Character Jacobian: modeling Chinese character meanings with deep learning model

Yu-Hsiang Tseng
Graduate Institute of Linguistics
National Taiwan University
seantyh@gmail.com

Shu-Kai Hsieh
Graduate Institute of Linguistics
National Taiwan University
shukaihsieh@ntu.edu.tw

Abstract

Compounding, a prevalent word-formation process, presents an interesting challenge for computational models. Indeed, the relations between compounds and their constituents are often complicated. It is particularly so in Chinese morphology, where each character is almost simultaneously bound and free when treated as a morpheme. To model such word-formation process, we propose the Notch (Nonlinear Transformation of Character embeddings) model and the character Jacobians. The Notch model first learns the non-linear relations between the constituents and words, and the character Jacobians further describe the character's role in each word. In a series of experiments, we show that the Notch model predicts the embeddings of the real words from their constituents and helps account for the behavioral data of the pseudowords. Moreover, we also demonstrated that character Jacobians reflect the characters' meanings. Taken together, the Notch model and character Jacobians may provide a new perspective on studying the word-formation process and morphology with modern deep learning.

1 Introduction

Recent years have witnessed a growing interest in modeling the internal semantic dynamics of compounds. Indeed, compounding is sometimes argued as a language universal, and it is claimed to be *protolinguistic fossils* which modern languages frequently elaborate. Compounds are usually loosely defined as forming words with two independent words, such as blackboard and pineapple (Libben, 2014; Bauer, 2009; Jackendoff, 2002). It is apparent that the number of potential combinations is already enormous, even in this simplest form of the two-constituent compound.

The productivity of compounding is particularly evident in Chinese word-formation. Due to the debating nature of Chinese wordhood, there is a vague boundary between word-forming affix and bound root when treating a Chinese character (字 zì) as a morpheme (Huang et al., 2017; Hsieh et al., 2018; Tseng et al., 2020). There is considerable flexibility in Chinese characters, where morphemes can be joined with one another either preceding or following them. For example, the following four words all start with the characters which are the ends of the previous one: 長老 zhǎng lǎo "elder", 老師 lǎo shī "teacher", 師範 shī fàn "teacher-training", 範圍 fàn wéi "area". The versatility of characters leads Hoosain (1992) to describe Chinese text as "a continuous parade of meaningful individual characters (morpheme)" (see Packard (2000) for a complete introduction in Chinese morphology).

However, the productivity of Chinese characters is hard to capture by a computational model. A traditional natural language pipeline starts with word segmentation, which removes the sublexical cues (characters) in the first step. Even for the later distributional semantic model with sub-word information, such as FastText (Bojanowski et al., 2016), it cannot accommodate the different meanings carried by individual Chinese characters. Recent deep learning models, e.g., BERT (Devlin et al., 2018), provide contextualized embeddings of each token. But, as the final representations are mixed (hence contextualized), it is unclear how the embeddings could relate to the original character tokens.

Therefore, we propose the Notch model (Nonlinear Transformation of Character embeddings), and with which we derive character Jacobians. We first train the Notch model to learn the relationships between the constituent

Proceedings of the 29th International Conference on Computational Linguistics, pages 152–162
October 12–17, 2022.

characters' and the whole word's meaning. Next, we show that character Jacobians derived from the model capture the different roles of characters in the whole word. The rest of this paper is organized as follows. First, in section 2, we briefly review the relationship between Chinese characters and words and their complicated behaviors in Chinese morphology. Section 3 describes the Notch model and shows that it not only predicts the real words but helps account for the behavioral data of pseudoword. Finally, in 4, we explore the character Jacobians and evaluate them on a Chinese affixoid dataset.

2 Related works

The vague boundary between affixation and compounding does not only occur in Chinese word-formation. For example, Plag (2003) points out the problem of *neoclassical elements*, such as the word *bio-logy*. While one would tend to treat *bio-*, and *-logy* as prefixes and suffixes, which would violate the basic assumption about the word structure; that is, there will be no *root* in this word. Plag argues that these words, often called combining forms, should be best treated as compounds.

Studying compounds and their constituents poses interesting questions in computational modeling. As the meaning of the compounds may be free from its composing elements, determining the meaning of a newly encountered compound is thus difficult (Jackendoff, 2002). One interesting attempt is to model the meanings of the constituents separated from their free-word counterparts (Günther and Marelli, 2021; Libben, 2014). The difference between the *as-constituent* and free-word representations is called *semantic shift*. A linear model is then built to simultaneously estimate the semantic shifts of the constituents and their linear relations with the compounds by linear algebra.

