# A CHAT ABOUT BORING PROBLEMS: STUDYING GPT-BASED TEXT NORMALIZATION

*Yang Zhang*, Travis M. Bartley†*, Mariana Graterol-Fuenmayor, Vitaly Lavrukhin, Evelina Bakhturina, Boris Ginsburg*
Nvidia Corporation,
City University of New York, Graduate Center†

## ABSTRACT

Text normalization - the conversion of text from written to spoken form - is traditionally assumed to be an ill-formed task for language models. In this work, we argue otherwise. We empirically show the capacity of Large-Language Models (LLM) for text normalization in few-shot scenarios. Combining self-consistency reasoning with linguistic-informed prompt engineering, we find LLM based text normalization to achieve error rates around 40% lower than top normalization systems. Further, upon error analysis, we note key limitations in the conventional design of text normalization tasks. We create a new taxonomy of text normalization errors and apply it to results from GPT-3.5-Turbo and GPT-4.0. Through this new framework, we can identify strengths and weaknesses of GPT-based TN, opening opportunities for future work.

***Index Terms*—** Text-normalization, GPT, large-language-models, in-context learning, finite state automata, text-to-speech

## 1. INTRODUCTION

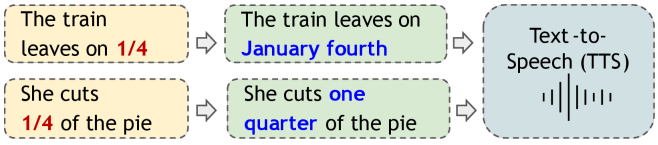


**Fig. 1**. Example of text normalization across semiotic classes. The same semiotic token ("1/4") has distinct normalizations depending on context.

Text normalization (TN) is an essential prepossessing step for text-to-speech generation. While the majority of character-pronunciation mappings may be learned from upstream grapheme-to-phoneme (G2P) or forced alignment, many orthographic symbols require contextual knowledge for successful verbalization. For example, consider the string "1/4" (as seen in Figure 1). Depending on context, pronunciation can vary between "January the fourth" and "a quarter". Such types of meaning-dependent tokens are often referred to as *semiotic tokens* and their TN context as *semiotic classes* [1]. Common examples of semiotic classes would be cardinal digits, dates, fractions, and time.

Two notorious limitations of TN as a language processing task are the scarcity of paired training data available and need for high (if not perfect) accuracy. The former is due to the infeasibility of covering all potential semiotic tokens in context (e.g. a dataset for all cardinal strings). The latter stems from the low fault-tolerant scenarios where these tokens occur (Engineering, Finance, Medicine) [2]. In such cases, the creation of 'unrecoverable errors', such as hallucination of an additional number or misreading of a currency symbol, can have disastrous consequences. As such, most production level systems, such as Google's Kestrel, rely on hand-coded weighted finite-state grammars (WFST) [3, 4]. These limitations on a seemingly trivial task has led Richard Sproat – one of the modern pioneers of TN - to claim, "until one can solve 'boring' problems like this [TN] using purely AI methods, one cannot claim that AI is a success [..] Various groups have tried, but so far nobody has eliminated the problem of unrecoverable errors." [5, 6, 3]

In this paper, we issue a challenge to Sproat [5]. We argue that *true* "unrecoverable" errors are, in fact, minimal for state-of-the-art language modeling methods. Rather, the issue lies in the documentation of TN errors. We believe TN requires a more nuanced error taxonomy beyond the binary of 'correct' and 'unrecoverable'. We introduce a granular label schema for GPT based TN (GPT-TN) error categorization, along with a simple labeling tool for analysis via the NeMo Text-Processing toolkit [7]. Using GPT-3.5-Turbo and GPT-4.0, we evaluate TN performance over few-shot prompting and consistency reasoning scenarios. Our analysis is as follows:

