is for the clarity and conciseness of Lazada product title. The main idea is to build an ensemble model from two different models that are both orthogonal and complementary to each other: one is a deep learning model and the other is a wide learning model. In our approach, the deep learning model was built from a small set of features generated from a deep neural network. On the other hand, the wide learning model was constructed on a large amount of features simply extracted from text processing. Our experiments have shown that such a combination of deep and wide learning helps to significantly improve the accuracy of the classification task.

KEYWORDS

Deep and wide learning, neural network, xgboost, text quality and conciseness, binary classification.

ACM Reference format:
Thanh Lam Pham and Quang Hieu Vu. 2017. CIKM AnalytiCup 2017 – Lazada Product Title Quality Challenge: Deep and wide learning for clarity and conciseness classification. In *Proceedings of CIKM conference, Singapore, November 2017 (CIKM'17),* 4 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Since product titles give users/customers the first impression of a product, having good quality product titles is extremely important in e-commerce systems. In many cases, if the titles of products are confusing, lengthy, or poorly formatted, they will have a severe impact on sales. Understanding the importance of having good product titles, Lazada has an internal quality control (QC) team to review their clarity and conciseness. When a product title is neither clear nor concise, it should be updated. This QC process, however, is human time-consuming. As a result, it is desirable to have a product title quality machine learning model that can automatically grade the clarity and the conciseness of a product title. This is the objective of the CIKM AnalytiCup 2017 – Lazada Product Title Quality Challenge, hosted by the ACM International Conference on Information and Knowledge Management (CIKM) [3].

Recent progress in Machine Learning is dominated by two successful learning paradigms. On one hand, simple bagging and boosting techniques are involved in more refined, better guided and faster trained models like the extreme gradient boosted decision trees (XGBoost). On the other hand, deep learning beats benchmark performance records with its ingenious concept of hierarchical learning with iteratively cleaned and selected features perfectly matching convolutional neural networks processing structure. In addition to these learning paradigms, ensemble methods are often used to improve the prediction results. In an ensemble method, individual models are trained separately without knowing each other, and their predictions are combined only at the inference time instead of at training time.

To address the Lazada Product Title Quality Challenge, in this paper, we propose a solution that employs all the advanced techniques mentioned above. In particular, we employ an ensemble model that combines two separate learning models: a deep learning model and a wide learning model. In our solution, while the deep learning model relies on a small set of features generated from a carefully designed and tuned deep neural network, the wide learning model uses a large amount of features extracted through simple text processing. Despite the differences in feature generation and selection, both models use XGBoost in their training process. The 8^{th} place of our solution in this competition has objectively proven its solid design and performance capabilities. This solution surpassed many other solutions submitted through 469 registered accounts from all over the world.

The remainder of this paper is organized as follows. In Section 2, we present related work. In Section 3, we introduce our proposed ensemble model. In Section 4, we show experimental study. Finally, in Section 5, we draw some concluding remarks and discuss potential improvements for the future.

2 RELATED WORK

In this section, we briefly introduce the machine learning techniques used in our model: Deep Learning, XGBoost and Ensemble model.

CIKM'17, November 2017, Singapore

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM. . . $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2.1 Deep Learning

Deep Learning (DL) refers to a class of machine learning techniques and architectures, where many layers of non-linear information processing stages in hierarchical architectures are exploited for representation learning [6]. In particular, a DL network represents a multi-layer neural network with the deeper structures compared to the shallow models like Support Vector Machines and a specific method where the data is processed at and in between layers. Even though the concept of DL was introduced long time ago, it has only gained popularity recently due the lower cost of computing hardware, the increaser speed of chip processing, and recent advances in DL algorithms. DL has been successfully employed for graphical modeling, optimization, pattern recognition, signal processing, and natural language processing [4].