In Chinese word formation, a character may play different morpho-semantic roles in different words, even if they are in the same position. Therefore, it may not be straightforward to accommodate such versatility into a single linear transformation. Yet, we could consider the relations between the constituents and compounds as a complex non-linear function; the linear transformation is then a local approximation at that specific local neighborhood. Here, we use the Jacobian matrix to obtain the local linear approximation, which is previously used to construct a saliency map and understand network properties (Papernot et al., 2016; Wang et al., 2016). However, before we compute and evaluate the Jacobian, we should first build our non-linear function between constituent and word, that is, the Notch model.

3 The Notch model

The purpose of the Notch model is two-fold. First, it should learn the semantic relationships between the variable-length character sequences (i.e., the constituents) and the whole words. As the semantics of constituents and words are both described by a semantic vector space, the model-learned relationship is essentially a function of $R^{kn} \rightarrow R^{n}$, with n being the semantic space's dimension and k being the word length. Second, the model also provides the Jacobians with which we characterize the character's role in a given word. Therefore, we first train the model to predict the whole word's embedding from the embeddings of its constituents. [1]

3.1 Model training

We trained the Notch model to learn the relations between whole words and constituents in a given embedding space. The embedding space we used was the Chinese word embeddings from Tencent AI lab (Song et al., 2018). The embedding dataset consisted of two million words with 100 dimensions. As the Tencent embedding is a more task-oriented NLP resource, entries (both simplified and traditional Chinese) included may contain both fine-grained words (words from conventional word segmentation) and coarse-grained words (short phrases or compounds in linguistic senses) [2]. Therefore, we chose the first 500 thousand of them for the more commonly used words. For comparison, in a manually-segmented Taiwan Mandarin corpus

[1] The code is available at https://github.com/lopentu/character-jacobian

[2] The steps of word segmentation of Tencent embeddings are described on https://ai.tencent.com/ailab/nlp/en/embedding.html

of 15M characters (CKIP, 1998), the number of unique words (word types) is 217K, and 122K of them occurred more than once in the corpus (i.e., words that are not hapax legomena).

These 500 thousand words are further randomly split into 490K training words and 10K testing words. Each word corresponds to a 100-dimension word vector. The word vectors were all first normalized into unit length. The training words comprise 1,809,674 character tokens, which are 8,749 unique characters (character types). There are 7,792 single-character words, 119,062 two-character words, 103,099 three-character words, 166,330 four-character words, and 93,717 words having five or more characters.

The Notch model's architecture was based on a pre-trained BERT (Devlin et al., 2018) (based on `bert-base-chinese` model) and followed by a task-specific fully-connected layer. At first, the model took a sequence of variable-length characters as input, which were first tokenized by character with the pretrained `bert-base-chinese` tokenizer. Next, the BERT's encoded representation of the first `[CLS]` token was further transformed with a fully-connected layer, which is responsible for projecting the embeddings from the BERT model space of 768-dimension into Tencent's embedding space of 100-dimension. The model was trained with a mean-squared-error objective, where the model tried to minimize the error between the predicted embeddings and the actual embeddings. AdamW (Loshchilov and Hutter, 2017) was used for optimization. The learning rate was 1e-4, β_1 was 0.9, $beta_2$ was 0.999 and L2 weight decay was 0.01. The learning rate was first warmed up for 100 steps and linearly decayed for the rest of the training. The model was trained for one epoch with a batch size of 32. The training took 25 minutes on a P100 GPU.

3.2 Evaluation on real words

We evaluate the model with the top-k accuracies of its predicted embeddings. Specifically, if the model's predicted vectors have the true embeddings as their closest k neighbors, the model's predictions are counted as correct.

The evaluation results are shown in Table 1. The overall top 1 accuracy of the Notch model's predicted embeddings is .67, the top 10 accuracy is .84. As the model's predictions are compared among the 10,000 candidates' word embeddings, the chance level of the top 1 and top 10 accuracy would be 1e-4 and 1e-3. The results indicate the model captures the relationships between the constituent characters and the corresponding word embeddings. The accuracies vary among different word lengths. The two-character word's accuracy is lower than the four-character one's (.63 vs. .75, respectively). As the Chinese four-character words are mostly idioms known for their semantic opaqueness, the pattern might initially seem counter-intuitive. However, a closer look into the word embeddings reveals that these four-character words are mostly coarse-grained words, which act like short-phrases. For example, 乘坐高鐵, which could be considered a two-word short phrase, 乘坐 chéng zuò "riding" and 高鐵 gāo tiě "high-speed rail".

