A Deeper Exploration of Natural Language Processing in Chinese

Yuxiang Pan
McGill University / Montreal, Canada
yuxiang.pan@mail.mcgill.ca

Ava Gilmour
McGill University / Montreal, Canada
ava.gilmour@mail.mcgill.ca

Haiqi Zhou
McGill University / Montreal, Canada
haiqi.zhou@mail.mcgill.ca

Abstract

We present a Python implementation of the Natural Language Processing (NLP) task of text classification for text in Chinese, with a deeper focus on feature engineering to include stroke and radical information for individual Chinese characters. For multi-class classification of text, we utilized Multinomial Naive Bayes Classifiers with identical hyperparameters to evaluate the performance of models with different feature selection strategies. The results of the model for different feature levels are then compared and the results are discussed.

The method of feature extraction concentrates on individual features of characters as opposed to a greater word or phrase level structure to develop a text classification model.

1 Introduction

While languages with compositional orthography, utilizing alphabets like Latin and Cyrillic, Chinese script is composed of individual characters. Unlike English whose smallest meaningful unit is derived from the combination of characters/alphabets into words, the semantic meaning of Chinese characters stems from their pictograph origin.

Each Chinese character is maximally made up of four radicals (偏旁部首) that contain individual semantic meanings. For example, the character 明 (míng), usually denoting brightness, is composed of 日 (rì), meaning sun, and 月 (yuè), meaning moon. In this manner, some aspect of semantic compositionality is present in Chinese orthography that may provide additional information to machine learning models in processing Chinese text.

Often, words from similar semantic categories have the same radicals, such as the radical 木 (mù), meaning wood, which is found in many plant-related words such as 林 (lín), meaning forest, and 果 (guǒ), meaning fruit.

Figure 1: Example of radical 木 and its pictograph relation with plants

Chinese characters are additionally composed of individual strokes which are ordered by various formation rules that remain consistent across pictographs. Both characters and radicals can then be broken down into a finite set of strokes in a set particular order. For instance, the character 木, which is also a radical, is written in four strokes: one horizontal (called 横 (héng)), one vertical (竖 (shù)), one left-slanting downward stroke (撇 (piě)), and one right-slanting downward stroke (捺 (nà)). As radicals are consistently characterized by these strokes in the same order, stroke orderings can encode radicals which can in turn encode semantic information.

Inspired by the article "Character-level Convolutional Networks for Text Classification," [ZZL15] which explores the potential of applying CNNs to conduct character-only text classification, we decided to similarly decompose individual Chinese characters to analyze the impact that further decomposition of characters in feature engineering have on the performance of a probabilistic linear classifier. Our goal is to compare the performance of the same model on the different granularity of features, which are word level, character level, radical level, and stroke level. From these results, we will deter-

mine if the finer levels of features are helpful on their own.

1.1 Related Work

Prior research has found that using radicals and strokes in word embeddings has led to better or comparable performance to other methods of vectorization [ea20]. The experiment shows outstanding performance in capturing local features and outperforms state-of-the-art approaches. Similar radical embedding has seen success in character vectorization [ea15] where complex methods including the use of stroke n-grams have improved performance in certain tasks.

The use of stroke n-grams is due to the semantic inconsistencies between the individual meaning of radicals and the characters that they compose; despite sometimes containing useful semantic information, radicals are often unrelated to the meaning of the character. For instance, the character 歌 (song) is made up of the radicals 哥 (brother) and 欠 (lack of), which are far away from the meaning of "song." As such, stroke n-grams are useful both as the sole embedding method, as presented in [ea18a], and in tandem with radical embedding [ea20], with the latter performing slightly better than the former. Both methods outperform the state-of-the-art embedding methods such as word2vec and GloVe.

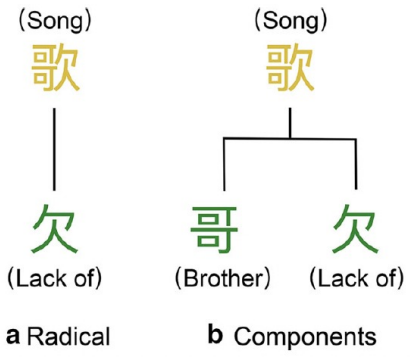

Figure 2: Example of radical decomposition

However, relying on additional information encoded by radicals is not always helpful in machine learning tasks; in particular, radical embeddings are not particularly effective in neural machine translation [ea18b]. With this in mind, we wanted to interrogate optimal methods for character embeddings with various design combinations (radical embedding, stroke n-grams, radical + stroke n-gram embedding).

