Inducing Character-level Structure in Subword-based Language Models with Type-level Interchange Intervention Training

Jing Huang[1] **Zhengxuan Wu**[1] **Kyle Mahowald**[2] **Christopher Potts**[1]

[1]Stanford University [2]The University of Texas at Austin

Abstract

Language tasks involving character-level manipulations (e.g., spelling corrections, arithmetic operations, word games) are challenging for models operating on subword units. To address this, we develop a causal intervention framework to learn robust and interpretable character representations inside subword-based language models. Our method treats each character as a typed variable in a causal model and learns such causal structures by adapting the interchange intervention training method of Geiger et al. (2022b). We additionally introduce a suite of character-level tasks that systematically vary in their dependence on meaning and sequence-level context. While character-level models still perform best on purely form-based tasks like string reversal, our method outperforms character-level models on more complex tasks that blend form, meaning, and context, such as spelling correction in context and word search games. Compared with standard subword-based models, our approach also significantly improves robustness on unseen token sequences and leads to human-interpretable internal representations of characters.

1 Introduction

Many common natural language tasks can fruitfully be described in terms of character-level manipulations. For instance, we resolve spelling mistakes with character-level edits, we perform unit conversions by moving decimal points and changing specific digits, and we play language games that center around anagrams, word reversals, character transpositions, and other operations on characters.

For some of these tasks, the best models may be ones that tokenize their inputs and outputs at the character level. Such models likely have the best chance of learning character-level concepts and operations. However, with only a few exceptions (Xue et al., 2022; Tay et al., 2022; Clark et al., 2022), our best general-purpose models at present do not tokenize their inputs into characters, but rather into words and subword units (Liu et al., 2019; Brown et al., 2020; Raffel et al., 2020; He et al., 2021; Black et al., 2022; Scao et al., 2022; Zhang et al., 2022). There is thus a tension between solving character-level tasks and developing task-agnostic solutions.

1. Character Reversal
`txpraa` ⇒ `aarpxt`

2. Unit Conversion
`convert 1.23 m to cm` ⇒ `123`

3. Unscramble
`tkneti` ⇒ `kitten`

4. Single Word Spelling Correction
`misspellde` ⇒ `misspelled`

5. Spelling Correction with Context
`the actuall name` ⇒ `the actual name`
`was actuall happy` ⇒ `was actually happy`

6. Word Search
`color: augustmacaronihsilgneerg` ⇒ `green`
`a written or spoken language: augustmacaronihsilgneerg` ⇒ `english`

Figure 1: Core tasks. System inputs are in green, outputs in blue. The tasks are all form-based and differ in the extent to which they depend on meaning and context.

In this paper, we develop a causal intervention-based framework for pushing subword-based models to encode character-level information in their internal representations, in effect teaching them which characters their tokens contain. The techniques are based on the interchange intervention training (IIT) method of Geiger et al. (2022b), which trains neural hidden representations to correspond to variables in a high-level causal model capturing aspects of the task domain. We apply IIT at the level of variable *types* (Type-level IIT), which allows us to learn robust, position-independent representations of characters in the hidden states of subword-based models. We compare against ap-

Findings of the Association for Computational Linguistics: ACL 2023, pages 12163–12180

proaches that tokenize inputs and/or outputs at the character level.

We introduce a suite of character-level evaluation tasks (Figure 1). All of these tasks depend heavily on character-level manipulation of forms, but they differ in terms of how much they (a) involve meaning and (b) depend on the full context of the input string (see Table 1). We find that, for tasks involving only meaning or only context (tasks 1–4), pure character-level modeling is superior. However, for the more challenging and intricate tasks that involve both meaning and context (tasks 5 and 6), subword tokenization models prove superior. Our Type-level IIT pushes these subword models to represent characters internally, which leads to the best overall models. Finally, we show that Type-level IIT leads to subword-based models with human-interpretable internal representations of characters.[1]

2 Related Work

2.1 Subword and Character Modeling

Subword-based models tokenize their inputs into words and word pieces, most of which are longer than individual characters. The most prominent subword tokenization methods are byte-pair encoding (Sennrich et al., 2016), word-piece tokenization (Schuster and Nakajima, 2012), and unigram language models (Kudo, 2018; Bostrom and Durrett, 2020). These methods have become standard for large pre-trained language models (Liu et al., 2019; Brown et al., 2020; Raffel et al., 2020).

Character-level models, by contrast, represent inputs as character sequences. These methods have generally not been as widely employed for large language models; the token sequences are much longer, which introduces significantly higher costs for both training and inference (Libovický et al., 2021; Mielke et al., 2021; Pinter, 2021). However, a few recent character-level large language models have proven highly successful on standard benchmarks (Xue et al. 2022; Tay et al. 2022; Clark et al. 2022; see also Dos Santos and Zadrozny 2014; Belinkov and Bisk 2018; Rosales Núñez et al. 2021).

Another line of research has sought to create hybrid character-level and subword (or word) models (Luong and Manning, 2016; Ma and Hovy, 2016; Pinter et al., 2017; Peters et al., 2018; Schick and Schütze, 2019; Aguilar et al., 2021). These methods typically modify the input layer, define additional weights to learn character embeddings, and construct character-to-word mappings.

2.2 Character Manipulation Tasks

Character manipulation tasks such as word scrambling and basic arithmetic are increasingly prominent in large language model evaluations (Brown et al., 2020; Wei et al., 2022). In addition, a number of recent efforts have focused on linguistic phenomena that depend, at least in part, on character-level manipulations. Examples include digit tokenization (Geva et al., 2020), creative blends like 'hangry' (Pinter et al., 2021), puns (Yu et al., 2020; Mittal et al., 2022), and the wordplay involved in crossword puzzle clues (Efrat et al., 2021; Rozner et al., 2021; Wallace et al., 2022).

These studies tend to show that subword tokenization models do not fully encode information about the characters contained in their tokens. Itzhak and Levy (2022) test RoBERTa (Liu et al., 2019) on a spelling task that requires it to map from words to characters. RoBERTa can correctly spell more than one-third of tested words, which is striking given its byte-pair encoding scheme but still far from reliable. (Interestingly, CharacterBERT (El Boukkouri et al., 2020) is not notably better at the task.) Kaushal and Mahowald (2022) directly probe models to see whether they implement token-to-character mappings, finding that even the best subword models are wrong about 10% of the time about this conceptually simple relationship.

2.3 Intervention-Based Training Methods

Our core technique is based in the interchange intervention method (IIT) of Geiger et al. (2022b). With IIT, one can train a neural model to conform to a high-level causal model of some aspect of the task domain while still allowing it to learn from data as usual. IIT belongs to a family of causal abstraction techniques (Beckers and Halpern, 2019; Beckers et al., 2020) that have proven successful for obtaining human-interpretable explanations of complex neural networks (Geiger et al., 2021; Wu et al., 2022). The key innovation of IIT is to extend these explanation techniques to model training. For an overview of these methods and additional connections to the literature, see Geiger et al. 2022a.

[1]We release our dataset and code at https://github.com/explanare/char-iit.