Len.	N	Top 1	Top 5	Top 10
1	162	.73	.85	.86
2	2,522	.63	.78	.81
3	2,123	.66	.79	.84
4	3,375	.75	.87	.90
≥5	1,818	.57	.72	.77
All	10,000	.67	.80	.84

Table 1: Top-k accuracies when predicting word embeddings from the constituents. All top-k accuracies are calculated based on the whole testing set. That is, the chance levels are randomly guessing a word among the 10,000 words regardless of their word length.

To further explore the model predictions, Table 2 shows samples of the prediction errors. The error patterns indicate that the model predicts the *meanings* not only from the character semantics but the general word-level information. For example, the second one is a transparent two-character word, 脫除 tuō chú "get rid of"; the composing characters of which both have meanings related to removal. The model thus consistently predicts the words with related meanings. Moreover, the predictions may also be related to word-level properties. For instance, the third example is the name of green tea, the meaning of

	Targets	Predictions	Translations
1	司仪 (16)	钦差; 刘大人; 侍立	*emcee: imperial commisioner; Lord Liu; wait upon*
2	脱除 (19)	去除, 消退, 卸去	*get rid of: discard; fade away; remove*
3	碧螺春 (17)	关汉卿, 庐山恋, 孝庄秘史	*bi-luo-chun (a green tea): Guan Hanqing, a Chinese playwright; Romance on Lushan Mountain, a Chinese movie; Xiaozhuang Epic, a TV drama*
4	观看中 (15)	正片中, 表演中, 故事当中	*watch-ing: in the film; during the performance; in the story*
5	社交距离 (16)	彼此了解, 交流能力, 像朋友一样	*social distancing: mutual understanding; communication skills; (be) like friends*
6	釜底抽薪 (14)	抢功, 破罐子破摔, 千夫所指	*solve the root of the problem (an idiom): take credit; (totally) giving up; blamed by everyone*

Table 2: The prediction errors of the Notch model. The numbers in the brackets are the location of the targets in the predictions; that is, the number 1 indicates the prediction is correct as the Top-1 prediction. The loose translations are provided in the last columns, where the words before the colons are target words, followed by three predictions separated by semicolons.

which is not decomposable to its constituents. However, while not semantically correct, the model's predictions are also other proper playwright or movie names. Similarly, the model does not capture the word meaning for a fully opaque term (e.g., the last one), but the predictions are related to other opaque idioms.

While the model's accuracies might suggest the words' semantics (word embeddings) could be almost determined from their constituents, there are caveats in this interpretation. On the one hand, the model inputs are character tokens, and the model has an embedding layer in its first layer. Therefore the model could learn or tune character embeddings to predict the final word embeddings. In this sense, the model does compute the final word embeddings based on the constituent embeddings. On the other hand, the BERT's transformer architecture keeps mixing and warping the constituent embeddings in its 12 encoding layers (Lee-Thorp et al., 2021; Tolstikhin et al., 2021; Mai et al., 2022); therefore, the final output of the `[CLS]` is no longer a simple linear combination of their constituents. That is, the constituent embeddings are themselves changing based on extra-constituent information; thus, it is context-dependent (Baggio et al., 2012). Finally, we are using a distributed semantic model to operationalize word semantics, the predictability may be the characteristic of the vector semantic model rather than word semantics itself.

Nonetheless, it is still interesting to ask what the Notch model captures among these constituents. Having demonstrated the Notch model could predict the word embeddings in real words, we next examine whether the model could predict the embeddings of novel constituent combinations, i.e., pseudowords. As pseudowords, by definition, do not occur in our linguistic uses, they cannot readily be given word embeddings to be compared to model predictions. Therefore, we turn to behavioral data in psycholinguistics to evaluate the model's prediction of pseudowords.

3.3 Evaluation on Pseudowords

In this experiment, we try to evaluate the Notch model by how the predicted embeddings shed light on the behavioral data of pseudowords. Although word recognition has been well studied, pseudowords in psycholinguistics, especially in lexical decision tasks (LDT), are experimental stimuli which experimenters have little interest in. However, regardless of real words or pseudowords, the information is accumulated over time for both words and pseudowords in an LDT task (Yap et al., 2015; Ratcliff et al., 2004). That is, pseudowords are not *devoid of meaning* (Hendrix and Sun, 2021; Chuang et al., 2021). Moreover, the behavioral data of the pseudowords provide unique insights into lexical processes when the word-level information does not yet influence them.

