Nano Backpack Language Model on Chinese Characters

Stanford CS224N Custom Project

Hao Sun
Stanford University
haosun@stanford.edu

Abstract

This project focuses on the Backpack Language Model proposed in the paper "Backpack Language Models" [1], which aims to improve lexical representations and non-contextual controllability in language modeling while maintaining contextual performance. To facilitate comprehension by beginners, we reproduced the Backpack model in a NanoGPT-styled [2] implementation and pretrained it on a character-based Chinese corpus. Our implementation achieves comparable or superior performance to a matching-sized GPT2 [3][4] model on perplexity and word prediction accuracy tasks, demonstrating the potential of the Backpack model in character-based languages. We further evaluate the lexical interpretability of sense vectors and experiment with basic character-level interventions for controllable generations.

1 Key Information to include

- Mentor: John Hewitt

2 Introduction

Language modeling is a critical task in natural language processing, where the objective is to compute the probability of the next word in a sequence given the preceding words. Large language models based on the Transformer architecture have recently achieved remarkable success in various NLP applications, such as text generation, machine translation, and question-answering. However, these models rely on contextualized word embeddings that may result in non-linear predictions, making it difficult to intervene predictably in all contexts. To address this issue, the authors of the *Backpack Language Models* [1] paper proposed a novel neural architecture, Backpack, for which the predictions are linear combinations of non-contextual representations called sense vectors while still expressing rich contextual information. Although the Backpack model has shown promising results in pretraining tasks, the training-optimized code base may pose challenges for beginners to comprehend or modify. Moreover, the model's effectiveness in languages with richer morphological structures than English remains uncertain due to difficulties interpreting and controlling individual tokens without stable semantics.

Our project presents a reproduction of the Backpack language model using an implementation inspired by NanoGPT [2], which supports pretraining, fine-tuning, and generation tasks to enhance the model's accessibility for beginners. We trained several Backpack and GPT2 [3] baseline models and evaluated them on perplexity and word prediction accuracy tasks. Our experiments demonstrate that our pretrained Backpack Language Model using character-based tokenization performs comparably or even better than a similarly sized GPT2 model. Furthermore, we propose and evaluate character-level interventions to mitigate gender bias and control lexical centroids, which exhibit promising results for generating controllable text in character-based Chinese language, despite the infrequent usage of single characters' meanings in the modern Chinese language.

3 Related Work

3.1 Word Representation with Deep Learning before Backpack

Numerous word embedding techniques have been proposed in the early stages of natural language processing with deep learning, including Word2Vec [5] and GLoVe [6], which represent words as numerical vectors. Word2Vec is a neural network-based method that learns word embeddings by predicting the probability of a word's occurrence given its context words or predicting the context words given a central word. It has been shown to be a Backpack by the authors of the Backpack paper [1]. In contrast, GLoVe creates a co-occurrence matrix for all pairs of words in a corpus and then factorizes this matrix using matrix factorization techniques. These methods produce high-quality word representations that capture the semantic and syntactic relationships between words but fail to perform well on language modeling tasks. Subsequently, modern language models with Transformer architecture [7] use contextualized word embeddings that can be automatically learned from pretraining and fine-tuning tasks. However, these models involve non-linearity in the prediction, making it difficult for word embeddings to directly represent non-contextual semantic information and challenging to achieve predictable intervention across all contexts.

3.2 Language Modeling with deep learning before Backpack

Language Modeling is a foundational task in Natural Language Processing, which involves computing the probability of the next word in a sequence, given the previous words. Early approaches to language modeling employed different structures of Recurrent Neural Networks (RNNs) [8] and attention mechanisms [9]. More recently, modern language models have adopted the Transformer architecture [7], with the GPT series [3] by OpenAI achieving notable success in generating high-quality and coherent text. The success has led to applications in various areas, such as article generation and chatbots. Nevertheless, as previously discussed, interpreting word embeddings in Transformer-based language models proves to be a challenge.

3.3 Backpack

In the Backpack paper [1], the authors proposed a novel neural architecture called "Backpacks," which achieves high performance on contextualization and non-contextual word representations. This approach represents each word in a sequence as a linear combination of sense vectors, with weights computed by an expressive network such as the Transformer. The linearity of the contributions of sense vectors to predictions encourages the sense vectors to specialize and encode rich notions of word meaning during pretraining. Furthermore, the authors conducted experiments on sense vectors, demonstrating their potential for predictable control across all contexts. We reproduced this model in an implementation styled after NanoGPT [2], and pretrained it on character-based Chinese language, thereby showing promise for applying the Backpack model to languages of this type.