- GPT-TN outperforms Kestrel, a state-of-the-art WFST-based TN system, by 40% percent
- Manual error analysis shows *true* "unrecoverable" GPT-TN errors to be near non-existent
- GPT-TN productions outperform Kestrel's normalization in conventional speech scenarios

*Equal contribution

## 2. METHOD

### 2.1. Error Taxonomy

We first motivate our rationale for a TN error taxonomy. One notes our definition of TN does not require an injective relation. In Figure 1, the date "1/4" may be verbalized as "January the fourth" or "the fourth of January" without changing meaning. However, normalization as "the day after January the third" would be perceived as incorrect - despite being grammatically and semantically equivalent.

This phenomenon relates to the linguistic notion of *felicity*. Broadly, felicitous statements are the subset of grammatical sentences that are contextually permissible (with *infelicitous* statements being the disjoint set) [8]. While in specific contexts only one sentence may be felicitous, other contexts permit many equivalent substitutions. In our analysis of TN errors, we find several cases where errors relate to felicity. We note two cases: (1) felicity regarding sentential context and (2) felicity in regards to semiotic class.

Consider TN for a US telephone number;

**Text**: 312-236-2012

**a**: three twelve two thirty six twenty twelve

**b**: three one two two three six two zero one two

Both strings are felicitous (for American English) in regards to the intended sentential meaning. Yet, were **a** to be taken as a gold example, **b** would be treated as an error. As such, the failure to account for felicity at the sentence level produces a false negative. An effective analysis of TN would need to account for this variation. In contrast, consider:

**Text**: b. 03-02-2001

**c**: born march second two thousand one

**d**: b zero three zero two two thousand one

Absent context, both **c** and **d** are valid normalizations, However, only **c** is acceptable for a date of birth and thus **d** is infelicitous. Yet, **d** still retains the original textual information and is thus preferable to other false productions (i.e. it is not 'unrecoverable'). A proper TN analysis would seek to note this distinction between errors.

It is the above reasoning that motiavates us to introduce error categories for GPT-TN analysis. They allow us to distinguish felicity related errors from those stemming from natural LLM behavior (i.e. noisy text-correction, translation of foreign words, hallucinations). In total, our analysis identifies six label categories (along with the binary 'Correct' and 'Incorrect' labels) for TN. They are:

- **format**: Errors derived from infelicitous normalization regarding semiotic class.
- **paraphrase**: Replacement or reordering of words, or substitution of acronyms with full wording. (e.g. "happily he danced" $\rightarrow$ "he danced happily.")
- **fix**: Grammar or spelling correction of a noisy sample. (e.g. "he sleep yesterday" $\rightarrow$ "he slept yesterday".)
- **artifact**: Introduction of non-original text due to idiosyncratic behavior of LLM prompting (e.g. Prepending ("Sure, I can do that").
- **translation**: Normalization of a string in a non-English sentence sample. (e.g. "Louis XIV était roi de france" $\rightarrow$ "Louis quatorze était roi de france".)
- **other**: Misc. category covering complete corruption of original sentence. Includes, but is not exclusive or unique to, 'unrecoverable errors.'

### 2.2. Dataset

For experiments, we use the Google TN dataset [9]. Comprised of around 1.1 billion words from Wikipedia, it is one of the few publicly available English TN datasets and provides sizeable coverage of common semiotic classes. Individual tokens are paired with semiotic class and a target normalization generated with Google's Kestrel TN system [3].

For development, we use 1000 sentences from the provided development set. Final test evaluations are performed over the entire test set, comprised of 7551 sentences. As Kestrel is designed for TTS, the dataset includes several artifacts such as "sil" and special suffixes for single character. We remove these as a prepossessing step.

### 2.3. Experiments

We compare GPT-3.5-Turbo and GPT-4.0 LLMs (March 1st and current release, respectively) against Google's Kestrel system. For all but the final accuracy results, we solely evaluate GPT-3.5-Turbo on the first 1000 samples from the development set.