Therefore, we derive two semantic indices from the Notch model and use statistical mod-

els to examine their effects on pseudoword processing. First, we identify the 50 nearest semantic neighbors of a pseudoword based on the model predicted vectors. The neighbors are selected among the whole 500K words in the dataset. Next, we create a vector whose elements are the 50 neighbor distances in ascending order, π_w, where w is the given pseudoword. Basing on the vector π_w, we calculate the average distance of the closest 5 neighbors, `smSimTop5`, and the difference of distances between the .90 and .10 quantiles, `smSimRange`. Specifically, the `smSimTop5` indicates how close the pseudoword is to a real word, and `smSimRange` captures how densely populated the pseudoword located in the word embedding space. The smaller the `smSimRange`, the more densely-packed the neighbors are in a given (hyper-)sphere; the larger the value, the more sparsely-populated the pseudoword is located.

We use the pseudoword data from the MELD-SCH dataset (Tsang et al., 2018). The dataset contains lexical decision data on 25,156 words from 504 native Chinese speakers. There are equal numbers of real words and pseudowords in the dataset, i.e., 12,578 words for each word type. Each of these word types has 1,020 one-character, 10,022 two-character, 949 three-character, and 587 four-character words. The dataset also includes additional character-level information for each word, e.g., number of strokes, character frequency, and number of meanings. This additional information serves as the statistical model's controlled variables, or the semantic indices' effects may be proxies or surrogates for other character-level effects. The inclusion of character-level variables also implies a single statistical model could not accommodate words of different lengths as they require a different number of variables. Therefore, we select two-character words as they are the most commonly occurred words in the MELD-SCH and the Tencent word embedding dataset.

We include 12 explanatory variables on this analysis, which are two semantic indices derived from the Notch model, namely `smSimTop5` and `smSimRange`; 10 character level indices, that is, character frequency for first and second constituents (`C1logcf`, `C2logcf`), number of strokes (`C1stroke`, `C2stroke`), number of words formed(`C1lognwf`, `C2lognwf`), number of meanings(`C1nom`, `C2nom`), number of pronunciations (`C1nop`, `C2nop`). These variables are used to predict error rates and response time in their respective models [3].

The statistical results are shown in Figure 1. The left panel shows the variable importance of 12 variables with respect to error rates and response times. The importance scores are estimated by the "mean decrease in accuracy" following the permutation principle in a 100-tree random forest (Hothorn and Zeileis, 2015). The figure shows the Notch-derived semantic index, `smSimRange`, is among the most important features in both error rate and response time models, along with the number of the word formed in second constituents `C2lognwf`. The other semantic index, `smSimTop5`, however, is the fourth most important feature in error rates but is the seventh one in response time.

A closer look at the variable effects with the generalized additive model (GAM) (Wood, 2011) also shows consistent patterns. [4] Here, we include the semantic indices and other important variables, namely the number of words formed and character frequency in the GAM models. All included variables are highly significant in the models. In particular, the partial effects shown in the center and right panels of Figure 1 indicate a nearly positive linear effect of both `smSimRange` and `smSimTop5`. The patterns suggest that the Notch-derived indices, especially `smSimRange`, help explain the behavioral error rates and response times. Specifically, when the pseudowords are in a less populated area (large `smSimRange` values), the participants tend to respond slower

[3] Due to their distribution characteristics, the character frequency, and error rates, number of meanings, and number of pronunciations are log-transformed. Response time is reciprocally transformed and multiplied by -1000 to keep the same sign and direction.

[4] Two GAM models have nearly the same explanatory variables, only RT model has an additional `C1stroke` due to its importance in the random forest analysis. `smSimRange` and `smSimTop5` are included as thin plate regression spline smoothing terms. Other pairwise character-level information is included as tensor-product smoothing terms. The worst concuvrities of the semantic indices are .13 and .09. The Pearson correlation between `smSimRange` and `smSimTop5` is .20