4 Approach

4.1 Architecture

4.1.1 Introduction

Two hierarchic implementations were abstracted: the *Backpack Model* with abstract layers that represent a Backpack general form, and the *Backpack Language Model* that defines the language modeling. We utilized and modified the Transformer block, layer normalization, GELU, and pretraining implementation from nanoGPT codebase [2] and the character-based Chinese tokenizer used in the project GPT2-Chinese [4].

4.1.2 Backpack

A Backpack model is a probabilistic model $p(\mathbf{y}|\mathbf{o}_{1:n}) = \text{softmax}(E_{\mathbf{o}_{1:n}})$ where $y \in \mathcal{Y}$ and $E : \mathbb{R}^{d \times n} \to \mathbb{R}^{|\mathcal{Y}|}$ is a linear transformation. It learns k sense vectors $C(\mathbf{x})_1, \cdots, C(\mathbf{x})_k$ for each word $\mathbf{x} \in \mathcal{V}$ to encode rich non-contextual meanings, and represent each word in a sequence as a linear combination of all the sense vectors in the context in a weighted sum function $\mathbf{o}_i =$

$\sum_{j=1}^{n} \sum_{\ell=1}^{k} \alpha_{\ell ij} C(\mathbf{x_j})_\ell$ where $\alpha_{\ell ij}$ is defined by a contextualization function $\alpha = A(\mathbf{x}_{1:n})$, and $A : \mathcal{V}^n \rightarrow \mathbb{R}^{k \times n \times n}$. In our implementation, the backpack model was defined with the following abstract neural layers: 1) a **sense vector layer** that introduces a sense function that converts the inputs to sense vectors, 2) a **contextualization layer** that introduces a weight function to calculate the contextualization weights, and output the weighted sum for each word in the sequence, and 3) a **logit layer** that introduces a logit function for the probabilistic model output.

4.1.3 Backpack Language Modeling

A Backpack language model is a probabilistic model $p(\mathbf{x}_j|\mathbf{x}_{1:j-1}) = \text{softmax}(E^\top \mathbf{o}_j)$ where linear weight matrix $E \in \mathbb{R}^{d \times |\mathcal{V}|}$ maps $\mathbf{o}_j \in \mathbb{R}^d$ to logits $E^\top \mathbf{o}_j \in \mathbb{R}^{|\mathcal{V}|}$. The sense function was parameterized $C(x) = \text{FF}(Ex)$ where FF: $\mathbb{R}^d \rightarrow \mathbb{R}^{d \times k}$ is a a feed-forward network, and contextualization weights $A(\mathbf{x}_{1:n}) = \alpha$ where $\alpha_\ell = \text{softmax}(\mathbf{h}_{1:n}^\top K^{(\ell)\top} Q^{(\ell)} \mathbf{h}_{1:n})$ for each predictive sense ℓ with matrices $K^{(\ell)}, Q^{(\ell)} \in \mathbb{R}^{d \times d/k}$ and $\mathbf{h}_{1:n}$ calculated by a Transformer with proper autogressive masking, i.e. $\mathbf{h}_{1:n} = Transformer(E\mathbf{x}_{1:n})$. These parameterizations are the same as the original paper [1]. The abstract layers defined in the Backpack model were inherited and completed in our implementation. Following are several details: 1) The feed-forward network was defined by parsing a feed-forward layer with residual connection and layer norm to dimension $4d$ and back to d, and secondly to dimension $4d$ and up to $k * d$. We started our experiment with no second residual connection. However, it was added later by unsqueezing the output from the first feed-forward layer by dimension k to match $k * d$ dimensions for better training stability. 2) The contextualization weight function was defined with mask filling and an extra dropout layer included after the Softmax function. 3) In order to ensure a fair comparison with the corresponding GPT2 model, one block was removed from the Transformer structure of the Backpack model. This modification was taken because a comparable number of parameters were utilized in the contextualization layer and the first feed-forward layer in the sense function.

4.2 Baselines

We employ a GPT2 model [3] as a baseline, pretrained using the same datasets, hyperparameters, and random seed as our Backpack model. The GPT2 and Backpack models have equal contextual parameters in the Transformer structure, whereas the Backpack model contains additional non-contextual parameters for the sense vectors. Furthermore, we adopt the Chinese GPT2-small Model [4] from Huggingface as a baseline for our 124M Backpack-small. The GPT2 and Backpack models share the same tokenizer and have an identical embedding size and the same number of layers and heads in the Transformer structure.