Task name	Meaning	Context	Splits
Reversal	–	–	20/4/1K
Unit Conversion	–	✓	30/4/1K
Unscramble	✓	–	100/4/2K
Single Word SC	✓	–	100/4/4/6K
Contextual SC	✓	✓	100/5/4K
Word Search	✓	✓	90/1/5/6/4K

Table 1: Character manipulation tasks. "SC" stands for spelling correction. All tasks are form-based, but they vary in meaning and context aspects. Our task set covers all combinations of meaning and context. The splits are ordered by train/val/test. The test splits are "In-Vocab (IV)" and "Out-Of-Vocab (OOV)" for tasks 1–3 based on whether source tokens are seen in training; "IV", "OOV", and "Real" with natural spelling errors for Task 4; "Independent" and "Dependent" for Task 5 based on whether a correction is context dependent; "OOV", "O" with overlapped words, "P" with paraphrased definitions, and combined "O+P" for Task 6.

3 Character-level Manipulation Tasks

Our suite of tasks (Figure 1) is designed to test models along aspects of form, meaning, and context. We provide a loose categorization of each task in Table 1. All character manipulation tasks involve aspects of form. However, the roles for meaning and context vary by task. Our task set covers all combinations of values. We also test two variants of spelling correction that differ in the role of context. For evaluating the form aspect, we construct In-Vocab (IV) and Out-Of-Vocab (OOV) splits with the source tokens in or out of the training vocab. For evaluating meaning and context aspects, we construct task-specific test sets detailed below.

3.1 Character Reversal

The Character Reversal task is to reverse the characters contained in the input string (e.g., `txpraa` ⇒ `aarpxt`). The inputs and outputs do not need to be valid English words. Hence the task is form only, with no meaning or context involved.

3.2 Unit Conversion

The Unit Conversion task takes a decimal number, a source unit, and a target unit, and applies decimal shifting (multiplication or division by power of 10), as in `convert 1.23 m to cm` ⇒ `123`. The units are large number numerals ("million", "billion", and "trillion") or length units ("centimeter", "meter", and "kilometer"). The correct way to move the decimal point depends on the units, but the manipulation of digits itself is a mechanical, string-oriented process of moving a character. Hence we categorize the task as involving form and context, but not meaning. It is in principle possible for a model to find a semantic (truly arithmetic) solution to this task, but this is not necessary to solve it.

3.3 Unscramble

The Unscramble task takes a random permutation of a word and outputs the unscrambled word (e.g., `tkneti` ⇒ `kitten`). Unlike Brown et al. (2020), we do not constrain the first or last letter of the permutations. Unscrambling involves meaning, as models need to recognize the sequence of characters in the output as valid English words. We construct the dataset from 30K English words by randomly permuting letters in each word.

3.4 Single Word Spelling Correction

The Single Word Spelling Correction task takes a word with a spelling error and outputs the correct word (e.g., `misspellde` ⇒ `misspelled`). We follow the setup of Belinkov and Bisk (2018) to introduce four types of synthetic errors: swapping two adjacent characters, substituting a character with its neighbors on the keyboard, deleting a character, and repeating a character. Similar to the Unscramble task, spelling correction involves meaning because the correction needs to create an attested English word. We construct the dataset from 30K English words by adding synthetic errors to each word. We also evaluate on the real spelling errors collected by Belinkov and Bisk (2018).

3.5 Spelling Correction with Context

Spelling Correction with Context adds contextual aspects to the previous single-word spelling correction task. Context can be critical in spelling correction as some spelling errors have multiple potential corrections; as shown in Figure 1, the error in "actuall" can either be a repeat of the letter "l" or a deletion of the letter "y", and the correct choice depends on the surrounding context. We extract sentences from the Wikipedia corpus[2] as context and introduce the same spelling errors as in our Single Word task. The context length is capped at 64 characters. For test sets, our focus is the context "Dependent" condition, as in our "actuall" example. We also evaluate an "Independent" condition in

[2]We use the version pre-processed by HuggingFace at https://huggingface.co/datasets/wikipedia

which only one correction is valid. This trivializes the role of the context and thus brings us closer to Single Word Spelling Correction.

3.6 Word Search

Our Word Search task is adapted from the popular Word Search Puzzle,[3] in which players find hidden words in a letter grid matching a theme, such as colors or animals. The task involves relating the meaning of the letters to the theme, i.e., the context.

We generate synthetic puzzles with the structure `definition: letters`, where `letters` contains 24 characters. The task is to find in `letters` a substring that, when reversed, is defined by `definition`. We use reversed words to avoid the confound that subword tokenization trivially reveals forward words. We use definitions from WordNet Synsets (Miller, 1995) and a set of at least 5 hyponyms per Synset. The task assumes a fixed set of words per definition.

For training, we generate examples where the `letters` contains two reversed English words at random positions, with only one matching the definition. The rest of `letters` contains words in the forward direction. For instance, in Figure 1, `augustmacaroni`**`hsilgne`**`erg` embeds `green` at the end and `english` at the 4th to last position.

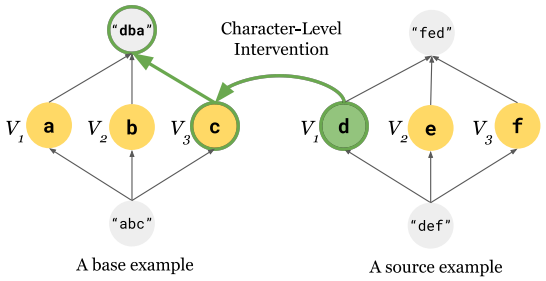
For test sets, we consider four variations: "OOV" with unseen tokenization of hidden words; "O" with the two backward words overlapped, as shown in our example above, which stress-tests the ability to recognize words; "P" with "paraphrased" definitions from *The Online Plain Text English Dictionary*,[4] testing the ability to understand context; "O+P" with both overlapped words and paraphrased definitions. Our expectation is that the "O+P" test scenario is the hardest in that it requires reasoning about the meaning of the full paraphrase and sophisticated character-level relations.

Unlike Task 5, where meaning and context mainly lie on the target side, Task 6 has context on the source side only, but meaning on both sides, allowing us to study the effects of subwords and characters on input/output.

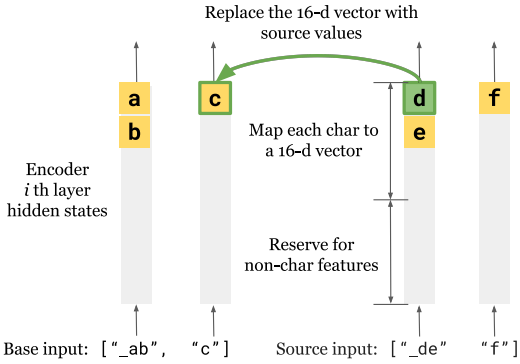
4 Character-level Interventions

Character-level inputs or outputs provide models with direct evidence for learning about characters as independent units or concepts. Subword tokenization methods do not provide such direct evidence, and it seems clear that these models do not reliably learn character-level concepts on their own (Section 2.2). In the current section, we present a method that seeks to address this shortcoming of subword models by training models to internally represent characters and the subword-to-character mappings needed for character manipulations.

