URL Classification with Deep Learning

Xiong Kexin
A0196551X
e0389037@u.nus.edu

Abstract—**This report is for the deep learning section of CS4248 final project. Web pages are usually categorized based on their titles and content, which can be very time consuming. Nowadays, URLs are becoming more and more human readable, which drives the use of Natural Language Processing (NLP) techniques to parse URLs and infer the category of a web page from them. In this project, we limit ourselves to performed web page classification only using Uniform Resource Locators (URLs). We applied deep learning classifiers on a dataset of more than 1.5M URLs across 15 categories, obtained from online repository dmoz. Given the enormity of the dataset, we ultimately trained our models on subsets of 5 categories, and achieved an accuracy of 81.23% using Deep Learning models.**

I. Introduction

Web pages classification (or categorization) is a machine learning problem which gets increasingly important day by day. Since the beginning of the Internet in the 1990's, the number of Internet users and the number of web pages serving users has grown at an exponential rate, resulting in a massive amount of information being readily available online and a rise in browser searches. Web page categorization promotes the development of information retrieval and searching algorithms, which can provide useful information for various downstream applications. For instance, web searching completes the knowledge-base of Question Answering [7] [8]; retrieval of detailed information about user intentions benefits recommender systems [2]–[6] in e-commerce websites. It is thus crucial to constantly improve the speed of finding relevant information from millions of websites. For this reason, topic-based classification on web pages is needed to be made.

Furthermore, web sites must be classified in order to establish internet access policies for organizations or individuals. Cyber protection tools may use web page classification to prevent malicious content from being viewed until it is displayed by the user. It's also is useful when one receives a link from a dubious source and would like to know more about its origins. The classification of web pages automatically necessitates the processing of enormous volumes of data. Manuel web page classification is very time-consuming. Therefore, manually classified web pages constitute a relatively limited percentage of current web pages.

Our project aims to classify a corpus of 1.5 million web pages from the dmoz Open Directory Project into their 15 classes [1]. URL classification is the process of assigning a URL to one of the predetermined categories. To classify a URL, we first need to extract feature representation from it. Feature extraction include tokenization, utilizing lexical features, URL component segmentation and word-level or character-level embeddings. We then pass these features to classifiers and compare their accuracy and f-1 scores.

In this project, classifiers are developed by using deep learning approaches. Deep learning techniques are artificial neural network-based systems with multi-layered architectures. In recent years, deep learning approaches to a variety of machine learning challenges have shown excellent results. In the case of natural language processing problems, Recurrent Neural Network based approaches can provide good results. For example, seq2seq architecture is widely used in machine translation and neural question generation [9], [10]. Convolutional Neural Network approaches are also found effective in some NLP applications. In this project,our deep learning classifiers are developed using CNN and Bi-directional RNN architectures.

April 27th 2021

II. Related Work

Due to the rapid development of computer and network technologies, the popularity of the web has exploded in a brief span of time. As a result, web pages classification has become an increasingly important issue today. While there are some structural differences between URL classification and conventional text classification problems, the algorithms and methods used in the classification process are quite similar.

Deep Learning has become popular in text classification over the past few years and has achieved remarkable results. Much of the work with Deep Learning methods involves learning word vector representations and performing classification tasks over learned vectors [14]. Words or tokens are projected from a one-hot encoding to a lower dimensional space to form word vectors. This process, known as word embedding, can be considered as a feature extractor that encodes the semantic and contextual features into a dense space. As a notably successful use of deep learning, embedding has been applied in various domains, including text, vision [12] or even music [11]. Besides word-level embedding, some researchers also introduced character-level embedding, which is to transfer characters as 1-hot encoding and project them onto unique vectors. This way of embedding can further decrease the dimensionality and better handle the rare words.

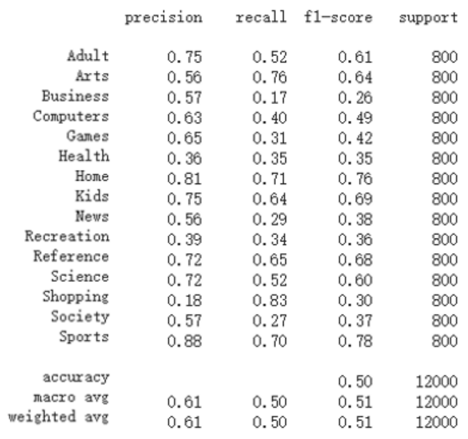
After getting feature representations of text input, neural networks are typically applied. Many neural networks such as long short-term memory (LSTM) and convolution neural networks (CNN) [15], [16] have proven to be powerful in text classification. Instead of traditional classifiers using statistical features, neural networks have the ability to automatically identify useful patterns from certain groups of words or characters appearing together. URL classification can be considered as a type of text classification. However, due to the special structure of URL text, some methods other than word embeddings and character embeddings have been proposed for better feature extraction [13]. To explore the effectiveness of Deep Learning towards URL classification, we conducted experiments using different classifiers and different embedding methods, then compared the results.