2.2 XGBoost

Boosting is a powerful meta-algorithm used to reduce prediction bias. The basic idea of boosting algorithm is to first produce a series of individually average performing models trained on the subsets of original data and then boost their performance by combining them together using various aggregation functions like majority vote or weighted average [9]. Gradient boosting is a version of boosting method that manages to achieve deeper performance gains compared to state-of-the-art predictors and is commonly used to solve regression and classification problems. XGBoost [2] is an implementation of the gradient boosted decision trees based on the extreme gradient boosting model [8]. Recently, XGBoost has gained increased popularity and attention due to its advantages of fast processing speed and high prediction performance. In particular, it has been used to win top prizes in many machine learning competitions hosted in Kaggle [1], a competitive data science platform.

2.3 Ensemble method

Ensemble methods are learning algorithms that construct a set of classifiers and then classify new data points by taking a (weighted) vote of their predictions [7]. Similar to XGBoost, ensemble method is a popular technique employed by winning teams in Kaggle's machine learning competitions [1]. In our work, we employ a simple average ensemble method from a deep learning model and a wide learning model. While the idea of leveraging deep and wide learning has already been introduced by Cheng et al. in [5], our work is very different from their work because we combine deep and wide learning in an "ensemble method" to serve the purpose of binary classification. Cheng et al. do not utilize ensemble method, instead, they jointly train wide linear models and deep neural networks to combine the benefits of memorization and generalization for recommender systems.

3 METHODOLOGY

In this section, we will first present our deep learning model. After that, we show our wide learning model before discussing our simple strategy to ensemble these models.

3.1 Deep learning model

Our deep learning model has a total of 234 features for clarity classification and 239 features for conciseness classification. In both cases, 200 features are extracted from a deep neural network (NN) that is designed to capture the sequential semantic meaning of the title and the interaction between the title and other features. This NN has four main layers:

- One embedding layer with word embedding to obtain semantic similarities of words based on the pre-trained Glove word2vec [10], character embedding to capture out of vocabulary in the business domain, and orthographic embedding to extract the signal from measurement units, abbreviation and size dimension.
- One shared bidirectional long short term memory (LSTM) layer to encapsulate the sequence of embedding tokens in a single vector of hidden size 100 in which half of the vector is built in a forward process and the other half is generated in a backward process.
- Two dense layers with 150 and 100 hidden nodes respectively to learn the non linear transformation by employing RELU activation.

In between these main layers, there are other supporting layers. They include Dropout layers that are added to avoid noise, a Translate layer to reduce size of each embedding vector from 300 to 100 dimension, Concatenation layers using 100 hidden nodes from title, joined categories and manual engineered features, and Batch Normalization layers which serves the purpose of scaling the input from previous layer to prevent over-fitting and get faster in training time.

In addition to the above 100 features generated from the deep neural network, the deep learning model includes some other features selected from the below list of features whose Pearson correlation between the features and the labels are higher than a predefined threshold of 0.02 for clarity classification and 0.05 for the conciseness classification.

- Label encoding of *cat1*, *cat2*, *cat3*, and *product type*.
- Entropy of chars and words, number of words, chars, ratio of nouns, out-of-vocabulary and stopwords, ratio of adjectives and numbers, word duplication ratio, and 2D TSNE feature of the title.
- Word2vec distance between title and common words in the title, the list of following specific keywords: {location, brand, unit, shipping, color, sexy }, and concatenation of *cat1*, *cat2*, *cat3*
- *Sku* length, number of measurement units and *price* converted in to same scale of Singapore dollars.
- Top 5 mean score and cosine distance by vectorizing title into tfidf or word2vec space.

3.2 Wide learning model

Our wide learning model consists of 3244 features for clarity classification and 3301 features for conciseness classification of which 3200 features are generated from simple text processing. In particular, we generate these 3200 text features in the following ways.

- Clarity text features: given a title, we first remove stop words from it. The reason is because stop words generally do not have any impact on the clarity of the title. In addition to stop word removal, we encode all numbers appearing in the title by a special keyword and all numbers followed by measurement units (e.g. ml, kg, inche...) by another special keyword. Finally, we use *CountVectorizer* method from the python *scikit-learn* library to count the frequency of the top 2700 keywords existing in both data sets (train and test). These are used as 2700 text features. Additionally, we also count the top 500 keywords from each of the 9 *cat1* categories to generate 500 other text features.
- Conciseness text features: we generate conciseness text features similar to the way we do for clarity text feature generation. The differences, however, are (1) we do not remove stop words since stop words do have an impact on the conciseness, (2) we do not encode numbers and measurement units, and (3) we stem the words before running *CountVectorizer* in other to capture root of words for training.