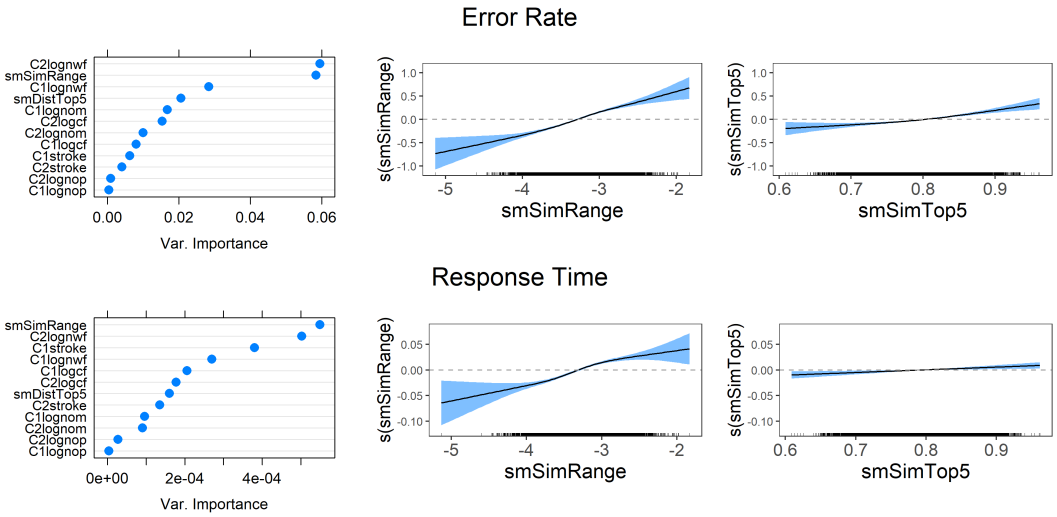

Figure 1: The statistical results of the Notch-model derived semantic indices: `smSimDist` and `SimRange`. The left panel shows the variable importance. The center and right panels show the partial effects estimated by GAM.

and make more errors. Similarly, when the pseudowords have similar real word neighbors (larger `smSimTop5`), the responses tend to be slow and more error-prone. Although the underlying lexical process is inevitably more complicated, the results already demonstrate the relevance of the Notch-model derived semantic indices.

After real words and pseudowords evaluations, we established the Notch model can capture relationships between the constituent and word embeddings. Such capacity would suggest the model may learn the representations of how a character would function as a constituent in a word. Therefore, we further probe into the model to examine its character-level representations.

4 Character Jacobian

In this section, we extract and evaluate the character-level information in the Notch model. Being a model based on BERT, the Notch model learns the contextualized embeddings of each token in its input sequence. The characteristic has been applied to the word sense disambiguation task, where the model is successfully employed to create sense embeddings (Loureiro and Jorge, 2019; Scarlini et al., 2020). Therefore, an interesting question to ask is, how could we extract the character-level context-dependent information from the Notch model?

Specifically, this character-level information should ideally differentiate the word contexts the character occurs. For example, the character 手 shǒu means "hand" as an independent word. However, the same character could occur at the start of the word and carries different meanings, such as 手氣 shǒu qì "luck" and 手臂 shǒu bì "arm". In addition, the meanings are also different when occurring at the end of the word: 歌手 gē shǒu "singer", or 分手 fēn shǒu "break up". That is, the character-level information should have different representations for the same character when it is used differently.

There are at least three different approaches to extract character-level information from the Notch model. The first one is to use the BERT input embeddings directly. Since the `bert-base-chinese` is a by-character tokenizer, the input embeddings act like character embeddings. However, this approach does not have access to other characters in the same word; therefore, it is not context-sensitive as we require. The second approach uses the token embeddings in the later layers of the BERT model, with the advantage that the token embeddings would be context-sensitive after layers of transformation. Yet, it also implies that the token embeddings have already incorporated or mixed the represen-

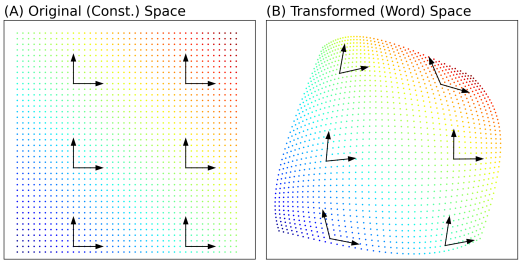

Figure 2: A visualization of the Jacobian matrix. The rectangular grid (left) is transformed into a warped mesh (right) by a non-linear function. The Jacobian matrix describes how the grid is transformed locally, as shown by the black arrows.

tations from other tokens. Hence, it is not straightforward to attribute the token to the input character anymore. The third approach, which is simultaneously context-sensitive and character-specific, is through the Jacobian matrix.

4.1 Jacobian matrix

The Jacobian matrix is a matrix whose elements are the first-order partial derivatives of a vector-valued function:

$$\nabla F(X) = \begin{bmatrix} \frac{\partial F_1(X)}{\partial x_1} & \cdots & \frac{\partial F_1(X)}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial F_m(X)}{\partial x_1} & \cdots & \frac{\partial F_m(X)}{\partial x_n} \end{bmatrix}$$

As the Notch model's inputs are discrete characters, we need first to convert the input characters to vectors to compute the Jacobian. Therefore, we leverage the input embeddings in the BERT embedding layer and convert the characters to vectors without introducing additional parameters. The converted embeddings are from the raw embeddings that are not yet involved with positional and sequence type encodings. The converted vectors are then used as the model input. Therefore, the Notch model could be considered as a function that brings the input vectors from the space of $R^{768 \cdot k}$, where k is the word length, to the word embedding space of R^{768}.

