Lexical Substitution as Causal Language Modeling

Ning Shi **Bradley Hauer** **Grzegorz Kondrak**
Alberta Machine Intelligence Institute (Amii)
Department of Computing Science
University of Alberta, Edmonton, Canada
{ning.shi,bmhauer,gkondrak}@ualberta.ca

Abstract

Causal language models such as the GPT series have achieved significant success across various domains. However, their application to the lexical substitution task (LST) remains largely unexplored due to inherent limitations in autoregressive decoding. Our work is motivated by our observation that existing LST approaches tend to suffer from a misalignment between the pre-training objectives of the language models that they employ, and their subsequent fine-tuning and application for substitute generation. We introduce PromptSub, the first system to use causal language modeling (CLM) for LST. Through prompt-aware fine-tuning, PromptSub not only enriches the given context with additional knowledge, but also leverages the unidirectional nature of autoregressive decoding. PromptSub consistently outperforms GeneSis, the best previously published supervised LST method. Further analysis demonstrates the potential of PromptSub to further benefit from increased model capacity, expanded data resources, and retrieval of external knowledge. By framing LST within the paradigm of CLM, our approach indicates the versatility of general CLM-based systems, such as ChatGPT, in catering to specialized tasks, including LST.[1]

1 Introduction

Lexical substitution task (LST) is to identify appropriate replacements for a designated target word in context while maintaining the contextual meaning and coherence of the text (McCarthy, 2002; McCarthy and Navigli, 2007). For example, given the sentence "Let me *begin* again", an LST system would be expected to provide words such as *start* or *commence* as substitutes for *begin*. LST is an important task due to its numerous applications, including word sense disambiguation (Hou et al., 2020), word sense induction (Eyal et al., 2022),

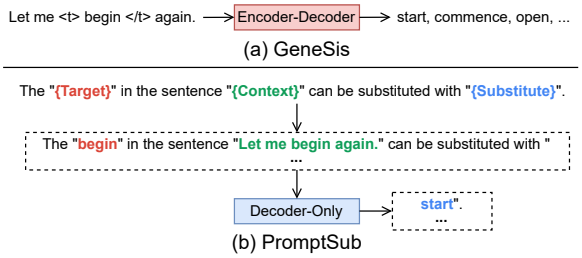

Figure 1: Comparison between (a) GeneSis (Lacerra et al., 2021b) and (b) our proposed PromptSub.

lexical simplification (Aumiller and Gertz, 2022), adversarial attacks and defenses (Li et al., 2021), semantic change detection (Card, 2023), and natural language watermarking (Yang et al., 2022).

Recent prior work on LST leverages pre-trained language models (PLMs), specifically *masked* language models (MLMs) (Lin et al., 2022; Michalopoulos et al., 2022; Omarov and Kondrak, 2023), of which BERT (Devlin et al., 2019) is a well-known example. Since MLMs are trained on the task of predicting likely words in a context where a single word is masked, they seem to be a natural fit for LST. However, masking a word is an information-losing process. As a result, the predicted substitutes may fit the context well, but can significantly alter the original meaning of the sentence.

As an alternative to masked language modeling, we propose to employ *causal* language modeling instead. While MLMs first encode the entire context around the mask and then decode output from this encoding, causal language models (CLMs) are trained to predict the next token in a sequence given *only the previous tokens* as context (Radford et al., 2018). This linear processing of text is referred to as *auto-regressive decoding*; by eschewing the need for discrete encoding and decoding phases, these models can achieve high performance in generative tasks, without an encoder that increases the number of parameters. These *decoder-only* models include

[1]Our code and data are publicly available on GitHub: https://github.com/ShiningLab/PromptSub.

the well-known GPT series (Brown et al., 2020), which powers popular language generation tools such as ChatGPT (OpenAI, 2023). However, prior methods for applying a pre-trained CLM to LST go no further than simple prompting (Lee et al., 2021).

In this paper, we present the first method to efficiently reduce LST to causal language modeling: PromptSub, a system based on lexical substitution via prompt-aware fine-tuning. Our approach bridges the gap between the pre-training of CLMs and their fine-tuning for LST via the same training objective (i.e., to predict the next token). By way of an innovative prompting strategy, PromptSub empowers a decoder-only CLM to leverage the full bidirectional context of a given LST instance, and also seamlessly integrate external knowledge into an auto-regressive language modeling strategy.

In our experiments, PromptSub consistently surpasses the previous best supervised method, GeneSis (Lacerra et al., 2021b), across all datasets, metrics, and settings. Figure 1 illustrates how GeneSis and PromptSub employ encoder-decoder (Sutskever et al., 2014) and decoder-only models respectively. Our extensive evaluations indicate that PromptSub either matches or exceeds previously published methods, establishing a new state of the art on the most recent LST benchmark, SWORDS (Lee et al., 2021). Notably, PromptSub outperforms MLM-based approaches, previously recognized as the state of the art, by a large margin (Yang et al., 2022; Wada et al., 2022). Our detailed analysis highlights the robustness and extensibility of PromptSub, showing that it can take advantage of greater model capacity, leverage a broad array of resources, and benefit from external knowledge through retrieval-augmented generation (RAG; Lewis et al., 2020b).