Our core method for doing this is interchange intervention training (IIT; Geiger et al. 2022b). IIT has three steps. (1) Define a high-level causal model of the problem domain. (2) Align a variable V in that causal model with a set of neurons N in one's target neural model. (3) Train the neural model to make predictions according to the causal model using not only standard input–output pairs, but also *interchange interventions*: counterfactual instances created by replacing the values of N in a target example with those obtained by processing a distinct input, with the counterfactual label provided by the causal model from step (1). The effect



(a) Intervention on the high-level causal model for the Reversal task. The variable representing the last character of the word "abc" is set to the value of the variable representing the first character of the word "def". The effect of intervention is the output of the base example is changed from "cba" to "dba".



(b) Aligned intervention on the neural model. Each character in the subword is mapped sequentially to a 16d vector in Encoder i th layer hidden states, at the same step as the subword. As the hidden state dimension is 512 and the max number of character per subword token is 16, the last 256 dimensions are reserved for non-character features.

Figure 2: Intervention for the Reversal task.

[3] https://en.wikipedia.org/wiki/Word_search
[4] https://github.com/eddydn/DictionaryDatabase

of this process is to train the model to represent the variable V in the neurons N, which leads to modular internal structure for the network.

4.1 Causal Models for Characters

The first step for IIT is defining a high-level causal model. To illustrate this, we focus on the Character Reversal task and then sketch how the needed causal models can be defined for our other tasks.

Our causal model for Character Reversal is given in Figure 2a. The input to this model is a single string; for illustrative purposes only, we specify that the string has length 3. The model creates three intermediate variables V_1, V_2, and V_3, one per character, and outputs the values of those variables in reverse order, concatenated back into a string.

This causal model is fully deterministic, and so we know exactly what will happen if we intervene on one of the variables V_i to change its value to another character. For example, if the input is `abc` but we set $V_3 =$ `x`, then the model will output `xba`.

For IIT, we perform such interventions using pairs of examples, a base input (left) and a source input (right) as in Figure 2a. We then take the value created at our target variable V_3 in the source example and use it in place of the value of V_1 in the base example. In our example, this amounts to replacing `c` with `d`, leading to output `dba`.

In most prior work on IIT, these interventions target the same variable in the base and source. Such interventions are certainly useful, but they would instruct the model to learn both the character and its position, whereas our tasks depend on characters as unified concepts. Thus, we allow type-level interventions like the one described in Figure 2a: V_1 can take on the value of V_3 because both have the same type. Type-level IIT is briefly explored in Geiger et al. 2022b, where it is used to achieve similarly position-independent representations for handwritten images.

A similarly simple model can be defined for our other purely form-based task, Unit Conversion, which simply moves decimal places around based on the unit specified in the input string. For tasks involving meaning, the programs are somewhat more complex due to their dependence on English. For example, the Unscramble causal model forms intermediate representations of the characters in its input, as in Figure 2a, but the mapping to an output depends on a lexicon. The spelling correction and word search tasks are similarly constrained by a lexicon. However, the important common theme of all these programs is that they create character-level intermediate variables as the basis for their final output behavior.

4.2 Aligning the Causal and Neural Models

The second step for IIT is to define an alignment of variables in the causal model with sets of neurons in the target neural model. We again illustrate with the Character Reversal task. Figure 2b summarizes our alignment: the character variables V_1, V_2, and V_3 are mapped to the first-layer hidden states of the Transformer Encoder. Each character in the subword is mapped sequentially to a 16d vector of the hidden state, at the same step as the subword.

For form-only tasks such as Reversal, the choice of Encoder layer is less critical, as the Decoder alone is sufficient to handle the task logic. For semantic tasks, where the task logic is dependent on the character values, character variables are best mapped to early layers in the network.

4.3 Training with Character Interventions

The third and final step for IIT is model training. IIT objectives have two core parts: a standard training objective and a counterfactual objective. The standard objective simply uses the available train data in the usual fashion. The counterfactual objective additionally requires models to conform to the high-level causal model under interventions of the sort depicted in Figure 2b. These two loss components are weighted by coefficients λ_1 and λ_2. (For a technical description of the IIT objective, see Appendix A.2.)

This process can be thought of in two steps. First, we intervene on the causal model as in Figure 2a: given a base and a source example, we select a character variable V_b from the base and V_s from the source, in this case, variables representing the third and first characters. Our chosen intervention assigns the value of V_s to V_b, i.e., $V_b \leftarrow$ `d`. This leads to the output `dba`. This output will play the role of a train label.

Next, we intervene on the neural model as in Figure 2b. For this, we copy the 16d vector corresponding to V_s computed from the source input to the 16d vector corresponding to V_b computed from the base input, carrying the gradients with this vector for backpropagation. This leads the model to predict some output string s. Unlike with the causal model, we do not know a priori what s will be. However, comparing s with the output of our

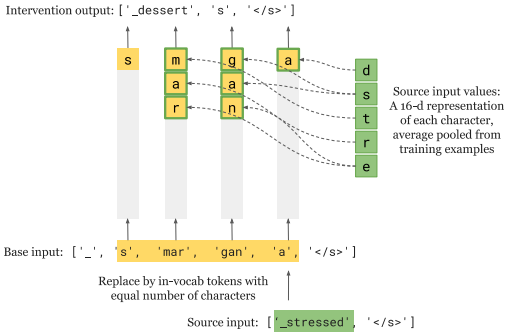


Figure 3: Resolving out-of-vocab inputs with interpretable character representations. To reverse the unseen token "stressed", we first replace the unseen token with random tokens seen in training. In this case, tokens from "smargana", the reverse of "anagrams". We then populate each position with an averaged representation of each character in the unseen vocab.

causal model (dba) gives us an error signal. The aggregate effect of these counterfactual updates is to push the model to localize information about the variables V_s and V_b in these aligned states.

4.4 Handling Out-of-Vocab Items

On its own, the above procedure does not provide a way to generalize to input tokens unseen in training. However, the interpretable character representations learned with Type-level IIT provide a natural solution. Figure 3 summarizes our approach. We first extract the 16d character representations from a set of training subword tokens and compute an averaged representation per character. Given an unseen subword token, we substitute the unseen token with seen tokens and populate representations of seen tokens with the averaged representation of each character in the unseen token. We show experimentally (Section 5) that this method leads to robust character manipulations over novel words.

5 Experiments

To evaluate how character, subword, and intervention-based models generalize with respect to form, meaning, and context, we experiment on the six character manipulation tasks in Figure 1.

5.1 Baselines

We consider three groups of tokenization approaches: (1) subword-based models (without IIT); (2) subword-based models with character-level input and/or output; and (3) character-level models. For (1), we fine-tune the pre-trained T5-small (Raffel et al., 2020).[5] We also experiment with in-context learning by prompting GPT-3 (Brown et al., 2020).[6] For (2), we simply change the tokenization of models in (1). For T5-small, we tokenize input and/or outputs into characters (for Unit Conversion, we only split digits and the decimal point). For GPT-3, we insert hyphen/space between characters in input and output. For (3), we fine-tune the pre-trained ByT5-small (Xue et al., 2022).[7] We choose T5/ByT5 for its Encoder–Decoder architecture.