The Jacobian of the input characters could be considered as the linear projection matrix that best approximates the non-linear transformation at a specific location in space. Informally, it describes how a slight perturbation of the character's input vector *nudges* the word vectors in the embedding space. Figure 2 shows a simple illustration of a non-linear function transforming a rectangular grid into a warped mesh. Note that in the figure, although the transformation is non-linear, each transformed arrows are still tangent to its transformed grid. The Jacobian provides a way to describe how each black arrow are warped in its specific location.

Moreover, as the function (i.e., the Notch model) is context-sensitive to its input characters, the Jacobian automatically encodes the context in which the character occurs. In addition, the Jacobian is the first derivative of the character embeddings. We can directly attribute the matrix to that character. Thus, the Jacobian matrix satisfies the requirements of character-level information.

The downside of this approach would be that obtaining Jacobian matrices is relatively computationally expensive. As opposed to the input embedding and token embedding approach, which only requires a vector of 768 dimensions (i.e., the model dimension) for each character in each context, the Jacobian approach will require a 100×768 matrix and additional steps to compute. However, the issue should be alleviated with the advancement of algorithms (Baydin et al., 2022) and the hardware.

In the following evaluation, we only focus on the two-character word hence the character Jacobian, $J^{(c_i)}$, is defined by the Jacobian, $\nabla F(c_i)$:

$$J^{(c_i)} \triangleq \nabla F(c_i) = \frac{\partial F(\nu_1, \nu_2)}{\partial \nu_i}$$

where ν_1 and ν_2 are the corresponding input vectors.

4.2 Evaluation on affixoids clustering

In this evaluation, we aim to examine to what extent the character Jacobians captures the characters' role in the words. The dataset we used is the "Common Affixoids Database" compiled by the CKIP group at Academia Sinica, Taiwan. The dataset includes different roles of a character, consisting of prefixes,

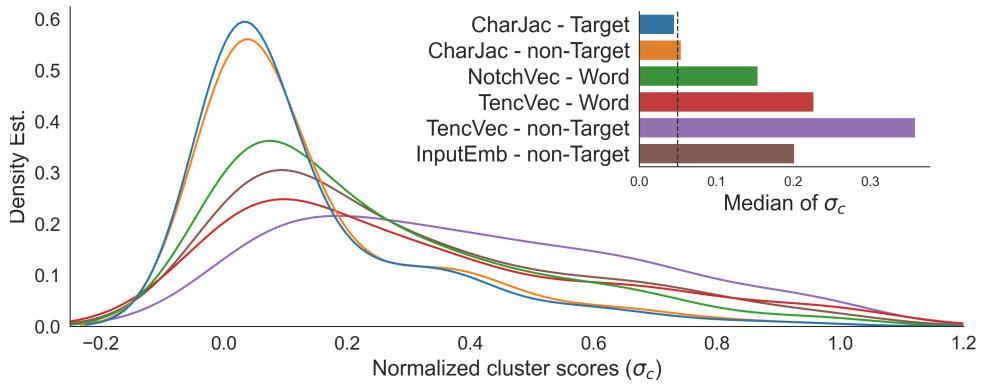

Figure 3: The distribution of normalized cluster scores, σ_c under different conditions. The inset figure shows the median of σ_c. The dashed line marks the position of .05 for the visual reference.

suffixes, or morphological roots. [5]. Characters with different roles or different meanings will have separate entries. There are 2,471 unique characters and 4,893 entries in the dataset.

Among these entries, we first identified a set of characters that have more than one entry in the dataset. For example, 土 had one entry indicating land or clay, such as 土石 tǔ shí "earth and stones" or 土堤 tǔ tí "embankment"; also, it had another entry indicating "native or local", such as 土狗 tǔ gǒu "native dog" or 土著 tǔ zhù "indigenous people". We extracted at least two instances (i.e., example words) for each entry and at most five instances whose word frequencies were larger than one. As a result, we selected 796 unique characters and 1,765 entries. Among these entries, there are 7,072 instances.

For each character, we compute the character Jacobians of each instance in different entries. For example, 土 has two entries, and each entry has two instances, then we compute four character Jacobians for each instance. These Jacobians are then compared to each other to obtain a distance measure. Here we use the L1-norm. If the character Jacobians indeed capture the characters' roles in different instances, the distances between character Jacobians of the same entry should be closer to those from other entries. That is, we could evaluate the character Jacobians by measuring the clustering performance implied by their pairwise distances. Here, we assess the clustering with the averaged silhouette score (Rousseeuw, 1987).