We seed models for normalization through the completion API, providing user examples in the schema "Normalize: {EXAMPLE}" and target normalization as sample conversation turns. We experiment with several prompting approaches. From the training dataset, we sample 1) random pairs of written-normalization sentences 2) written-normalization semiotic token pairs, one for each class (see [1] for list of all classes) with surrounding sentence context and 3) semiotic token pairs, one for each class, without surrounding context. For 2) and 3), we experiment with varying the number of samples per semiotic class (r=1,2,3). For 1), we simply sample the same total quantity of examples for all variations of 2).

After prompting experiments, we choose the best two configurations and augment performance using self-consistency

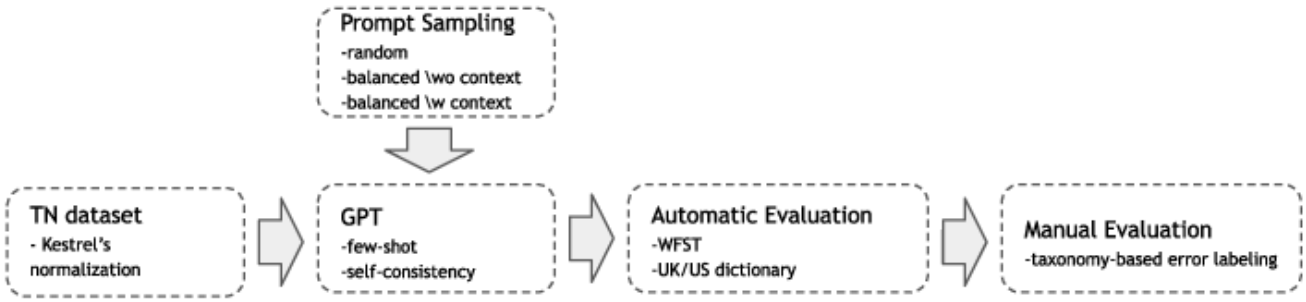


**Fig. 2**. GPT normalization and evaluation pipeline.

reasoning as in [10]. We poll for 20 completion outputs over a temperature of 0.5 and use majority voting to determine final output. After evaluating over the development set, we use the best consistency+prompting configuration pair for test evaluation using both GPT-3.5-Turbo and GPT-4.0. (See GPT-TN normalization pipeline in Figure 2.)

## 3. RESULTS

### 3.1. Evaluation

Prior to final evaluation, we filter out irrelevant variations between target and prediction samples. We remove common GPT insertions ("No need for normalization", "This sentence seems to be incomplete") and filter obvious non-English samples using regex search. Simple felicitous variations (such as "August fourth" vs. "fourth of August") are corrected using a WFST-grammar. We also use a short pre-defined dictionary to correct for differences in UK/US spelling. Finally, we screen for variations in punctuation and article insertion.

For final accuracy results, we perform manual evaluation with our error taxonomy. This is done through an HTML-based tool (see Figure 3) to efficiently label the remaining test cases with error types as described in Section 2.1.

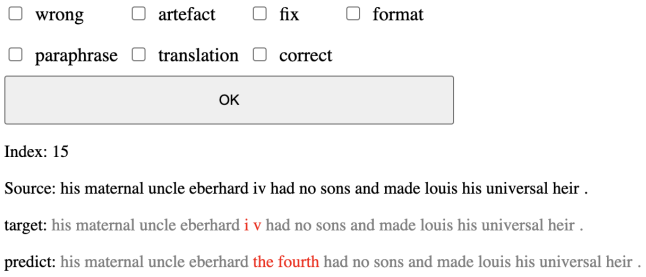


**Fig. 3**. HTML error labeling tool. "Source" is written form, "target" is Kestrel's normalization and "predict" is GPT's normalization.