For Tasks 5–6, we also consider context-only baselines to show that solving the task indeed requires form. For Task 5, we replace each typo with a mask token and fill with T5-small, which leads to 0% accuracy. For Task 6, we randomly select a word from the definition to words mapping, which has 9.4% accuracy on both "OOV" and "O" splits.

5.2 Intervention-based Models

We apply our character intervention-based method to the pre-trained T5-small model. The coefficients λ_1 and λ_2 for the base and IIT losses are set to 1.

5.3 Evaluation

We use the test sets described in Section 3 (Table 1).

For metrics, we use the sequence-level accuracy, i.e., the percentage of outputs that exactly match the label. For Unscramble, we allow anagrams of the label that are valid English words and non-identical to input. For Single Word Spelling Correction, we allow any valid English words that satisfy the synthetic error rules. We report average accuracy across runs.

For decoding, the T5/ByT5 models use greedy decoding. For IIT models, OOV splits are evaluated with the average-pooled character representations, computed from 2K randomly sampled training examples (see Section 4.4 for details).

5.4 Results

Table 2 presents our results for all our tasks, grouped by the informal typology in Table 1.

Our task suite reveals the accuracy trade-offs between subword and character models when generalizing with respect to form, meaning, and context. For form-based tasks (Tasks 1–2, Table 2a), pure character-level models (Char-ST and ByT5) achieve a clear win. As the meaning aspect is added to the output (Tasks 3–4, Table 2b), the best overall

[5] https://huggingface.co/t5-small
[6] GPT-3 davinci-003 engine, used in December 2022.
[7] https://huggingface.co/google/byt5-small

Method	Reversal IV	Reversal OOV	Unit Conversion IV	Unit Conversion OOV
Subword	49.29	0.25	86.84	65.00
+IIT	59.72	28.01	95.02	67.65
Char-T	99.15	5.73	99.58	69.29
+IIT	99.73	87.72	99.98	75.10
Char-S	53.42	17.94	94.07	79.08
Char-ST	**99.80**	97.26	**99.98**	**86.63**
ByT5	99.22	**99.09**	99.68	84.39
GPT-3	46.40	45.00*	84.20	94.20*
GPT-3-C	75.80	73.40*	56.20	58.80*

(a) Tasks without significant meaning components. Reversal is not contextual whereas Unit Conversion is.

Method	Unscramble IV	Unscramble OOV	Spelling Correction IV	Spelling Correction OOV	Spelling Correction Real
Subword	97.80	2.91	69.29	63.21	44.85
+IIT	98.97	72.63	77.02	63.91	51.79
Char-T	92.29	3.67	71.11	23.00	30.08
+IIT	96.17	69.98	76.74	70.14	38.46
Char-S	**99.46**	72.96	**78.54**	**82.08**	**55.59**
Char-ST	97.68	71.21	74.62	77.37	25.70
ByT5	99.19	**74.71**	76.14	80.08	31.32
GPT-3	50.80	38.20*	78.80	78.40*	73.00
GPT-3-C	16.20	14.00*	64.00	71.00*	63.80

(b) Tasks with significant meaning components but no contextual modulation.

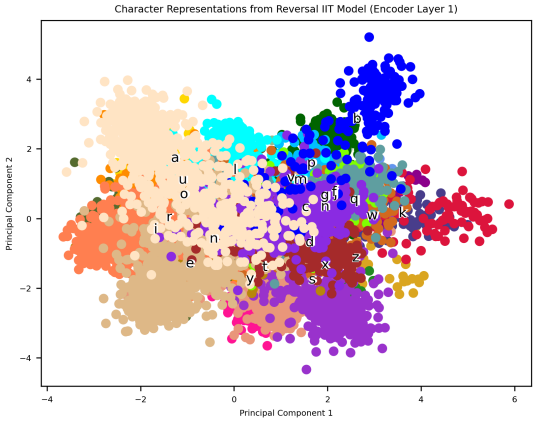
Method	Independent	Dependent
Subword	58.11	36.59
+IIT	67.00	**46.55**
Char-T	69.25	35.00
Char-S	**73.49**	45.26
Char-ST	69.50	30.98
ByT5	72.88	33.66
GPT-3	87.00	78.40

(c) Spelling Correction with Context, with significant form and meaning components. The "Dependent" split shows significant contextual effects, while the context "Independent" split does not.

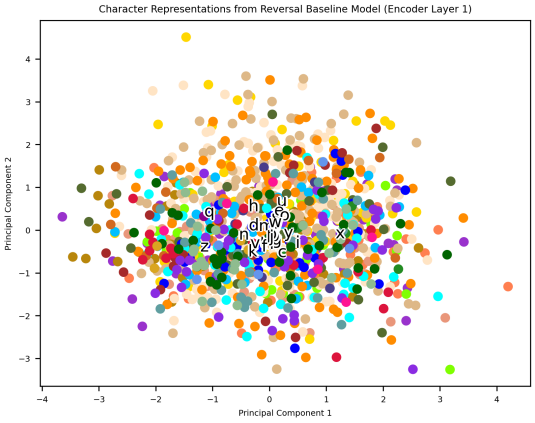
Method	OOV	O	P	O+P
Subword	24.07	93.65	71.17	64.13
+IIT	61.27	**94.27**	72.19	**64.82**
Char-T	6.73	73.18	74.89	50.52
Char-S	**85.74**	91.79	62.70	51.11
Char-ST	56.18	57.08	73.11	42.67
ByT5	68.62	72.67	**84.06**	57.52
GPT-3	60.00*	75.61	48.54	47.14

(d) Word Search with significant form and meaning components. "OOV": Hidden words with unseen tokenization; "O": Overlapping hidden words; "P": Paraphrased definitions; "O+P": Both overlapping words and paraphrased definitions.

Table 2: Sequence-level accuracy, with best non-GPT results in bold. "Subword": T5 subword model. "+IIT": Joint training with character-level interventions. "Char-T": T5 with character-level target sequences. "Char-S": T5 with character-level source sequences. 'Char-ST": T5 with character-level source and target sequences. "ByT5": ByT5 character model. "GPT-3": GPT-3 `davinci-003`. "GPT-3-C": GPT-3 with hyphen or space separated characters in source and target. *For GPT-3, the IV vs. OOV distinction is tricky, since the subword vocab is different from the T5 one.



(a) Character representations from model trained with character-level interventions. Characters corresponding to the same letter share similar representations.



(b) Representations extracted at the same locations from a baseline model.