2 Related Work

Conventional LST techniques predominantly capitalize on external knowledge bases (Hassan et al., 2007; Szarvas et al., 2013a; Hintz and Biemann, 2016) and learned word embeddings to identify and rank potential substitution candidates based on predefined metrics (Melamud et al., 2015b,a; Garí Soler et al., 2019). These methods often depend heavily on external resources like WordNet (Miller, 1995), with additional processes such as the manual ranking and rule construction often required to optimize outcomes. Recognizing these limitations, recent initiatives have emerged to harness the advantages of PLMs.

Prior work indicates that contextualized representations obtained from PLMs can be applied to LST by incorporating context-based scores (Seneviratne et al., 2022) and decontextualized embeddings (Wada et al., 2022). In an effort to augment PLMs with knowledge derived from lexical resources, Lin et al. (2022) proposed involving gloss matching in pre-training. Michalopoulos et al. (2022) advocate for the incorporation of structured knowledge from lexical databases.

On the one hand, certain of these approaches utilize PLMs primarily as feature extractors. Thus, the complete potential of PLMs remains untapped due to the disconnect between their pre-training objectives and subsequent applications. On the other hand, to align with pre-training, others (Zhou et al., 2019) estimate the probability distribution of potential replacements through masked language modeling (Devlin et al., 2019). This inclination towards MLMs, as opposed to CLMs, has led to the over-representation of encoder-only PLMs, leaving the application of decoder-only architectures largely unexplored.

Similarly, while prior work has explored the ideas of enriching LST inputs with target words (Arefyev et al., 2020) and semantic knowledge (Omarov and Kondrak, 2023), how to inject such knowledge into PLMs remains an open question. This issue is particularly true within the prevailing trend of unsupervised methods that exclude the fine-tuning stage.

Supervised approaches to LST, such as those by Szarvas et al. (2013a,b), were initially limited by data scarcity until the advent of GeneSis (Lacerra et al., 2021a,b) and ParaLS (Qiang et al., 2023). GeneSis adopts a sequence-to-sequence model, generating substitutes given the context and marked target word. By concatenating multiple datasets, fine-tuning a PLM specifically for LST was made viable, achieving strong results despite the scarcity of annotated data in the domain. ParaLS produces substitutes through a paraphraser, utilizing a heuristics-based decoding strategy. This facilitates fine-tuning PLMs on paraphrase data, which is available in relatively large quantities.

However, in both GeneSis and ParaLS, a discernible gap persists between the pre-training of PLMs and their subsequent fine-tuning. Furthermore, they are both rooted in an encoder-decoder framework (Lewis et al., 2020a), depending on external resources, and require post-processing steps

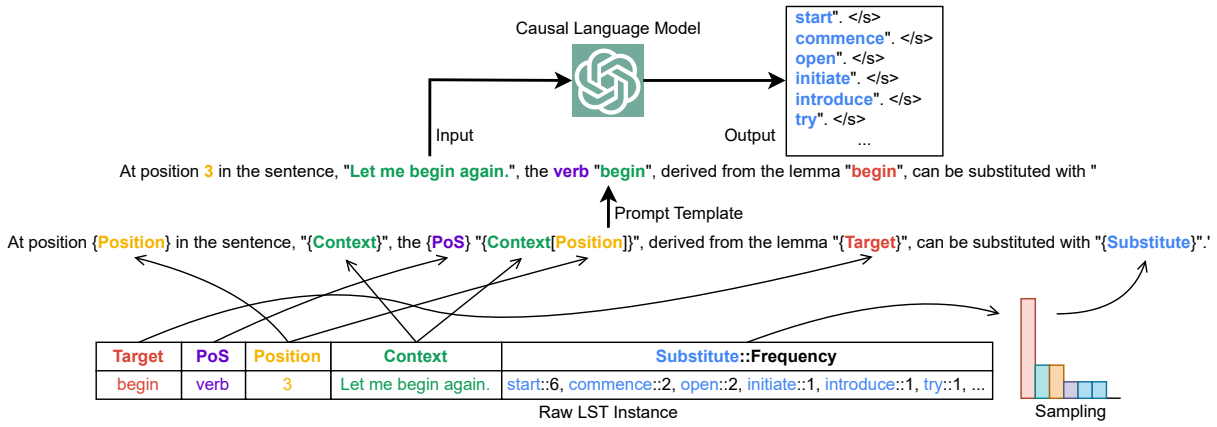

Figure 2: An illustration of PromptSub. An LST instance is transformed into a description by populating a prompt with details. A CLM estimates the probability distribution of potential substitutes at the final placeholder.

that involve adjustable heuristics and thresholds. This raises a pivotal question: does LST have to be approached in a two-step manner, where substitutions are first generated and then reranked using manually designed scores? Or, is it possible to create a single-step, end-to-end, generative solution, that also sidesteps the need for external resources and manually-crafted heuristics? In presenting PromptSub, we argue for the latter: a first-of-its-kind single-step approach to generating substitutions via a decoder-only language model.