III. Methodology

A. Data Pre-processing

We first did some exploratory data analysis on the raw database of 1.5 million urls, which was unevenly distributed amongst 15 different categories. After dropping all invalid rows, the training data was still more than sufficient, so we randomly sampled 10% of them to perform some preliminary experiments. The traditional classifiers and Deep Learning models were unable to achieve any satisfactory results. We hypothesised that there were too many classes in the raw dataset, making classification difficult. Thus, we decided to select several classes from the raw dataset, and perform experiments on this reduced dataset instead.

We ran a Naive Bayes classifier on a dataset that was balanced in each category, then compared the precision, recall and f1 score (shown below) to find the 5 easiest (Arts, Home, Kids, Reference, Sports) and the 5 hardest (Business, Health, News, Shopping, Society) categories to classify.



```
              precision    recall  f1-score   support

       Adult       0.75      0.52      0.61       800
        Arts       0.56      0.76      0.64       800
    Business       0.57      0.17      0.26       800
   Computers       0.63      0.40      0.49       800
       Games       0.65      0.31      0.42       800
      Health       0.36      0.35      0.35       800
        Home       0.81      0.71      0.76       800
        Kids       0.75      0.64      0.69       800
        News       0.56      0.29      0.38       800
  Recreation       0.39      0.34      0.36       800
   Reference       0.72      0.65      0.68       800
     Science       0.72      0.52      0.60       800
    Shopping       0.18      0.83      0.30       800
     Society       0.57      0.27      0.37       800
      Sports       0.88      0.70      0.78       800

    accuracy                           0.50     12000
   macro avg       0.61      0.50      0.51     12000
weighted avg       0.61      0.50      0.51     12000
```

Fig. 1. Selection of 5 easiest and 5 hardest categories

After reduction, we still found an imbalanced distribution in our newly selected dataset. To find out if an unbalanced distribution has influence on our model, we performed upsampling to correct this severely unbalanced distribution (Fig2).

B. Deep Learning classifiers

While traditional classifiers can achieve good performance in URL classification, there are still some limitations in extracting URL features: (1) inability to

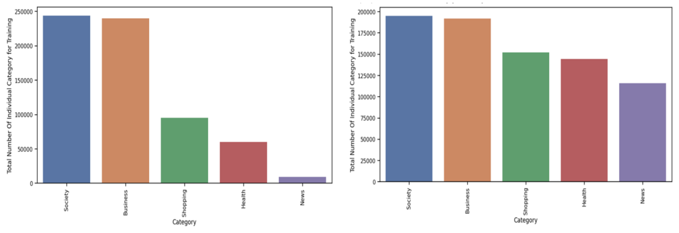


Fig. 2. Upsampling

learn sequential or semantic meaning. (2) require too much human work in feature engineering. (3) inability to handle new words that did not appear in the training process. To address and solve the above issues, we turned to the Deep Learning methods.

For Deep Learning, expert feature engineering was omitted, since the neural networks can automatically learn features. First, we tried models like CNN and LSTM with word embeddings, tokenizing each URL with punctuations. However, these did not give satisfactory results. We even tried the most popular and successful NLP model Bert, which was also unsuccessful. After much discussion, we believed that it is because we used word-level embedding when Bert is already pre-trained at the word-level. We felt that a key difference between our task and traditional text classification is that word-level features in URLs are usually much weaker. Intuitively, urls contain many non-alphanumeric characters and "irregular words", which makes it hard to build dense and meaningful word embeddings. All our experiments using word-level embeddings have poor performances compared to other methods (see in 4. Evaluation).