Figure 4: Comparison of character representations from a Reversal task model trained with character-level interventions and a baseline model. We use layer 1 for both. Each dot represents a character extracted from different subword tokens, where the color represents the value of the character and the numerals give the string position.

model becomes the one with character inputs and subword outputs (Char-S). With more complicated interactions between form, meaning, and context (Tasks 5–6), subword-based models have a clear advantage on splits where form alone is insufficient to determine the output. For the "Dependent" split in Table 2c, subword models on the target side (Subword+IIT, Char-S) are the best. For the "O+P" split in Table 2d, subword models on both sides (Subword, Subword+IIT) are the best. These observations align with the expectations one might have based on prior literature.

Our IIT models are able to combine the advantage of subword models with character models, leading to the best accuracy on tasks involving form, meaning, and context. On the "Dependent" and "O+P" splits, Subword+IIT models outperform the second best models Char-S and Subword by 1.29%/14.30% and 9.96%/0.69%. Moreover, for form-based generalization, IIT also substantially boosts accuracy on all five OOV splits by an average of 28.21% compared to the Subword model, improving robustness on unseen token sequences.

Even with 175B parameters, GPT-3 is affected by tokenization. We observe similar trade-offs between subword vs. character input/output on Reversal, Unscramble, and Spelling Correction, with the exceptions possibly due to character inputs reducing the value of GPT-3's pretraining.

6 Analysis and Discussion

6.1 Error Analysis on Word Search

To further understand models' biases towards using form, meaning, and context, we analyze performance on the Word Search task. Specifically, we measure how well the predictions match characters in the `letters` or the meaning of the `definition`. We define two new metrics: CharMatch, the percentage of predictions that are a substring of the reversed `letters`, and DefMatch, the percentage of predictions that matches the `definition`. Both metrics would be 100% for a model with 100% sequence-level accuracy. However, they diverge when models make wrong predictions that only capture some aspects of form, meaning, or context. A model biased towards using form would have high CharMatch but low DefMatch, and vice versa for a model biased towards meaning and context.

Table 3 shows the results of this analysis. Subword-based models are biased towards using meaning and context for generalization and so have higher DefMatch scores, whereas character-level models are biased towards using form and so have higher CharMatch scores. These findings are consistent with what we observed in previous experiments. For the "P" split, character-level models (ByT5, Char-ST) perform well, as they exploit shortcuts in the `letters` to identify word boundaries, which are removed in the "O" and "O+P" splits. For this task, only Subword models appear to be viable, and Subword+IIT is the best variant.

Method	CharMatch	DefMatch
Subword	67.80	67.75
+IIT	72.96	**67.97**
Char-T	96.68	50.74
Char-S	66.64	51.99
Char-ST	**99.75**	42.87
ByT5	99.68	57.67

Table 3: CharMatch and DefMatch on the O+P split of Word Search task.

6.2 Interpretable Character-Level Structure

Finally, we note a qualitative advantage of the Subword+IIT models: they embed accurate, coherent representations of characters, illustrated in Figure 4a, with some meaningful clustering of characters (e.g., vowels cluster towards the left). The character representations are 16d vectors extracted at the intervention sites (as shown in Figure 2b) over 2K examples. We use Principal Component Analysis (PCA) to reduce the vectors to 2d and plot the results. As a comparison, we also plot representations extracted at the same locations from the "Subword" baseline model in Figure 4b. As expected, these show no evidence of internal character-level representations. (Appendix D provides similar visualizations for our other tasks.)

7 Conclusion

Character-level tasks have emerged as an Achilles heel for large language models that use subword tokenization. These models do not reliably represent the mapping from subword tokens to the characters they contain, and thus they stumble with character-level manipulations. We showed that Type-level IIT can help. Using Type-level IIT, we trained networks to internally represent characters, and we introduced a new suite of tasks that assess models on character-level tasks involving different combi-

nations of form, meaning, and context. While our Subword+IIT models lag behind character-level tokenization models on simple character-level tasks, they are superior for tasks that blend form, meaning, and context. Overall, these findings suggest that intervention-based methods like IIT provide a powerful set of techniques for training models to modularly represent different kinds of information at different levels of abstraction.

Acknowledgments

This research was supported in part by an Amazon Faculty Research Award to CP and National Science Foundation Grant No. 2104995 to KM. Our thanks to Karel D'Oosterlinck and Atticus Geiger for insightful discussion.

Limitations

The datasets and models produced by this work are intended for research purposes only, not for real world applications. In light of this, we do not see any serious risks with the artifacts produced, though we acknowledge that there can be subtle but significant biases caused by how our task examples interact with how our base models were pretrained. This concern is perhaps especially noteworthy for GPT-3, as we have only partial knowledge of its structure and training inputs.

There are potential risks stemming from IIT as well. With IIT, one shapes aspects of the training process using a high-level causal model. To the extent that this model is intentionally or unintentionally biased in problematic ways, those biases are likely to be amplified in the target model. However, for the current work, the risks here seem minimal, as we are focused on character-level tasks that are mostly games.

References

Gustavo Aguilar, Bryan McCann, Tong Niu, Nazneen Rajani, Nitish Shirish Keskar, and Thamar Solorio. 2021. Char2Subword: Extending the subword embedding space using robust character compositionality. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1640–1651, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Sander Beckers, Frederick Eberhardt, and Joseph Y. Halpern. 2020. Approximate causal abstractions. In *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pages 606–615. PMLR.

Sander Beckers and Joseph Y. Halpern. 2019. Abstracting causal models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):2678–2685.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *International Conference on Learning Representations*.

Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. 2022. GPT-NeoX-20B: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745*.

Kaj Bostrom and Greg Durrett. 2020. Byte pair encoding is suboptimal for language model pretraining. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4617–4624, Online. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. 2022. Canine: Pre-training an Efficient Tokenization-Free Encoder for Language Representation. *Transactions of the Association for Computational Linguistics*, 10:73–91.

Cicero Dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *International Conference on Machine Learning*, pages 1818–1826. PMLR.

Avia Efrat, Uri Shaham, Dan Kilman, and Omer Levy. 2021. Cryptonite: A cryptic crossword benchmark for extreme ambiguity in language. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4186–4192, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Hicham El Boukkouri, Olivier Ferret, Thomas Lavergne, Hiroshi Noji, Pierre Zweigenbaum, and Jun'ichi Tsujii. 2020. CharacterBERT: Reconciling ELMo and BERT for word-level open-vocabulary representations from characters. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6903–6915, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. 2021. Causal abstractions of neural networks. In *Advances in Neural Information Processing Systems*, volume 34, pages 9574–9586.

Atticus Geiger, Zhengxuan Wu, Karel D'Oosterlinck, Elisa Kreiss, Noah D. Goodman, Thomas Icard, and Christopher Potts. 2022a. Faithful, interpretable model explanations via causal abstraction. Stanford AI Lab Blog.

Atticus Geiger, Zhengxuan Wu, Hanson Lu, Josh Rozner, Elisa Kreiss, Thomas Icard, Noah Goodman, and Christopher Potts. 2022b. Inducing causal structure for interpretable neural networks. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 7324–7338. PMLR.