5 Experiment Training Backpack LMs

5.1 Data

For pretraining, we employed three corpora: wiki2019zh [10], news2016zh [10], and webtext2019zh [10], which are composed of 1.04 million Wikipedia entries, 2.5 million news articles, and 4.1 million Q&As, respectively, resulting in a total dataset size of 14.3G. To prepare the data, we set aside 1% of the data for the test set and 0.5% for the development set. The data was processed according to the methodology employed in NanoGPT [2], randomly partitioned into blocks of size 1024 for each training step on each GPU using a constant random seed. In addition, Shakespeare dataset [11] was used for debugging our implementation.

To evaluate our models on contextualization performances, we employed the Chinese WPLC [12] dev set, which includes 4,827 test cases and is utilized for assessing top-1 word accuracy in word prediction with long-term context.

5.2 Evaluation method

To evaluate the contextual performance of the Backpack and GPT2 baseline models, we computed perplexities on a 1% subset of the retained corpus from the datasets. We also utilized the Chinese WPLC dev set [12] to evaluate the models' ability to contextualize and predict words accurately.

Specifically, each test case comprised a long sentence with at least 150 Chinese characters, with the last significant word being masked and having a length of 2 to 4 characters. The objective of the task was to predict the masked word, and we evaluated the performance of the models based on their top-1 and top-3 accuracy. As this task was originally tested on both masked language models and casual language models, we designed two methods to evaluate our autoregressive models properly: The first approach involved generating a precise number of characters based on the length of the masked tokens and selecting the top predictions via beam search, while the second approach involved generating characters until the length of the output tokens equaled the length of the original sentence. We retained ten generations from the beam in every step, penalized the outputs by adding the log probability of the original ending characters, and then selected the top generations. While the first method was deemed unfair to autoregressive language models as they could generate words with the incorrect length or tend to continue the sentence, the second approach was more comparable to the masked language model.

5.3 Experimental details

The pretraining process for the 30M Backpack-micro and the GPT2-micro baseline models involved training on 3$\times$ RTX3090 GPUs, using a batch size of 184,320 tokens for 500,000 gradient steps with cross-entropy loss, the AdamW optimizer, 2,000 warmup steps, and linear decay on the learning rate starting from 6e-4. The model with the best performance on the dev set was retained by evaluating at intervals of 1000 steps. The Transformer structure comprised six layers, six heads, and an embedding size of 384, with dropout disabled for efficient attention in torch 2.0. Three attempts were made to fix the structure of the Backpack language model during pretraining. In the first experiment, gradient explosion occurred without the second layer norm and the residual connection. In the second experiment, a layer norm was erroneously added to the logit layer, breaking strict linearity. In the final attempt, one layer was removed from the contextualization layer of the Transformer structure to match the size of the corresponding GPT2 model. The 30M Backpack-micro with 16 sense vectors was trained for 2.5 days, the Backpack-micro with 64 sense vectors was trained for 5 days, and the GPT2 baseline was trained for 1.5 days. A 124M Backpack-small model was further pretrained on 4$\times$ RTX3090 GPUs for 4 days, with a batch size of 245,760 tokens for 250,000 gradient steps, using 16 sense vectors and a Transformer structure comprising 12 layers, 12 heads, and an embedding size of 768, to match the Huggingface Chinese GPT2 [4] baseline.

5.4 Results

During the experiment, it was observed that pretraining the Backpack model was more challenging to stabilize compared to the baseline model, although the overall loss curve of the 16-sense vector Backpack LM was similar to the baseline. It was also challenging to stabilize the pretraining for the Backpack model with 64 sense vectors until full convergence, so we additionally evaluated the micro models at their halfway point, i.e., 250,000 steps. It was discovered that adding an extra GPU led to an increase of 61,440 tokens per batch, resulting in a more stable pretraining process. Unfortunately, due to time constraints, repeating the pretraining on the micro models was not feasible.

In general, the Backpack models with 16 sense vectors performed comparably or slightly better than the same-sized GPT2 model regarding perplexity score and significantly better regarding WCPC accuracy (Table 5.4). Besides, it was intuitive that a character-based Backpack LM would require more sense vectors than a word-based one to represent word composition information and richer morphological structures. Our findings showed that the Backpack-micro with 64 sense vectors performed significantly better on both tasks than the one with 16 sense vectors.