Thus, we tried character embedding. We first identified unique characters and represented each as a vector. URLs were then transformed into matrices encoded with alphabetic order, on which CNNs and RNNs were applied. Due to the limited number of characters, character embedding easily generalizes new words. It can also relate "irregular words" to regular words by comparing their alphabetic ordering. For example, it is hard to accurately split the URL token "whatsmyname" into several correct words, but character embedding can still learn the similarity between "whatsmyname" and "whats" "my" "name" through observing similar alphabetic order (Fig3).

whatsmyname
whats
my
name

Fig. 3. "whatsmyname" related to "whats" "my" "name"

Apart from char-level embeddings, we also tried an advanced word embedding method in the form of URLNet [13]. URLNet obtains the word embedding as a combination of the original word embedding and the embeddings of the individual characters in that word. We set any word that appears only once to an unknown token in word embedding, and represent it as a unique vector using character embedding. The final URL representation is the sum of two matrices. This method of embedding solves the inability to embed rare words in word-based models, and extracts richer features to represent the urls. This process, as explained in the original paper, is shown in Fig 4 below:

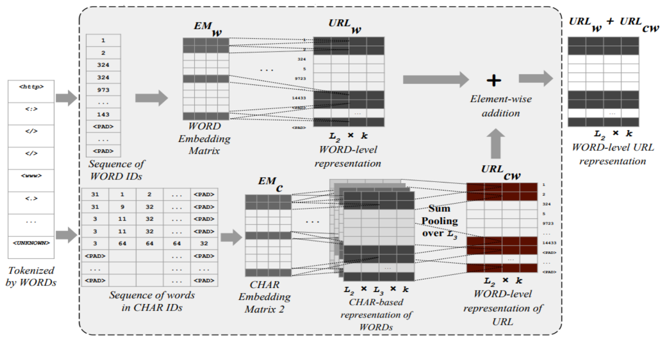

EM_W
URL_W
URL_W + URL_CW
Element-wise addition
Sequence of WORD IDs
WORD Embedding Matrix
L_2 × k WORD-level representation
L_2 × k WORD-level URL representation
Tokenized by WORDs
EM_C
URL_CW
Sum Pooling over L_3
Sequence of words in CHAR IDs
CHAR Embedding Matrix 2
L_2 × L_3 × k CHAR-based representation of WORDs
L_2 × k WORD-level representation of URL

Fig. 4. URLNet [13]

Moreover, to explore whether the poor results of word embedding is due to the lack of background in certain domains, we pre-trained a word embedding matrix in certain domains using dmoz.csv, which contains descriptions of websites in many domains. Since those descriptions do not match with the urls, we cannot simply add features extracted from descriptions to our url features. Therefore, we use those descriptions as an auxiliary corpus in certain domains to pretrain word vectors. To rule out other factors, we only pre trained word embeddings on one category and applied it to the "one vs rest" experiment in said category, to see if there's any improvement.

		Easy Dataset		Hard Dataset	
Embedding method	Model	Accuracy	F1 score	Accuracy	F1 score
word-level	CNN	74.51%	0.7314	69.12%	0.6705
word-level	Bi-LSTM	74.37%	0.7419	73.94%	0.7327
char-level	TextCNN	81.23%	0.8107	74.41%	0.7346
char-level	Bi-LSTM	78.90%	0.7865	70.99%	0.7
URLNet	URLNet	79.53%	0.7913	72.61%	0.7194

TABLE I
EXPERIMENT RESULTS

		Business (One vs Rest)		Shopping (One vs Rest)	
Embedding method	Model	Accuracy	F1 score	Accuracy	F1 score
word-level	CNN	79.43%	0.7943	87.62%	0.8762
word-level	Bi-LSTM	81.47%	0.8147	87.81%	0.8781
pretrained word-level	CNN	79.27%	0.7926	running	running
pretrained word-level	Bi-LSTM	81.17	0.8117	running	running
URLNet	URLNet	79.53%	0.7913	72.61%	0.7194

TABLE II
ONE VS REST EXPERIMENT RESULTS

IV. EXPERIMENTS

This section presents the results from our various experiments mentioned in part 3. We compared three different ways of embedding: character embeddings, word embeddings and URLNet embeddings. We chose TextCNN and Bi-directional LSTM as the neural network classifier, which have been proven to be the most effective. Our experiment results show that character embeddings and URLNet are better than pure word embeddings. This is because urls contain many non-alphanumeric characters and "irregular words" which makes it difficult to extract meaningful word features. TextCNN with character embeddings achieved the highest accuracy, 81.23% for the 5 easiest categories and 74.41% for the 5 hardest categories. (Table 1)

We also tried pre-trained word vectors in one vs rest experiments. The results show that there's not much difference between using and not using word embeddings pre-trained on corpus in certain domains. (Table 2)

V. DISCUSSION

Among all the Deep Learning models, TextCNN with character embedding achieved the highest accuracy of 81.23% and 74.41%, which surpasses all the other models.