Mor Geva, Ankit Gupta, and Jonathan Berant. 2020. Injecting numerical reasoning skills into language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 946–958, Online. Association for Computational Linguistics.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: decoding-enhanced bert with disentangled attention. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Itay Itzhak and Omer Levy. 2022. Models in a spelling bee: Language models implicitly learn the character composition of tokens. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5061–5068, Seattle, United States. Association for Computational Linguistics.

Ayush Kaushal and Kyle Mahowald. 2022. What do tokens know about their characters and how do they know it? In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2487–2507, Seattle, United States. Association for Computational Linguistics.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Jindřich Libovickỳ, Helmut Schmid, and Alexander Fraser. 2021. Why don't people use character-level machine translation? *arXiv preprint arXiv:2110.08191*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1054–1063, Berlin, Germany. Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Sabrina J Mielke, Zaid Alyafeai, Elizabeth Salesky, Colin Raffel, Manan Dey, Matthias Gallé, Arun Raja, Chenglei Si, Wilson Y Lee, Benoît Sagot, et al. 2021. Between words and characters: A brief history of open-vocabulary modeling and tokenization in NLP. *arXiv preprint arXiv:2112.10508*.

George A Miller. 1995. Wordnet: A lexical database for English. *Communications of the ACM*, 38(11):39–41.

Anirudh Mittal, Yufei Tian, and Nanyun Peng. 2022. Ambipun: Generating humorous puns with ambiguous context. *arXiv preprint arXiv:2205.01825*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Yuval Pinter. 2021. Integrating approaches to word representation. *arXiv preprint arXiv:2109.04876*.

Yuval Pinter, Robert Guthrie, and Jacob Eisenstein. 2017. Mimicking word embeddings using subword RNNs. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 102–112, Copenhagen, Denmark. Association for Computational Linguistics.

Yuval Pinter, Cassandra L. Jacobs, and Jacob Eisenstein. 2021. Will it unblend? In *Proceedings of the Society for Computation in Linguistics 2021*, pages 474–476, Online. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

José Carlos Rosales Núñez, Guillaume Wisniewski, and Djamé Seddah. 2021. Noisy UGC translation at the character level: Revisiting open-vocabulary capabilities and robustness of char-based models. In *Proceedings of the Seventh Workshop on Noisy User-generated Text (W-NUT 2021)*, pages 199–211, Online. Association for Computational Linguistics.

Joshua Rozner, Christopher Potts, and Kyle Mahowald. 2021. Decrypting cryptic crosswords: Semantically complex wordplay puzzles as a target for NLP. In *Advances in Neural Information Processing Systems*.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. BLOOM: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Timo Schick and Hinrich Schütze. 2019. Attentive mimicking: Better word embeddings by attending to informative contexts. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 489–494, Minneapolis, Minnesota. Association for Computational Linguistics.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5149–5152. IEEE.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Yi Tay, Vinh Q. Tran, Sebastian Ruder, Jai Gupta, Hyung Won Chung, Dara Bahri, Zhen Qin, Simon Baumgartner, Cong Yu, and Donald Metzler. 2022. Charformer: Fast character transformers via gradient-based subword tokenization. In *International Conference on Learning Representations*.

Eric Wallace, Nicholas Tomlin, Albert Xu, Kevin Yang, Eshaan Pathak, Matthew Ginsberg, and Dan Klein. 2022. Automated crossword solving. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3073–3085, Dublin, Ireland. Association for Computational Linguistics.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Zhengxuan Wu, Karel D'Oosterlinck, Atticus Geiger, Amir Zur, and Christopher Potts. 2022. Causal Proxy Models for concept-based model explanations. ArXiv:2209.14279.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. ByT5: Towards a Token-Free Future with Pre-trained Byte-to-Byte Models. *Transactions of the Association for Computational Linguistics*, 10:291–306.

Zhiwei Yu, Hongyu Zang, and Xiaojun Wan. 2020. Homophonic pun generation with lexically constrained rewriting. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2870–2876, Online. Association for Computational Linguistics.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. OPT: Open Pre-trained Transformer language models. *arXiv preprint arXiv:2205.01068*.

Supplementary Materials

A Training Details

A.1 IIT Data Generation

Given a training dataset D, generating character-level IIT data can be viewed as sampling triplets of a base example $(x_b, y_b) \in D$, a source example $(x_s, y_s) \in D$, and an intervention example (x_{inv}, y_{inv}) where the i-th character of x_{inv} either comes from the i-th character of x_b (no intervention on i-th character) or a character in x_s (an intervention on i-th character) and y_{inv} is the intervention label. Note that (x_{inv}, y_{inv}) does not need to be in D.

Now we describe the generation algorithm: (1) randomly sample a base example (x_b, y_b); (2) construct x_{inv} by randomly selecting a subset of characters C from x_b as the intervention variables and randomly assign each character in C an intervention value. In our experiments, for each base example, we use a subset of at most 8 characters in tasks 1–4, up to all 64 input characters in task 5, and up to all 24 characters in the `letters` in task 6; (3) For tasks with simple causal models, such as Reversal and Unit Conversion, compute the intervention label y_{inv} based on x_{inv}. If the causal model is not defined over x_{inv}, go back to step (2) to re-sample x_{inv}. Alternatively, for tasks with more complicated causal models, check if there exists an example in D with input equals to x_{inv}. If so, use its label as y_{inv}. If not, go back to step (2) to re-sample x_{inv}; (4) Search for a source example $(x_s, y_s) \in D$ where x_s contains all the intervention values needed to construct x_{inv}. If no such x_s exists, go back to step (2) to re-sample x_{inv}. Otherwise, yield the triplet and go back to (1) until the program generates a total of N triplets. In our experiment, we use an N to be 5 to 10 times larger than the size of D.

A.2 IIT Training Objectives

Given a generative language model f, we can decompose f into pre-intervention layers f_{pre} and post-intervention layers f_{post}, i.e. $f = f_{pre} \circ f_{post}$. For a model trained with the standard maximum likelihood objective $L(f(x), y)$ over input x, output $f(x)$, and label y, we can simply add the IIT objective $L(y'_{inv}, y_{inv})$, where $y'_{inv} = f_{post}(g(f_{pre}(x_b), f_{pre}(x_s)))$ is the intervention output computed from base input x_b and source input x_s with intervention g (which sets a subset values of $f_{pre}(x_b)$ to a subset of values of $f_{pre}(x_s)$), and y_{inv} is the intervention label. The final loss function is a linear combination of the two terms $L = \lambda_1 L(f(x), y) + \lambda_2 L(y'_{inv}, y_{inv})$, where λ_1 and λ_2 are coefficients balancing the two terms.

A.3 Training Hyperparameters

For each task, models are trained until convergence, which leads to approximately 100% accuracy on the training set and over 95% accuracy on validation sets. For T5-based models, the training takes up to 40/20/20/40/30/60 epochs for tasks 1–6. ByT5 models, due to its large size, tend to converge early and overfit on task 1–2, hence we reduce the training epochs on the first two tasks to 10 and 5. All models are trained with a batch size of 16, using Adam optimizer with an initial learning rate of 0.0005 and a linear learning rate decay that halves the learning rate at the end.