WCPC is a challenging evaluation task as it requires the model to have long-distance contextualization ability and some world knowledge to determine the masked word. For the WCPC score, we also found that our 124M Backpack-small performed on tied with ALBERT-xxlarge [13] on top-1 accuracy and outperformed all BERT family baselines [14][15][16] on top-3 accuracy using the ending words penalizing strategy. However, when compared by the direct prediction strategy, it performed much worse than the casual language model PanGu-α [17] with 200 billion parameters. Curiously, we found that the WCPC accuracy goes down with the direct prediction strategy on both Backpack LM and GPT2 from 250K steps to 500k steps. Based on our intuition, we hypothesize that with increased training steps, the model can learn more complex patterns or word compositions and, as a result, predicts more complex words that contain more characters. However, since the strategy does not

allow the language models to foresee the future patterns in the target sentence, this may lead to poorer performance in predicting the fix-sized masked tokens.

Model	Senses	Steps	PPL ↓	WCPC top-1 ACC ↑	WCPC top-3 ACC ↑
Backpack-micro	16	500K	16.25	1.93% / 2.98%	3.56% / 7.46%
GPT2-micro	-	500K	16.66	1.26% / 2.44%	2.55% / 5.51%
Backpack-micro	16	250K	17.46	1.76% / 2.92%	4.04% / 7.31%
Backpack-micro	64	250K	15.10	1.96% / 3.07%	4.18% / 8.08%
GPT2-micro	-	250K	17.83	1.57% / 2.36%	3.17% / 5.43%
Backpack-small	16	250K	**10.74**	2.94% / 4.45%	6.61% / **11.83%**
GPT2-small (HF)	-	-	12.60	2.18% / 3.46%	4.91% / 9.41%
Pangu-α	-	-	-	12.7% / -	- / -
BERT-base	-	-	-	- / 7.3%	- / 10.1%
RoBERTa-base	-	-	-	- / 6.5%	- / 9.8%
MacBERT-large	-	-	-	- / 6.8%	- / 10.6%
ALBERT-xxlarge	-	-	-	- / 4.5%	- / 6.5%

Table 1: Language modeling performance; GPT2-small is from Huggingface. The baseline WCPC accuracies are from the original paper. For perplexity, lower is better; for accuracy, higher is better.

6 Analysis

6.1 Sense Vectors

6.1.1 Visualizing Senses

Following the Backpack paper, we projected the sense vectors of characters onto the vocabulary, denoted as $EC(\mathbf{x})_\ell \in \mathbb{R}^{|\mathcal{V}|}$, to illustrate the contribution of the sense vectors towards predictions. The outcomes are in the appendix (see Table 6). As hypothesized, specific sense vectors captured word composition rules, whereas others captured semantic relatedness or associations.

6.1.2 Word Representations

In character-based languages, words are constructed through one or several characters in a complex manner. For instance, in Chinese, some words are composed of characters with sub-meanings (Type I), while some borrowed words from foreign languages only use the pronunciations of the characters (Type II). There are also four-character words that represent lengthy allusions, with the characters representing the critical objects in the allusion (Type III).

Two methods were explored to represent composed words using the sense vectors of the constituent characters. The first method involved computing the average value of the sense vectors of the constituent characters to represent the word's sense vector. The second method involved representing the word's sense vector by an approximate weighted sum of the sense vectors of the constituent characters. This method was motivated by the intuition that a stably composed word's sense vector should maintain a *consistent ratio* across all contexts. To prove the feasibility of the second method, we designed several prompts (Appendix A) that fit different types of words and calculate the average contribution ratio of each character's sense vectors among all constituent characters in the word and how much each contribution is away from the average value. Our experimental results showed that each character's contribution ratio in a word on each sense vector for prediction remained stable across various contexts. Furthermore, the stability of word compositions was observed to follow the order of Type III > Type II > Type I, as expected. Sample results are presented in Table 6.1.2. For example, the word "新闻" (news) is a Type I word composed of the characters "新" (new) and "闻" (hear). We observed that the differences in weights assigned to the two characters varied by no more than $\pm 10\%$ for 49% of the senses across all prompts. Moreover, only 7% of the senses showed a variation exceeding $\pm 20\%$ in the contribution ratio.

Word	Type	$\leq \pm 10\%$	$\leq \pm 20\%$	$\geq \pm 20\%$
新闻 (news) = 新 (new) + 闻 (hear)	I	49%	43%	8%
沙发 (sofa) = 沙 (so-) + 发 (fa)	II	65%	26%	9%
齐天大圣 (Monkey King: Great Sage, Equal of Heaven) = 齐 (equal) + 天 (heaven) + 大 (great) + 圣 (sage)	III	86%	13%	1%

Table 2: How the contribution ratio of sense vectors on characters of a word varies among the different contexts. A more minor variation in the contribution ratio indicates a more stable word composition.