### 3.2. Final Performance

Table 2 shows GPT-3.5-Turbo's exact accuracy on the development set without manual labeling. Sampling two sentences per semiotic class in context yields the highest accuracy of 91.1% after automatic evaluation. After self-consistency the accuracy improves to 92.1%.

We use this inference recipe to evaluate GPT-3.5-Turbo and GPT-4.0 over the test dataset. (see Table 3.) From GPT-3.5.-Turbo's 692 errors after automatic evaluation, manual labeling found only 264 to be actual errors. (see Section 2.1). GPT-4.0 significantly improves results, reducing the initial 540 errors after automatic evaluation to only 108 errors.

The 540 test cases where GPT-4.0 differs from Kestrel after automatic evaluation we also manually analyzed for Kestrel's normalization quality. With 174 errors, we found GPT-4.0 makes 40% less errors than Kestrel, mainly attributed to the infelicitous normalization category ("format").

## 4. DISCUSSION

We first note that self-consistency polling and semiotic class coverage is critical for improved downstream performance. Both GPT-3.5-Turbo and GPT-4.0 greatly benefit from these methods. Interestingly, there appears to be a saturation point of sample quantity, with both models deteriorating in performance with r=3 classes across all sampling methods. We surmise this is likely due to the 'lost in the middle' phenomenon [11]: essential context for normalization can be lost over large token spans. This is likely why 'w/o context' sampling can remain competitive, with the lack of real context dependencies permitting scaling.

As seen in the disparity between Table 2 and Table 3, the introduction of an error taxonomy is critical for TN analysis. We see that among all models, infelicitous normalizations ('format') made up the vast majority of errors. Actual "unrecoverable" errors made up less than 1% of all total normalizations from the GPT-4.0 model. While this is still 8x the number in Kestrel, we note that this factor may be misleading. Our analysis showed that several supposed errors were due to contamination in either source data or target predictions (see

| |
|---|
| **Input:** 2007 i triple e conference<br>**GPT [acceptable]:** two o o seven i e e e conference<br>**Target:** two o o seven i triple e conference. |
| **Input:** http : //web.archive.org<br>**GPT [wrong]:** h t t p : / / w e b . a r c h i v e . o r g<br>**Target:** h t t p : slash slash web dot archive dot org |
| **Input/target:** it was originally mooted<br>**GPT[wrong]:** it was originally suggested |
| **Input:** mcmath , p .267<br>**GPT [wrong]:** mcmath , p two hundred sixty seven<br>**Target:** mcmath , p point two six seven |
| **Input:** student stories of 9/11<br>**GPT [correct]:** student stories of nine eleven<br>**Target [wrong]:** student stories of nine elevenths |
| **Input:** ch .2 document 75<br>**GPT [correct]:** chapter two document seventy five<br>**Target [wrong]:** chains point two document seventy five |
| **Input [wrong]:** more than rs.10 , 00 , 000 , 00<br>**GPT [correct]:** more than ten crore annual business<br>**Target [wrong]:** more than ten rupees , o o , o o o , o o |

**Table 1**. Examples of GPT-4.0 normalizations errors. Target normalizations are generated from Google's Kestrel Text Normalization system.

Table 1). As such, the slight rise in unrecoverable errors belies GPT-TN's ability to correct for data contamination.

Further, upon comparison of felicitous productions by GPT-4.0 against Kestrel, we found many of the former's productions to be of better quality. While both Kestrel and GPT-4.0 provided recoverable normalizations, the latter's was of greater quality and more colloquial.

Taking in both qualitative and quantitative observations, we conclude that GPT-TN provides both more accurate and higher quality normalizations than WFST-based methods. In particular, the former excels in cases where the range of felicitous normalizations is particularly high. Even in cases of error, the rate of 'unrecoverable' errors is minor. Though in cases of zero-tolerance, Kestrel still has a slight edge.