A.4 Model Size and Computational Cost

The pre-trained T5-small model has 6 encoder layers and 6 decoder layers, with 60 million parameters in total. The pre-trained ByT5 model has 12 encoder layers and 4 decoder layers, with 300 million parameters in total. Our character-level intervention method does not add any additional weights to the pre-trained model.

We train all models on a single NVIDIA TITAN RTX card with 24GB memory. For the Subword baseline, the training time varies per task from 0.25 to 6 hrs, unit conversion being the fastest and contextual spelling correction being the longest. Compared with Subword models, IIT models take 2.5 times (as IIT training is added on top of base training). Char-T, Char-S, Char-ST models take 1.5 times (due to longer input sequences up to a factor of 3 and output sequences up to a factor of 2). ByT5 models take 4 times. For inference cost, the ratio roughly holds except for IIT, which has the exact same inference cost as Subword baseline.

B In-context Learning Details for GPT-3

To assess whether large language models like GPT-3 have the ability to learn character-level manipulations, we evaluate one of the largest publicly available foundation models GPT-3 (`davinci-003` 175B) on four of our tasks: reversal, unit conversion, unscramble and single-word spelling correction. For all of our tasks, we adapt the in-context learning paradigm without further fine-tuning. We provide k-shots in-context learning demonstrations with input-output pairs before query model results for an unseen testing input. We set the temperature to 0.0 with a short task description in the beginning. We allow maximally 64 generated tokens. In addition, we evaluate performance by providing character-level parsing by separating a word character-by-character using the hyphen (i.e., "-") for alphabet letters or space for number (since "-." is a single token in GPT-3 vocab). Hyphens are added for both the input and output strings. Spaces are inserted before digits and decimal point only, where the space and the digit is tokenized into a single token. We choose k to be 50 without character-level parsing, and 25 with character-level parsing to avoid exceeding the prompt length restriction. For evaluation, we follow the metrics for evaluating T5-based models. We use GPT-3 models from `OpenAI` for all of our experiments.[8] Examples for each task are included in Figure 5 to 8.

```
Please follow the instructions to manipulate the characters of the INPUT string
and generate the desired OUTPUT string. Please reverse the input string.

INPUT: rewols
OUTPUT: slower

[additional demonstrations abbreviated to save space]

INPUT: etaivbo
OUTPUT: obviate
```

Figure 5: Example GPT-3 prompt (gray) and targeted GPT-3 completion (bold) for the word reversal task.

```
Please follow the instructions to convert the unit of the number mentioned in
the INPUT string and generate the desired OUTPUT string.

INPUT: unit conversion: 91.2 cm to m
OUTPUT: 0.912

[additional demonstrations abbreviated to save space]

INPUT: 755.7 km in m
OUTPUT: 755700
```

Figure 6: Example GPT-3 prompt (gray) and targeted GPT-3 completion (bold) for the unit conversion task.

C License and Distribution

Below are the license and distribution of artifacts used in this research.

Wikipedia corpus: The Wikipedia dump is licensed under the Creative Commons Attribution-ShareAlike 3.0 Unported License (CC BY-SA) and the GNU Free Documentation License (GFDL). We access it through a pre-processed subset "20220301.en" provided by HuggingFace[9].

[8] https://openai.com/api/

[9] https://huggingface.co/datasets/wikipedia#licensing-information

```
Please follow the instructions to manipulate the characters of the INPUT string
and generate the desired OUTPUT string. Please unscramble the input string.

INPUT: m-e-o-s-h
OUTPUT: h-o-m-e-s

[additional demonstrations abbreviated to save space]

INPUT: l-e-a-s-t
OUTPUT: t-a-l-e-s
```

Figure 7: Example GPT-3-C prompt (gray) and targeted GPT-3-C completion (bold) for the word unscramble task.

```
Please follow the instructions to manipulate the characters of the INPUT string
and generate the desired OUTPUT string. Please reverse the input string.

INPUT: r-e-w-o-l-s
OUTPUT: s-l-o-w-e-r

[additional demonstrations abbreviated to save space]

INPUT: n-e-g-e-d
OUTPUT: d-e-g-e-n
```

Figure 8: Example GPT-3-C prompt (gray) and targeted GPT-3-C completion (bold) for the word reversal task.

```
Please follow the instructions to manipulate the characters of the INPUT string
and generate the desired OUTPUT string. Please correct any spelling error of
the input string.

INPUT: transported in an impure alfalfz seed shipment coming
OUTPUT: transported in an impure alfalfa seed shipment coming

[additional demonstrations abbreviated to save space]

INPUT: letter nold from the corresponding slot in a font
OUTPUT: letter mold from the corresponding slot in a font
```

Figure 9: Example GPT-3 prompt (gray) and targeted GPT-3 completion (bold) for the spelling correction with context task.

The Online Text Plain English Dictionary: The Online Text Plain English Dictionary (OPTED) is distributed under the license here[10]. We access the JSON version publicly available on GitHub[11].

WordNet and NLTK: The WordNet software and database is distributed under WordNet 3.0 license[12]. We access it through the NLTK 3.7 package, which is distributed under the Apache 2.0 License[13].

Huggingface packages: We use the transformers 4.22.2 and the datasets 2.5.2 packages, both are distributed under Apache License 2.0.[14]

[10] https://www.mso.anu.edu.au/~ralph/OPTED/
[11] https://github.com/eddydn/DictionaryDatabase
[12] https://wordnet.princeton.edu/license-and-commercial-use
[13] https://github.com/nltk/nltk/wiki/FAQ
[14] https://github.com/huggingface/transformers/blob/main/LICENSE

```
Please follow the instructions to manipulate the characters of the INPUT string
and generate the desired OUTPUT string. Please find an reversed valid English
word from the provided letters. The meaning of the word is expressed in the
input string.

INPUT: a motor vehicle with four wheels: tseuqninoteahpnarrowness
OUTPUT: phaeton

[additional demonstrations abbreviated to save space]

INPUT: a small vehicle moved on wheels: elbitrevnocbamboonootrac
OUTPUT: convertible
```

Figure 10: Example GPT-3 prompt (gray) and targeted GPT-3 completion (bold) for the word search task.

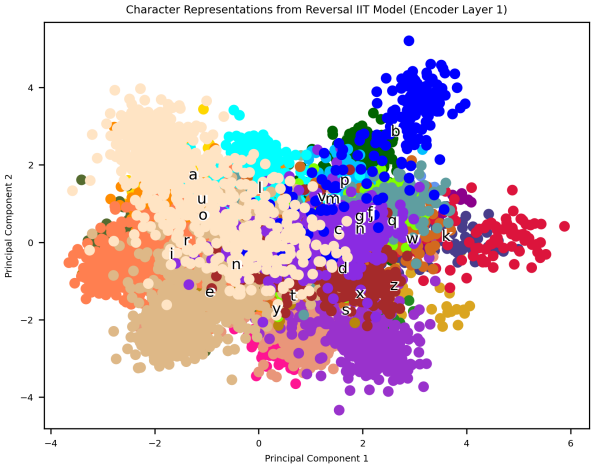
PyTorch packages: We use PyTorch 1.12.1 distributed under BSD License (BSD-3)[15].