Besides text features, the wide learning model also uses the following sets of features:

- Basic features: they consist both original features from the data set, *country*, *price*, *ptype*, and 3 features extracted from *sku* using characters at positions from 1 to 5, 6 to 7, and 8 to 9 (we found that *sku* itself contains some useful information related to the product, e.g., brand of the product).
- Statistical features: these are features generated from getting statistics of *title* and *desc*. They include the overall length (in both number of characters and number of words), the number of all-digit words, non-digit words, measurement units, the total number of words and the number of repeated words.
- Categorical based features: instead of directly using *cat1*, *cat2*, and *cat3* as features, we generated replacement features by grouping data of these categories and calculating mean and maximum values for certain features on these categories (e.g., we calculate mean and maximum values of total number of words for each of categories in *cat1*, *cat2*, and *cat3*).

These extra features make up a total of 44 other features in the clarity model and 101 other features in the conciseness model. Note that we used different sets of features for clarity prediction model and conciseness prediction model simply because we found that such a separation gave us a better performance. We think that this separation makes sense due to the different natures of clarity and conciseness.

3.3 Result ensemble

Our ensemble method happens in two steps as follows.

- Our first step relies on the simple weighted average where we tuned the weights based on both local validation scores and our submission scores. In particular, given that our deep learning model has a better performance compared to our wide learning model, we set a higher weight for the predictions generated from the deep learning model.
- The ensemble result from the previous step is then adjusted based on means of the input labels. For example, given that we have 34228/36283=94.336% samples with clarity label 1 and 24866/36283=68.533% samples with conciseness labe1 1, we adjusted our predictions to achieve such means values for clarity and conciseness.

4 EXPERIMENTAL STUDY

In this section, we will present our experimental study from the validation phase to the testing phase and discuss our learned lessons.

4.1 Validation phase

Since we had many submissions to use in this phase, we had an opportunity to test all of our ideas and learned many lessons. In particular, with respect to the deep learning model, below are lessons learned.

- We know that a pre-trained word2vec model from Glove needs to be re-trained according to specific task. However, it is important to learn that we also need to employ Translate layer to reduce the number of parameters to make the model work for us. Otherwise, the performance of the model is still not good.
- In order to deal with out of vocabulary words, the embedding of orthographic and character level are beneficial to increase input sequences.
- Using the same model for both clarity and conciseness classification tasks is not optimal. Instead, we have to use separate architectures (in terms of parameters) and different sets of features for each classification task.
- Even though we use end-to-end deep learning to generalize the semantic insights of title, having selected features is still necessary to improve the accuracy of our clarity and conciseness models.

In addition to the above lessons learned from experimental study with the deep learning model, we have the following lessons withdrawn from our experiments with the wide learning model.

- Removing stop words and stemming other words help to improve the score by 0.002 in the clarity classification. However, for the conciseness classification, removing stop words does not help as stop words do contribute to the performance of the model. Without stop words removal and with word stemming, the score of conciseness classification is improved by 0.005. Another lesson with the text processing process is that *PorterStemmer* gives us the better performance compared to other stemmer methods (e.g., *SnowballStemmer* or *LancasterStemmer*).
- Given that all numbers and measurement units are supposed to convey a clear meaning, getting numbers

and measurement units encoded to a single keyword helps to increase clarity score by 0.001.

- Given that the features generated from *CountVectorizer* are sparse, employing XGBoost for the classification is the better than other methods due to its robust to noise and its ability to deal well with sparse data. For example, besides XGBoost, we tried with LogisticRegression and even a Deep Neural Network for this big set of features, but for both models, the scores were lower than that of XGBoost (from 0.01 to 0.02).
- Given a large number of features, only a small subset should be used to train for each tree in the XGBoost. In particular, our best tuned value for *coltree_bysample* is 0.35. Having higher values for this parameter leads to both longer training times and worse scores.