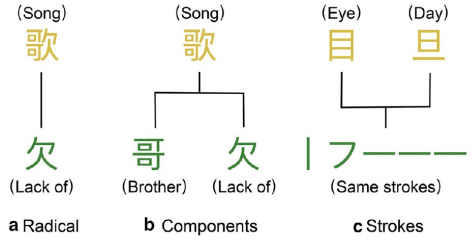

Figure 3: Graph for the extended approach

2 Setup

For the model, we utilized sklearn's Multinomial Naive Bayes Classifier due to its prior consistent performance in similar text classification tasks such as those defined in Programming Assignment 1.

To develop mappings of individual characters to stroke orders and radicals, we relied on existing databases to extract these features. For stroke features, the extracted feature set from [ea18a] was used to map characters to the strokes that compose them. This was performed by parsing character text data and replacing each character with its corresponding strokes to develop character-level n-grams from the corpus.

For extracting radicals, a union of 2 databases is used: $database.csv$ contains 11 thousand entries and $char2comp.txt$ contains 5 thousand [1]. Note that the radicals extraction will be incomplete since there are around 55 thousand Chinese characters and even without counting the repetition, both databases sum up to 16 thousand. Thus, Out-Of-Vocabulary (OOV) items will be assigned naively with the token $[UNK]$.

For the n-gram model, the Count Vectorizer was passed the n for the n-gram to develop counts accordingly. Additionally, the model was tested with `analyzer=word` and `analyzer=char` which determine the boundaries that determine the creation of the n-grams by the vectorizer.

[1] partial implementation and databases can be found in the following GitLab repo: https://gitlab.cs.mcgill.ca/yuxiang.pan/comp-550-final-project

2.1 Data

To test our character, stroke, and radical-based models, we used the Wikipedia Title Dataset which contains a list of 12 categories for 160,000 titles of Wikipedia articles. All of these titles are in simplified Chinese. The length of each title varies from 2 to 25 characters in length. In preprocessing this data, we removed alphanumeric characters as well as punctuation in order to isolate written Chinese text data.

3 Results

The stroke-level model performed better than both the radical and character-level models in the task of text classification with an average accuracy of 53.4% on the training set and 50.3% on the testing data. The model performed better with larger n-grams, reaching a peak accuracy of 63% on the training data and 57% on the test set when including n-grams up to 15-grams from 3-grams. When very large n-grams are created, however, the model overfit onto the training data, as seen in the (25, 30) n-gram range model. This increase in performance, however, comes at a cost of a drastically longer runtime as more n-grams are created and accounted for by the model. For example, when trigrams to 15-grams were specified to the model, classification of the entire dataset took 8 minutes and 47 seconds while when trigrams to 5-grams were specified, the model terminated after 58 seconds of computation.

On the other hand, the radical-level model yields lower accuracy than the stroke-level, but with similar climb. It has an average accuracy of 32.4% for the training data, and 14% for the testing data. The peak performance happens in moderately large n-gram models, which are 47% and 27% for training and testing and performs poorly with large n-gram models as observed in the last entry of Table 2.

The character-level classification is achieved by first preprocessing our data with TFIDF vectorizer, and then training a Multinomial Naive Bayes model with the vectorized data. The result is quite simple, hence there is no need for an individual table. From observation, the training accuracy is around 94% whereas the testing accuracy is only 22%.

Analyzer	n1	n2	Training Accuracy (%)	Test Accuracy (%)
char	1	2	39	39
char	1	3	48	48
char	1	4	53	53
char	1	5	56	54
char	2	2	40	40
char	2	3	48	48
char	2	4	53	53
char	2	5	56	54
char	3	3	49	48
word	3	3	49	49
char	3	4	53	52
char	3	5	56	53
word	3	10	59	41
char	3	10	60	56
word	3	15	59	41
char	3	15	63	57
char	25	30	84	41

Table 1: Results from the stroke-level n-gram model.

Analyzer	n1	n2	Training Accuracy (%)	Test Accuracy (%)
char	1	2	25	20
char	1	3	28	21
char	1	4	30	24
char	1	5	31	25
char	2	2	27	20
char	2	3	32	22
char	2	4	35	23
char	2	5	36	23
char	3	3	35	19
char	3	4	37	22
char	3	5	38	22
char	3	10	45	26
char	3	15	47	27
char	25	30	8	3

Table 2: Results from the radical-level n-gram model.