Furthermore, to establish a reference, we build a null distribution for each averaged silhouette score with random permutations. We randomly permute the instance labels 1,000 times for each character and compute one silhouette score. That is, we calculate the same averaged silhouette score as if the meanings of the characters no longer group the example words. These permuted scores will form a null distribution to which the silhouette score will be compared. Finally, we compute normalized cluster scores σ_c for each character to indicate the clustering performance. The normalized cluster scores are defined by $1 - P_{\text{null}}(X)$, that is, the probability of obtaining the values higher than the observed silhouette scores assuming the null hypothesis is true. A lower σ_c would indicate the corresponding silhouette score is less likely to result from the random chance, hence, the better clustering.

In addition to the character Jacobian of the target character (`CharJac-Target`, e.g. 土 in 土石), we also include five conditions for comparison. The `CharJac-nonTarget` refers to the character Jacobian of the character in non-target position (e.g. 石 in 土石). This condition show the roles of the other character in the same word. Next, the `NotchVec-Word` and `TencVec-Word` are computing the same silhouette scores but using the word vectors from the Notch predictions and Tencent embeddings respectively. These conditions provide the baseline for word-level semantics. Finally, the `TencVec-nonTarget` and `InputEmb-nonTarget` both compute the

[5]The dataset is publicly available at `http://turing.iis.sinica.edu.tw/affix`

scores based on the embeddings of the character at non-target position. The former one uses the single-character word embeddings from Tencent dataset, while the later one uses the input embeddings from the first layer of Notch model. These two conditions serve as the baselines for the information the non-target character could provide in clustering.

The results are shown in Figure 3. It can be seen the two conditions of character Jacobian have distinct distributions compared to other conditions. The inset plot in Figure 3 further shows the median of σ_c for each condition. It is apparent that the character Jacobians, regardless of the target or non-target position, could form the clusters better, as indicated by the lower values compared to other conditions. The comparison across conditions additionally reveal that the clustering results cannot be achieved by the word-level semantic (the `NotchVec-Word` and `TencVec-Word` condition), or by considering the character as single-character alone (the `TencVec-nonTarget` and `InputEmb-nonTarget`). The results demonstrate that character Jacobian captures an important aspect of the character's role in the word.

5 Conclusion

In this paper, we present the Notch model, from which we derive the character Jacobians. In a series of experiments, we show that the model predicts the embeddings of the real words from their constituents and helps account for the behavioral data of the pseudowords. In addition, we also show that character Jacobians capture characters' roles in the words, which reflect the meanings of the characters.

The approach to study compounding by modeling the word embeddings and exploring their Jacobians could also be applied to other languages. Multilingual language models and word embeddings are readily available in the community. However, the most interesting question is how to study compounding if the language's writing system may introduce spaces in the compounds, such as the case in English. The commonly-used word embeddings only include entries with no interword spaces. That is, there will be entries for *earthquake*, *airport*, but no entries for *rush hour*, *coffee mug*. Moreover, it could be argued that the interword spaces bear significance in cognitive processing (Juhasz et al., 2005). It will be thus an interesting future work to systematically study compounding in this case.

Character Jacobians open up new possibilities to study Chinese characters or morphology with deep learning models. However, as the Jacobian is an abstract mathematical object, other future works include further investigating its relation with morphological rules and lexical categories and how it connects to the distributional semantics.

Acknowledgements

This study was supported by Ministry of Education, Taiwan, Grant Number 110L9A001; and Ministry of Science and Technology (MOST), Taiwan, Grant Number MOST 110-2634-F-001-011.

References

Giosuè Baggio, Michiel Van Lambalgen, and Peter Hagoort. 2012. The processing consequences of compositionality. In *The Oxford handbook of compositionality*, pages 655–672. Oxford University Press.

Laurie Bauer. 2009. Typology of compounds. In *The Oxford handbook of compounding*.

Atılım Güneş Baydin, Barak A. Pearlmutter, Don Syme, Frank Wood, and Philip Torr. 2022. Gradients without backpropagation.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Yu-Ying Chuang, Marie Lenka Vollmer, Elnaz Shafaei-Bajestan, Susanne Gahl, Peter Hendrix, and R Harald Baayen. 2021. The processing of pseudoword form and meaning in production and comprehension: A computational modeling approach using linear discriminative learning. *Behavior research methods*, 53(3):945–976.