Finally, with respect to our ensemble model, when we tried to reuse some good features of a single model in another one (e.g., copy some good features of the deep learning model to the wide learning model), while we could improve the performance of the latter model a bit, the improvement generated from the ensemble model was lost. This means that the more the difference between our two single models, the better the performance of the ensemble model. In other words, when the two single models are orthogonal and complementary to each other, it is a good signal to ensemble these models to improve the performance.

4.2 Testing phase

Compared to the validation phase, there are not many lessons learned in the testing phase. The reason is simply due to the limited number of submissions in this phase. Nevertheless, there are still some precious lessons as follows.

- Given the number of limited submissions, our initial strategy was to submit only ensemble results from our two models. However, upon receiving the result 0.340752 for clarity and 0.243781 for conciseness, we decided that we still needed to submit a single model separately to see its performance and from the result to guess the performance of the other model. We chose to evaluate the performance of the deep learning model, which was expected to be better than the wide learning model and the result shown that we got 0.345620 for clarity and 0.24363 for conciseness. It is interesting to see that the clarity score of the single model was even better than that of the ensemble model. Thus, an implication is that we had to increase the weight for the deep learning model, and subsequently it helped to increase the ensemble model score a bit later.
- Since most features in our model come from text processing and statistics based on text, we thought that the more data we have, the better features we get for the model. However, upon adding extra data from the valid data set, the performance of our model is worse (a step back of 0.005 from the score). Thus, in later steps, we only used data from the training data set and testing data set for feature generation.
- It is interesting to see that the adjustment of our ensemble results based on means of samples did not work well in this phase even though it helped in the validation phase. The reason is because there is a shift in the means of the results. By submitting all 1's prediction, we could know the correct means of the testing data. Fortunately, we did not need to use our submission slot since we found that other teams did that and we only needed to copy their scores from the leader board. With the correct means from the testing data, our adjustment strategy worked again.

5 CONCLUSION

In this paper, we have presented our ensemble method that combines a deep learning model and a wide learning model to solve the challenge of binary classification for product title clarity and conciseness. While the deep learning model is trained with a deep neural network and only consists of hundred plus features, the wide learning model covers a large set of more than three thousand features. Given that there are many differences between the deep learning model and the wide learning model, an ensemble of the two models helps to significantly improve the performance of the classification.

Since there are some techniques we have not tried in the model due to our limited time spent for the competition, they could be included in the future work. One thing is that we can try to use external data sources. It is because what we have done so far is limited to the provided data sets. However, we believe that the model can be improved with external data sources (e.g., by simple crawling the Lazada website to extract more data or getting external data from Amazon for similar products). Another thing we can do in the future is to employ other techniques for text processing (e.g., to use TruncateSVD to reduce the dimensionality of the feature space for the wide learning model).

REFERENCES

[1] 2010. Kaggle. https://www.kaggle.com/. (2010).
[2] 2016. XGBoost. https://github.com/dmlc/xgboost/. (2016).
[3] 2017. CIKM AnalytiCup: Lazada Product Title Quality Challenge. https://competitions.codalab.org/competitions/16652. (2017).
[4] Y. Bengio, A. C. Courville, and P. Vincent. 2013. Representation learning: a review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 8 (2013).
[5] H. T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, and H. Shah. 2016. Wide & Deep Learning for Recommender Systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS)*. 7–10.
[6] L. Deng. 2012. Three Classes of Deep Learning Architectures and Their Applications: A Tutorial Survey. *APSIPA Transactions on Signal and Information Processing* (2012).
[7] T. G. Dietterich. 2000. Ensemble Methods in Machine Learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems (MCS)*. 1–15.
[8] J. H. Friedman. 2001. Greedy function approximation: a gradient boosting machine. *The Annals of Statistics* 29, 5 (2001).
[9] L. Mason, J. Baxter, P. Bartlett, and M. Frean. 1999. Boosting algorithms as gradient descent. In *Proceedings of the International Conference on Neural Information Processing Systems*.
[10] J. Pennington, R. Socher, and C. D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543.