4 Analysis

The better performance of the stroke n-gram model relative to the character-level model may be due to the increase in feature data as larger n-grams are required to capture the detail of more pictographically complicated characters. Because different characters require fewer strokes than others, there may be inconsistencies in the encoding of such characters when stroke-level n-grams are utilized. The universal increase in performance of the model when larger n-grams were created hints that these complicated characters are not well encoded by smaller n-grams and semantic detail may be lost.

Additionally, due to the consistent and repetitive presence of certain radicals in characters, these smaller n-grams may be too general to accurately predict categorization. The decomposition of character data into stroke data drastically increases the length of the input data to the model, resulting in extended computation time as well as memory usage. While this decomposition may be more computationally intensive on both the model and the vectorizer, the increased information provided by the stroke data is shown to increase the performance of a base linear probabilistic model. Overall, the use of stroke n-grams as input to a linear probabilistic model resulted in the best accuracy of our classifiers.

Considering the results from the radical-level n-gram model, the change in accuracy with respect to n-gram parameters follows the general trend of stroke-level but with lower accuracy. A reasonable hypothesis is that there exists ambiguity in radical identification as radicals can have multiple meanings, and the model in this experiment is not designed to handle such ambiguity. Stroke-level model triumphs because strokes are unique and instead of handling ambiguous relations, they completely disregard the consideration. However, the trend differs with the large n-gram model as the accuracy drops significantly. One can explain such an observation by error propagation in radical identification. In other words, when a radical n-gram is large, then uncertainty starts to pile up, thus accuracy decreases as confidence drops.

The observation of the character-level model indicates that the model is overfitting as training data yields outstanding accuracy while new data (testing) gives poor performance. Therefore, an improvement would be to use any sort of generalization technique, such as cross-validation to assess the model on multiple subsets of the data to increase overall robustness. Since the Chinese database is relatively hard to find, one can also use data augmentation to manually increase data size.

5 Conclusion

We have presented an analysis of text classification of Chinese language data focusing on the effects that different levels of feature granularity had on model performance. Our work presents information as to the efficacy of radical, character, and stroke-level feature engineering and how such strategies affect the performance of a linear probabilistic model.

The approach in this paper is particularly relevant in a complex character-based language like Chinese. Without such a simplification, NLP in Chinese needs to deal with thousands of unique characters, which entails catastrophic computational resources as the baseline complexity of NLP algorithms is already large. Working in a finer scope, one only needs to deal with hundreds of radicals or dozens of strokes, which is a significant reduction. If this decomposition is applied to Chinese text, NLP strategies that may work for alphabet-based languages may be applied to character-based languages as well.

However, due to the low performance of the stroke n-gram-based model and the radical-based model, it is evident that these features are insufficient in alone characterizing text and should ideally be combined to develop a more robust classification model.

5.1 Future Work

While we mainly concentrated on the feature engineering portion of developing a machine learning model as opposed to tweaking the model itself, experimentation with different kinds of models would be useful in determining the efficacy of encoding text with different levels of detail. For example, a CNN with multiple channels could utilize infor-

mation from different levels of detail (i.e. stroke, radical, and character). This might assist in developing more detailed embeddings of individual characters. Further experimentation with linear models or LSTMs could also illuminate potential avenues of further development when utilizing radical and stroke information for characters.

5.2 Contribution

Gilmour implemented the model and stroke-level tasks and set up the default training code, Pan performed radicals-level tasks and the majority of the report with its formalization in LaTeX, and Zhou did character-level tasks and database retrieval with preprocessing.

References

[ea15] Shi et al. Radical embedding: Delving deeper to chinese radicals. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, 2015.

[ea18a] Cao et al. cw2vec: Learning chinese word embeddings with stroke n-gram information. *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[ea18b] Tan et al. Radical-enhanced sequence to sequence model for chinese-english neural machine translation. *LIGN167: Deep Learning for Natural Language Processing*, 2018.

[ea20] Wang et al. Radical and stroke-enhanced chinese word embeddings based on neural networks. *Neural Process Lett 52, 1109–1121*, 2020.

[ZZL15] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in Neural Information Processing Systems 28*, 2015.