6.1.3 Lexical Relationship Test

We evaluated the lexical relationship of the sense vectors using two datasets: Wordsim-240[18] and Wordsim-297[18]. Based on the discussion in the last subsection, to represent a word using the sense vectors of multiple characters, we proposed two methods: (1) averaging all the sense vectors of the constituent characters and (2) combining sense vectors by using the contextual weights of the contextualization function when the constituent characters are solely inputted. To assess the quality of the resulting lexical representations, we computed Pearson product-moment correlation coefficients between the relationship scores in the datasets and the cosine similarities of each word pair across all the sense vectors of our models. For the GPT2 model, we represented each word by averaging the embeddings of the constituent characters.

Our results in table 6.1.3 show that using the average of the sense-2 vectors of the characters outperforms contextually weighted sense vectors on both datasets. Both methods in our Backpack Model outperformed the same-sized GPT2 model, but the results were significantly inferior to word embeddings trained *directly on words* using methods such as word2vec[5] or GLoVe[6].

Representation	Wordsim-240	Wordsim-297
Backpack-micro 16 senses #2 (AVG)	0.318	0.395
Backpack-micro 16 senses #2 (CTX)	0.252	0.338
Backpack-micro 64 senses (250K steps) #61 (AVG)	0.333	0.444
Backpack-micro 64 senses (250K steps) #61 (CTX)	0.257	0.318
GPT2-micro	0.178	0.275
Backpack-small 16 senses (250K steps) #2 (AVG)	**0.379**	**0.491**
Backpack-small 16 senses (250K steps) #2 (CTX)	0.324	0.364
GPT2-small (Huggingface)	0.313	0.437
CBOW (word-tokenized)	0.561	0.626
GloVe (word-tokenized)	0.558	0.584

Table 3: Pearson product-moment correlation coefficients between the provided scores and the cosine similarities of the word pairs are calculated. Character-tokenized Backpack LMs outperform GPT2 but are inferior to word-tokenized models.

6.1.4 Four-Character Idiom Composition

We attempted to investigate which sense vectors played a dominant role when the model used the first three characters of idiomatic phrases as input to predict the last character. However, we encountered difficulty in interpreting the character composition of idiomatic phrases. For example, when analyzing the phrase "画蛇添(足)" i.e., "drawing legs on a snake," which means "an unnecessary and redundant act that spoils the original effect or even makes it worse," by stacking weights of the first three characters on 16 or 64 sense vectors, we found that using any single sense vector for prediction did not significantly lead the model to output the target character, even though the model correctly outputted "足" i.e., "leg" after performing a weighted sum of these sense vectors. We projected 16 sense vectors onto the vocabulary and examined their projections onto the character; however, we observed that none exhibited a disproportionately large or small projection onto the resulting character.

6.2 Sense Vectors for Control

In this section, we showcase two character-level interventions on the sense vectors as proof-of-concept.

6.2.1 Mitigating gender bias

In Modern Chinese, most professions are composed of two or more Chinese characters, making direct debiasing of stereotypically gendered profession nouns difficult. To address this issue, we attempted two approaches: 1) identifying the characters within the composed words that contain gender-biased meanings and debiasing them from their sense vectors, and 2) directly debiasing the sense vectors of the composed words using the method discussed in Word Representations.

We hypothesized that the first approach could be practical because many Modern Chinese words are combined from ancient single-character words that represent a relevant meaning to the composed words. For example, the word "士兵" (soldier) is composed of "士" (man/warrior) and "兵" (arms), both of which carry stereotypical male bias. In our experiments, we attempted to identify the sense vectors of characters that contain gender stereotypes and compared $abs(EC(\mathbf{x_{he}})\ell - EC(\mathbf{x_{she}})\ell)$ to determine which sense vectors contribute to gender bias. We found that sense 3 contributed the most bias. Using the method described in the Backpack paper, we reduced the weight of sense 3 on these characters. We evaluated how the composed words were gender debiased by creating several prompts (Appendix A) that fit all the profession words, filling in the target word, and computing the average bias probability score of "他 (he/him)" versus "她 (she/her)" as $\mathbb{E}_{X \in prompts}[max(\frac{p(he|\mathbf{x})}{p(she|\mathbf{x})}, \frac{p(she|\mathbf{x})}{p(he|\mathbf{x})})]$.

Baseline. We employed a similar approach as described in the Backpack paper, which was inspired by the work of Bolukbasi et al. (2016)[19]. Specifically, we computed the gender bias direction using the difference between the embeddings of the words "他 (he/him)" and "她 (she/her)," denoted as $E\mathbf{X_{he}} - E\mathbf{X_{she}}$, and then projected the embeddings of the biased characters onto the nullspace of this direction.