## 5. CONCLUSION

We empirically demonstrate the effectiveness of GPT-TN for at-scale text-normalization. After careful analysis of TN errors, we find that modern LLMs can outperform rule-based normalization systems by 40%, with minuscule instances of "unrecoverable" normalization errors. We show that GPT-TN outputs are of higher quality and can self-correct for corrupted inputs.

Further, we demonstrate a need for greater nuance in TN analysis. TN outputs can cover a wide range of felicitous targets and are ill-served by binary label schemes. We encourage future work to explore refinements to this taxonomy to develop even more effective TN systems.

| Prompt Samp. Method | r=1 | r=2 | r=3 |
|---|---|---|---|
| w/ context | 88.8/91.1 | **91.1/92.1** | 90.5 |
| w/o context | 89.8 | 90.4/91.5 | 89.8 |
| Rand. | 87.1 | 88.7 | 88.9 |

**Table 2**. Sentence level normalization accuracy over prompting techniques for GPT-3.5-Turbo. Evaluation is performed over first 1000 samples of Google TN development set. First number represents % accuracy based on the corresponding prompt sampling method and number of examples (**r**) per semiotic class ("Rand." samples equivalent total samples as "w/ context"). Second number (if provided) is the % accuracy after self-consistency via majority voting (N=20 responses; t=0.5 temperature).

| Error Type | GPT-3.5 | GPT-4.0 | Kestrel |
|---|---|---|---|
| Errors (Post Auto) | 692 | 540 | 540* |
| Errors (Post Manual) | 264 | 108 | 174 |
| Foreign Language | 29 | 11 | 26 |
| Format | 173 | 62 | 143 |
| Paraphrase | 26 | 10 | 1 |
| Fix | 2 | 9 | 0 |
| Artifact | 19 | 9 | 3 |
| Other | 15 | 7 | 0 |
| Final Accuracy | .965 | **.986** | .977 |
| Unrecoverable Errors | N/A | 8 | 1 |

**Table 3**. Breakdown of normalization errors by label type for GPT-3.5-Turbo, GPT-4.0 and the WSFT-based Kestrel TN. Errors '(Post Auto)' and 'Errors (Post Manual)' refer to errors found after automatic evaluation and further after manual analysis, respectively. Kestrel errors are derived by manual error analysis of target normalizations of GPT-4.0 errors.

## 6. REFERENCES


[1] Paul Taylor, *Text-to-Speech Synthesis*, Cambridge University Press, 2009.

[2] Hao Zhang, Richard Sproat, Axel H Ng, Felix Stahlberg, Xiaochang Peng, Kyle Gorman, and Brian Roark, "Neural models of text normalization for speech applications," *Computational Linguistics*, 2019.

[3] Peter Ebden and Richard Sproat, "The kestrel tts text normalization system," *Natural Language Engineering*, 2015.

[4] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman, *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*, Addison-Wesley Longman Publishing Co., Inc., USA, 2006.

[5] Richard Sproat, "Boring problems are sometimes the most interesting," *Computational Linguistics*, vol. 48, no. 2, pp. 483–490, June 2022.

[6] Richard Sproat, "Lightly supervised learning of text normalization: Russian number names," in *IEEE Spoken Language Technology Workshop*, 2010.

[7] Yang Zhang, Evelina Bakhturina, and Boris Ginsburg, "NeMo (Inverse) Text Normalization: From Development to Production," in *Interspeech*, 2021.

[8] John L. Austin, *How to do things with words*, William James Lectures delivered at Harvard University in 1955. Harvard University Press, Cambridge, Massachusetts, second edition. edition, 1975 - 1975.

[9] Richard Sproat and Navdeep Jaitly, "An RNN model of text normalization," in *Interspeech*, 2017.

[10] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou, "Self-consistency improves chain of thought reasoning in language models," *arXiv preprint arXiv:2203.11171*, 2022.

[11] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang, "Lost in the middle: How language models use long contexts," *arXiv preprint arXiv:2307.03172*, 2023.