Pre-trained T5-small and ByT5-small models: Both models are distributed under the Apache 2.0 License[16][17]. We download the models from Huggingface.

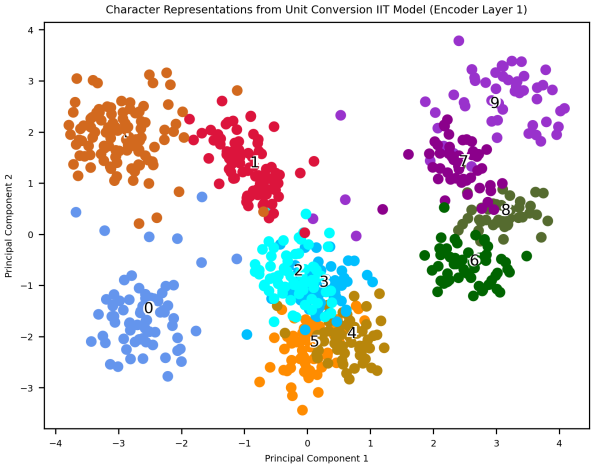
D Visualization of IIT character representations

We visualize the character representations extracted from models trained with character-level interventions. The character representations encode human-interpretable structures including (1) clear character-based clusters in low-dimension projection of hidden representations (2) larger inter-cluster distance between vowels and consonants (with letter "y" mostly in between).

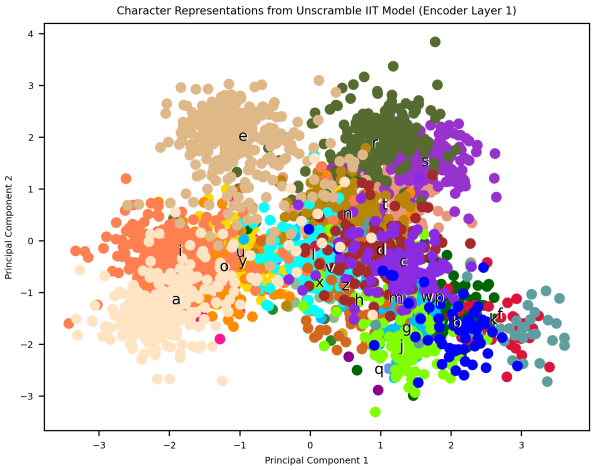
[15] https://pypi.org/project/torch/
[16] https://huggingface.co/t5-small
[17] https://huggingface.co/google/byt5-small



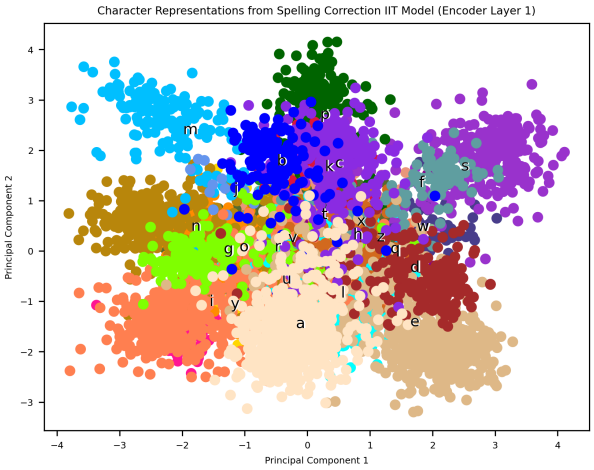
(a) Reversal with letters a to z.



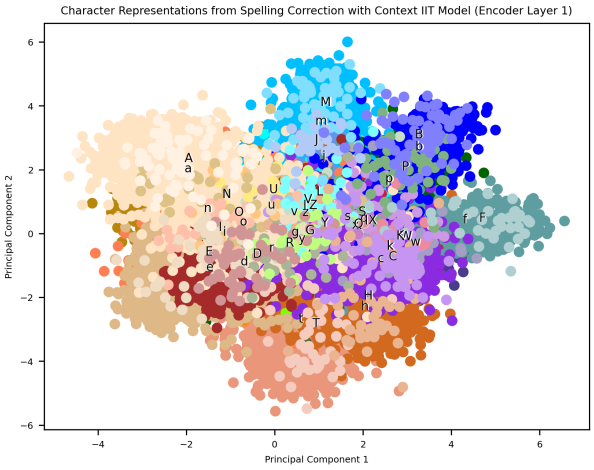
(b) Unit Conversion with digits 0 to 9 and the decimal point.



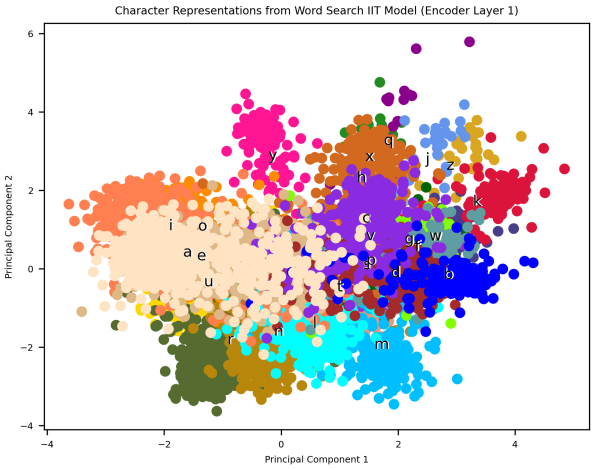
(c) Unscramble with letters a to z.



(d) Spelling correction with letters a to z.



(e) Contextual Spelling Correction with letters A to Z and a to z. Tint colors represent the upper cases, while darker colors represent the lower cases. We omit clusters of digits, punctuation marks, and white space due to their large inter-cluster distances to letters.



(f) Word Search with letters a to z.

Figure 11: Character representations from subword models trained with character-level interventions. Each dot represents a character extracted from different subword tokens, where the color represents the value of the character. The character label for each cluster is anchored at the cluster center. Figure best viewed in color.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Yes, in the required limitation section.

☑ A2. Did you discuss any potential risks of your work?
Yes, in the required limitation section.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract and Section 1.

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 3, 4, 5, Appendix A, and Appendix B.

☑ B1. Did you cite the creators of artifacts you used?
Yes, we cite the Wikipedia dataset in section 3.5 and the The Online Text Plain English Dictionary in section 3.6. The rest of the datasets are generated by us. We cite the T5, ByT5, and GPT-3 pre-trained models in section 5.

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Yes, in Appendix C.

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
We used publicly available resources according to the terms they were released. For intended use of our models, see the limitation section.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. Our datasets are synthetic.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Section 3 and 5.

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Section 3.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

C ☑ Did you run computational experiments?

Section 5, Appendix A, and Appendix B.

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Appendix A.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 5 and Appendix A.

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 5 and Appendix B.

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Appendix C.

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Not applicable. Left blank.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Not applicable. Left blank.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Not applicable. Left blank.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Not applicable. Left blank.