Results. We experimented with investigating the effect of removing sense 3 from several characters on bias scores of profession words containing those characters. The bias ratios resulting from this experiment are reported in the table. 6.2.1. Our experimentation demonstrated that removing sense 3 substantially decreased the bias in words that were originally more biased while producing a considerably lesser impact on words with lower levels of bias. Nonetheless, this approach yielded significant improvements compared to the GPT2 baseline. Additionally, we confirmed the observation made by the authors of the Backpack paper that the original Backpack model exhibited lower bias than the original GPT2 model across all instances in the experiment.

Character	Target Word	GPT2	GPT2 proj	Backpack	half #3	remove #3
兵 (arms)	士兵 (soldier)	521.90	519.20	27.31	21.02	16.40
兵 (arms)	伞兵 (paratrooper)	12.87	12.84	8.76	7.70	6.78
警 (alert)	警察 (police)	18.60	18.44	5.80	4.77	3.94
警 (alert)	交警 (traffic police)	14.95	14.91	4.55	3.54	2.75
洁 (clean)	保洁 (cleaner)	3.29	3.13	1.78	1.69	1.64
教 (teach)	教师 (teacher)	4.14	4.14	2.82	2.69	2.57

Table 4: Character-level bias ratio; by partially or totally removing sense 3, the character and the words composed by the character get debiased. A perfect unbiased model would achieve a ratio of 1.

Besides, we explored the second approach by removing sense 3 for both constituent characters. Surprisingly, this approach was less effective than the first approach. To investigate whether there exists a specific sense vector to remove for all characters in all compositional words for gender debiasing, we experimented and observed that reducing sense 10 significantly reduced the bias in the word 警察 (police); however, the reducing sense 10 method did not generalize to other words. We hypothesize that the model might not effectively learn the gender-representing information due to the limited model size and pretraining steps. Some critical gender-related information might still distribute among several sense vectors.

6.2.2 Lexical Centroid Control

Focusing on sub-meanings or properties in a word constructed by multiple characters makes more sense in character-based languages. For instance, the Chinese word "词典" which means "dictionary," is composed of the characters "词" (word) and "典" (book, in ancient Chinese), and when generating text from input containing this word, the model could focus on either the "word" or "book" property. By adjusting the weights of the sense vectors of the constituent characters, we were able to shift the lexical centroid and bias the model toward generating text related to a specific property. Specifically, we conducted experiments to amplify the contribution of the first or second character four times each while keeping the total contribution of the word unchanged in the output. We found that the model tended to generate sentences that relate to the amplified character with greater probability, as shown in Appendix A. We assessed the efficacy of the proposed method by computing the ratio of expectations for the controlled model relative to an uncontrolled model in the context of predicting semantically related characters from an open-topic prompt as $\mathbb{E}_{\mathbf{c}_{target}}[\frac{p(\mathbf{c}_{target}|\mathbf{x}_{amp})}{p(\mathbf{c}_{target}|\mathbf{x})}]$. Table 6.2.2 illustrates an instance of the outcome of amplifying characters in the word "沙滩" (sandy beach). Notably, the findings indicate that character-specific semantics were the most amplified. We hypothesize that this work can assist in scenarios where it is necessary to precisely generate expressions that convey the author's intended meaning in a short sequence, such as poetry, songwriting, or beginning a discourse around one of the meanings in a polysemous word.

AMP	丘 (dune)	撒 (sanding)	堡 (castle)	粒 (particle)	石 (stone)	海 (sea)	人 (people)	球 (ball)	晒 (bask)
1,1	1	1	1	1	1	1	1	1	1
4,1	2.72	2.48	2.18	1.58	1.26	0.71	0.83	0.77	0.28
1,4	0.46	0.57	0.58	0.81	0.96	1.02	1.11	1.17	2.52

Table 5: Alterations in character prediction expectations following amplification of sense vectors for 沙 (sand) and 滩 (beach)

7 Conclusion and Limitations

In this project, we presented implementing, pretraining, and evaluating a character-based Chinese language model using the NanoGPT-styled Backpack LM architecture. We conducted extensive experiments on sense vector visualizations, word representations, lexical relationships, and idiom compositions and explored two approaches to character-level interventions. Our results demonstrate the potential of Backpack LM in language modeling tasks for character-based languages, the interpretability of the sense vectors on the character and word level, and the potential of character-level interventions across various contexts.