CKIP. 1998. *Academia Sinica Balanced Corpus: Content and description (Technical Report No.95-02/98-04)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Fritz Günther and Marco Marelli. 2021. Caoss and transcendence: Modeling role-dependent constituent meanings in compounds. *Morphology*, pages 1–24.

Peter Hendrix and Ching Chu Sun. 2021. A word or two about nonwords: Frequency, semantic neighborhood density, and orthography-to-semantics consistency effects for nonwords in the lexical decision task. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 47(1):157.

Rumjahn Hoosain. 1992. Psychological reality of the word in chinese. In *Advances in psychology*, volume 90, pages 111–130. Elsevier.

Torsten Hothorn and Achim Zeileis. 2015. partykit: A modular toolkit for recursive partytioning in R. *Journal of Machine Learning Research*, 16:3905–3909.

Shu-Kai Hsieh, Yu-Hsiang Tseng, Chih-Yao Lee, and Chiung-Yu Chiang. 2018. Fluid annotation: A granularity-aware annotation tool for Chinese word fluidity. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Chu-Ren Huang, Hsieh Shu-Kai, and Chen Keh-Jiann. 2017. *Mandarin Chinese words and parts of speech: A corpus-based study*. Routledge.

Ray Jackendoff. 2002. *Foundations of language: Brain, meaning, grammar, evolution*. Oxford University Press, USA.

Barbara J Juhasz, Albrecht W Inhoff, and Keith Rayner. 2005. The role of interword spaces in the processing of english compound words. *Language and cognitive processes*, 20(1-2):291–316.

James Lee-Thorp, Joshua Ainslie, Ilya Eckstein, and Santiago Ontanon. 2021. Fnet: Mixing tokens with fourier transforms.

Gary Libben. 2014. The nature of compounds: A psychocentric perspective. *Cognitive neuropsychology*, 31(1-2):8–25.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization.

Daniel Loureiro and Alípio Jorge. 2019. Language modelling makes sense: Propagating representations through WordNet for full-coverage word sense disambiguation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5682–5691, Florence, Italy. Association for Computational Linguistics.

Florian Mai, Arnaud Pannatier, Fabio Fehr, Haolin Chen, Francois Marelli, Francois Fleuret, and James Henderson. 2022. Hypermixer: An mlp-based green ai alternative to transformers.

Jerome L. Packard. 2000. *The Morphology of Chinese: A Linguistic and Cognitive Approach*. Cambridge University Press.

Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. 2016. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE.

Ingo Plag. 2003. *Word-formation in English*. Cambridge University Press.

Roger Ratcliff, Pablo Gomez, and Gail McKoon. 2004. A diffusion model account of the lexical decision task. *Psychological review*, 111(1):159.

Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.

Bianca Scarlini, Tommaso Pasini, and Roberto Navigli. 2020. SensEmBERT: Context-Enhanced Sense Embeddings for Multilingual Word Sense Disambiguation. In *Proceedings of the Thirty-Fourth Conference on Artificial Intelligence*, pages 8758–8765. Association for the Advancement of Artificial Intelligence.

Yan Song, Shuming Shi, Jing Li, and Haisong Zhang. 2018. Directional skip-gram: Explicitly distinguishing left and right context for word embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 175–180, New Orleans, Louisiana. Association for Computational Linguistics.

Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. 2021. Mlp-mixer: An all-mlp architecture for vision.

Yiu-Kei Tsang, Jian Huang, Ming Lui, Mingfeng Xue, Yin-Wah Fiona Chan, Suiping Wang, and Hsuan-Chih Chen. 2018. Meld-sch: A megastudy of lexical decision in simplified chinese. *Behavior research methods*, 50(5):1763–1777.

Yu-Hsiang Tseng, Shu-Kai Hsieh, Pei-Yi Chen, et al. 2020. Computational modeling of affixoid behavior in chinese morphology. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2879–2888.

Shengjie Wang, Abdel-rahman Mohamed, Rich Caruana, Jeff Bilmes, Matthai Plilipose, Matthew Richardson, Krzysztof Geras, Gregor Urban, and Ozlem Aslan. 2016. Analysis of deep neural networks with extended data jacobian matrix. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 718–726, New York, New York, USA. PMLR.

S. N. Wood. 2011. Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models. *Journal of the Royal Statistical Society (B)*, 73(1):3–36.

Melvin J Yap, Daragh E Sibley, David A Balota, Roger Ratcliff, and Jay Rueckl. 2015. Responding to nonwords in the lexical decision task: Insights from the english lexicon project. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 41(3):597.