According to the results of the experiments, character embedding is better than word embedding for URL classification. This is because urls contain few completed, meaningful words, which makes it hard for word embeddings to extract high quality word features. Character embedding is able to extract sequential information of characters. It can also relate "irregular words" to regular words by identifying their similar alphabetic ordering. However, character embedding is not capable of capturing "word senses" from URLs, which may be one of the reasons why the improvement is not obvious. Besides, since we simply tokenized each URL with punctuations in the deep learning section, there could be further improvement if more advanced tokenizers are applied.

We also conducted one vs rest experiments in "Business" and "Shopping" domains using word embedding pre-trained on the specific domain. Since the word embedding can only capture limited senses in our training URLs, we thought that a lack of domain-specific pre-trained embeddings may also be a reason why word embedding performs poorly. However, the results show that there's not much difference between using and not using pre-trained embeddings, which indicates that the lack of background in certain domains may not be the reason why word embedding performs badly.

VI. Conclusion

In this project, we have experimented with different embedding methods and different classifier models to help with URL classification. It is evident that character-level embedding with Deep Learning models perform slightly better than any other way of embedding. Even at our best, URLs just simply contain too much noise for meaningful word-level separation to work.

All in all, we have achieved moderate success despite our limited computing resources. With more resources, we should be able to find better tokenizers, better embedding methods, better feature extraction methods and Hyperparameter tuning. This would allow rapid, high-level web page classification based on URLs an achievable reality.

Acknowledgment

We would like Professor Kan Min-Yen, and the entire teaching team of CS4248 - Natural Language Processing for all their help and support throughout the semester.

References

[1] Kaggle URL Classification Dataset [DMOZ] https://www.kaggle.com/shawon10/url-classification-dataset-dmoz

[2] Gao, C., Lei, W., He, X., de Rijke, M., & Chua, T. S. (2021). Advances and Challenges in Conversational Recommender Systems: A Survey. arXiv preprint arXiv:2101.09459.

[3] Lei, W., He, X., de Rijke, M., & Chua, T. S. (2020, July). Conversational Recommendation: Formulation, Methods, and Evaluation. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 2425-2428).

[4] Lei, W., He, X., Miao, Y., Wu, Q., Hong, R., Kan, M. Y., & Chua, T. S. (2020, January). Estimation-action-reflection: Towards deep interaction between conversational and recommender systems. In Proceedings of the 13th International Conference on Web Search and Data Mining (pp. 304-312).

[5] Lei, W., Zhang, G., He, X., Miao, Y., Wang, X., Chen, L., & Chua, T. S. (2020, August). Interactive path reasoning on graph for conversational recommendation. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 2073-2083).

[6] Li, S., Lei, W., Wu, Q., He, X., Jiang, P., & Chua, T. S. (2020). Seamlessly Unifying Attributes and Items: Conversational Recommendation for Cold-Start Users. arXiv preprint arXiv:2005.12979.

[7] Zhu, F., Lei, W., Wang, C., Zheng, J., Poria, S., & Chua, T. S. (2021). Retrieving and Reading: A Comprehensive Survey on Open-domain Question Answering. arXiv preprint arXiv:2101.00774.

[8] Zhang, Y., Zhang, X., Wang, J., Liang, H., Jatowt, A., Lei, W., & Yang, Z. (2020). GMH: A General Multi-hop Reasoning Model for KG Completion. arXiv preprint arXiv:2010.07620.

[9] Lei, W., Jin, X., Kan, M. Y., Ren, Z., He, X., & Yin, D. (2018, July). Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 1437-1447).

[10] Pan, L., Lei, W., Chua, T. S., & Kan, M. Y. (2019). Recent advances in neural question generation. arXiv preprint arXiv:1905.08949.

[11] Liang, H., Lei, W., Chan, P. Y., Yang, Z., Sun, M., & Chua, T. S. (2020, October). PiRhDy: Learning Pitch-, Rhythm-, and Dynamics-aware Embeddings for Symbolic Music. In Proceedings of the 28th ACM International Conference on Multimedia (pp. 574-582).

[12] Frome, A., Corrado, G., Shlens, J., Bengio, S., Dean, J., Ranzato, M. A., & Mikolov, T. (2013). Devise: A deep visual-semantic embedding model.

[13] Le, H., Pham, Q., Sahoo, D., & Hoi, S. C. (2018). URLNet: Learning a URL representation with deep learning for malicious URL detection. arXiv preprint arXiv:1802.03162.

[14] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. Journal of machine learning research, 12(ARTICLE), 2493-2537.

[15] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

[16] Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. arXiv preprint arXiv:1509.01626.