Despite these promising results, there are several limitations to our study. First, we had limited GPU resources, which prevented us from attempting a larger batch size during pretraining. We later found that adding one more GPU significantly improved the stability of the model. Second, our word interventions depend on the sub-meanings of the characters, and we currently have no solution to effectively intervene in transliterated words by modifying the sense vectors of the characters that only represent phonetic information. Therefore, intervening in character-based languages where many words are transliterated, such as Korean, remains challenging. Third, we observed that although our approach enables greater flexibility in character-level sense vectors to represent richer morphological structures, word representations by characters are less interpretable than word sense vectors learned by models using word tokenizations, particularly for complex words such as idiomatic phrases. We believe that this issue could be mitigated by increasing the number of sense vectors in larger contextualization models and using more pretraining data. Further research is required to address these limitations and explore the potential of word representations and interventions in character-based languages.

References

[1] Anonymous (in Feb 2023). Backpack language models. In *Association for Computational Linguistics (ACL)*, Unpublished.

[2] karpathy. nanogpt. https://github.com/karpathy/nanoGPT.

[3] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[4] Zeyao Du. Gpt2-chinese: Tools for training gpt2 model in chinese language. https://github.com/Morizeyao/GPT2-Chinese, 2019.

[5] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[6] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[8] Ilya Sutskever, James Martens, and Geoffrey Hinton. Generating text with recurrent neural networks. pages 1017–1024, 01 2011.

[9] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[10] Bright Xu. Nlp chinese corpus: Large scale chinese corpus for nlp, September 2019.

[11] karpathy. char-rnn. https://github.com/karpathy/char-rnn.

[12] Huibin Ge, Chenxi Sun, Deyi Xiong, and Qun Liu. Chinese WPLC: A Chinese dataset for evaluating pretrained language models on word prediction given long-range context. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3770–3778, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[13] Liang Xu, Hai Hu, Xuanwei Zhang, Lu Li, Chenjie Cao, Yudong Li, Yechen Xu, Kai Sun, Dian Yu, Cong Yu, Yin Tian, Qianqian Dong, Weitang Liu, Bo Shi, Yiming Cui, Junyi Li, Jun Zeng, Rongzhao Wang, Weijian Xie, Yanting Li, Yina Patterson, Zuoyu Tian, Yiwen Zhang, He Zhou, Shaoweihua Liu, Zhe Zhao, Qipeng Zhao, Cong Yue, Xinrui Zhang, Zhengliang Yang, Kyle Richardson, and Zhenzhong Lan. CLUE: A Chinese language understanding evaluation benchmark. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4762–4772, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018. cite arxiv:1810.04805Comment: 13 pages.

[15] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019. cite arxiv:1907.11692.

[16] Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang, and Guoping Hu. Revisiting pre-trained models for Chinese natural language processing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 657–668, Online, November 2020. Association for Computational Linguistics.

[17] Wei Zeng, Xiaozhe Ren, Teng Su, Hui Wang, Yi Liao, Zhiwei Wang, Xin Jiang, ZhenZhang Yang, Kaisheng Wang, Xiaoda Zhang, Chen Li, Ziyan Gong, Yifan Yao, Xinjing Huang, Jun Wang, Jianfeng Yu, Qi Guo, Yue Yu, Yan Zhang, and Yonghong Tian. Pangu-α: Large-scale autoregressive pretrained chinese language models with auto-parallel computation. 04 2021.

[18] Yilin Niu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. Improved word representation learning with sememes. In *Annual Meeting of the Association for Computational Linguistics*, 2017.

[19] Tolga Bolukbasi, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam Tauman Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *NIPS*, 2016.

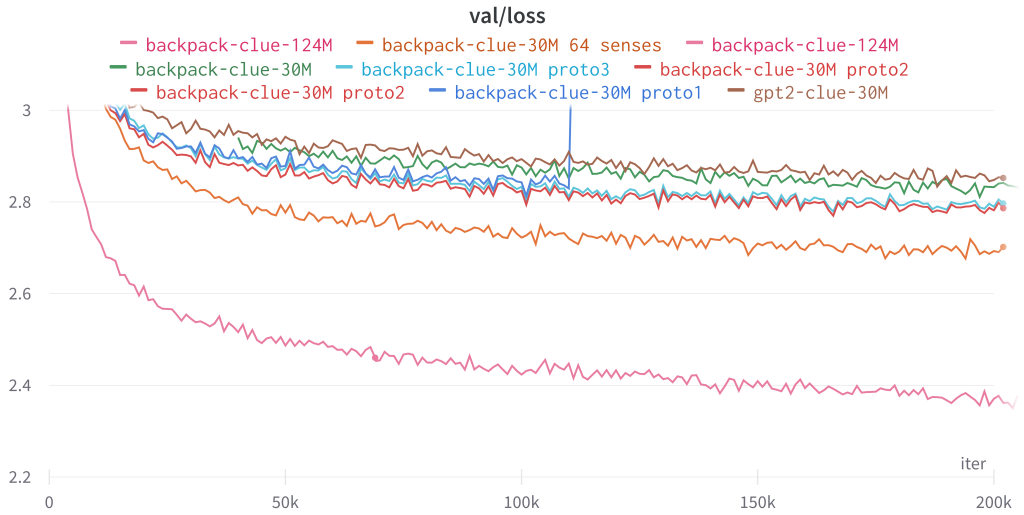

Figure 1: Losses on the dev set during pretraining

A Appendix

Sense Vector 8 (*Word Composition*)

天 (sky / day)	地 (ground / land)	沙 (sand)	哲 (wise / sagacious)
(天)然 (natural)	(地)毯 (carpet)	(沙)漠 (desert)	(哲)蚌 (Drepung monastery)
(天)鹅 (swan)	(地)址 (address)	(沙)棘 (sea-buckthorn)	(哲)学 (philosophy)
(天)蝎 (Scorpion)	(地)铁 (subway)	(沙)滩 (sandy beach)	(哲)理 (philosophic theory)
(天)津 (Tianjin)	(地)狱 (hell)	(沙)甸 (sardine fish)	(哲)𝔰 (*internet slang*)
(天)使 (angel)	(地)方 (place)	(沙)俄 (Tsarist Russia)	(哲)学 (philosophy)

Sense Vector 13 (*Meaning Relatedness*)

天 (sky / day)	地 (ground / land)	沙 (sand)	哲 (wise / sagacious)
疲 (tired)	耕 (plow)	群 (plenty)	氏 (Mr. / Mrs.)
宅 (home)	旱 (dry)	礁 (reef)	悖 (rebel / paradox)
泡 (bubble)	稼 (crops)	尘 (dust)	批 (criticize)
遛 (dog walking)	墅 (villa)	沉 (seek)	抽 (extraction / abstract)
腻 (be bored)	农 (farm)	霎 (instant)	畴 (area)

Sense Vector 7 (*High-level Meaning Relatedness*)

天 (sky / day)	地 (ground / land)	沙 (sand)	哲 (wise / sagacious)
午 (noon)	瞿 (thoroughfare)	04 (沙尔克04 / FC Schalke 04)	婚 (marry)
凌 (approach)	挚 (fondle)	钛 (titanium)	皆 (all)
傍 (side)	姜 (ginger)	剂 (dose)	身 (body)
早 (morning)	滚 (roll)	蹴 (kick)	佤 (Va ethnic group)
晚 (night)	溴 (Bromine)	郝 (*family name*)	死 (death)

Table 6: The sense vectors in the same index are considered to have a particular facet of character usage. Each column contains the characters with the highest scores under the projection of the sense vectors on the vocabulary.

prompt	English
WORD	WORD
有很多WORD将要	There are a lot of WORD to be
WORD被	WORD is
每天都有WORD	There are WORD everyday to be
电视里经常能看到WORD	You can often see WORD on TV
不管你喜不喜欢，WORD还是那么	Whether you like it or not, WORD is still so
有哪位同学解释一下WORD是什么意思	Can anyone explain what WORD means?

Table 7: General prompts for different type of nouns

prompt	English
那个WORD说，	That WORD said,
这个WORD相信	This WORD believes
WORD进到屋子里，	The WORD enters the house,
WORD坐在车里，然后	The WORD sat in the car, and then
WORD走了过来，	Then WORD came over,

Table 8: General prompts for gender bias evaluations

Word	AMP	Output
沙滩 (sandy beach) 沙(sand) 滩(beach)	1,1	这片沙滩非常好看，有些人看到这个沙滩就会想到... (This beach is very beautiful, some people will think of ... when they see this beach)
沙滩 (sandy beach) 沙(**sand**) 滩(beach)	4,1	这片沙滩非常好看，沙滩上有一个小小的沙堆 (This beach is very beautiful, there is a **small pile of sand** on the beach)
沙滩 (sandy beach) 沙(sand) 滩(**beach**)	1,4	这片沙滩非常好看，有些人看到这个片子，就会想到这里 (This beach is very beautiful, some people will think of it when they see this **film**)
人间 (world) 人(people) 间(between)	1,1	在这人间里，有多少人是真正的朋友 (In this world, how many people are true friends)
人间 (world) 人(**people**) 间(between)	4,1	在这人间里，有多少人是真正的自由自在 (In this world, how many people are truly **free**)
人间 (world) 人(people) 间(**between**)	1,4	在这人间里，有多少人在一起 (In this world, how many people are **together**)

Table 9: Generative outputs on the lexical centroid control task