3 Methodology

In this section, we formally define LST and CLM, outline our sampling strategy, and detail our prompt engineering techniques.

3.1 Definitions

We introduce our theoretical framework that reduces LST to CLM, building upon two binary problems we defined.

Lexical substitution task (LST) involves identifying suitable replacements for target words while preserving the contextual meaning of the sentence. Formally, given an input sentence $S = w_1^n$ containing a target word w_x, the objective of LST is to return a ranked list of m appropriate replacements for w_x, which are selected from a vocabulary $\mathcal{V}$. For example, consider *begin* as the target word w_x in the sentence S = *"Let me begin again"*. If we are to specify $m = 3$ substitutes, a reasonable output would be ["start", "commence", "open"].

Causal language modeling (CLM) refers to prediction of the next word in a sequence given the preceding words. Formally, given a sequence of words $s = w_1^n$ of length n, the objective of CLM is to model the conditional probability distribution of the next word: $p(w_{n+1} \mid w_1^n)$. CLM is autoregressive: words are predicted one at a time, conditioned on the context of the previous words. By applying a decoder-only model repeatedly, CLM can be used to model the probability of any sequence of words: $p(w_{n+1}^{n+k} \mid w_1^n)$.

We define a binary decision problem of lexical substitution (LexSub) which returns TRUE if two words are lexical substitutes in a given sentence, and FALSE otherwise (Hauer and Kondrak, 2023):

LexSub(S, w_x, w_y) := "the word w_x can be replaced by the word w_y in the sentence S without altering its meaning"

Similarly, we define a binary decision problem of word prediction (WP) as:

WP(S, w) := "the word w has the same meaning as the masked word in the sentence S"

LexSub is thus reducible to WP in a straightforward way:

$$\text{LexSub}(S, w_x, w_y) \Leftrightarrow \text{WP}(S, w_x) \wedge \text{WP}(S, w_y)$$

In practice, implementations of methods for LexSub or WP may return a probability value instead of a Boolean. LST datasets often require a ranked list of substitutes for each instance. To satisfy this, given a method for solving WP as we defined, we can simply rank each word w in the vocabulary by the probability returned by WP(S, w). To apply CLM to LST, we constrain the word to be identified (in WP) or replaced (in LexSub) to appear at the end of the context. We can thus model LexSub and WP as *autoregressive* language modeling tasks suitable for use with decoder-only models.

3.2 PromptSub definition

The most direct application of CLM to LST would entail modeling the probability distribution at the position of the target word given only the preceding words, denoted as $p(w_x \mid w_1^{x-1})$, where $w_x \in \mathcal{V}$. However, this would omit w_{x+1}^n, the part of the sentence *after* w_x, which may contain vital information. An example can be found in Appendix B.

We therefore propose **PromptSub**, the first LST method to give CLMs access to the full context of an LST instance. PromptSub uses carefully constructed prompts which allow a CLM to produce a substitute based on the full context, including the target word w_x itself.

The following prompt template illustrates how a CLM can be fine-tuned for LST:

The "w_x" in "S" can be substituted with "y". </s>

where S is the input sentence, w_x the target word, and y a selected gold substitute. The underlined part is what the decoder-only model is fine-tuned to predict. Formally, given an LST instance, we construct a prompt s by filling in the placeholders in a prompt template with w_x and S. This reconstruction allows us to reframe LST as CLM, where the objective is to model the probability of: $p(s_{z+1}^{z+4} \mid s_1, \cdots, \{w_x\}, \cdots, \{S\}, \cdots, s_z)$. To ensure the generation of appropriate substitutes, we fix the last five tokens as follows:

- s_z: an open quotation mark
- s_{z+1}: a sampled gold substitute y
- s_{z+2}: a close quotation mark
- s_{z+3}: a period
- s_{z+4}: the end of sentence symbol </s>

Using static quotation marks and a period effectively aids in extracting the eventual substitutes from the generated text. In practice, we notice no adverse effects on loss or performance, and outputs always reliably incorporate these punctuation marks before the sentence concludes.

We then fine-tune the decoder-only model to specifically minimize the cross-entropy loss on s_{z+1}, s_{z+2}, s_{z+3}, and s_{z+4}, where s_{z+4} is included for the model to learn the end of inference. We can then generate a list of potential substitutes $\hat{y}$ by sampling from the probability distribution at $s_{z+1} \in \mathcal{V}$. Consider again our LST example from Section 3.1. We construct the filled prompt as follows:

The "begin" in "let me begin again." can be substituted with "start". </s>

3.3 Sampling strategy

Generating a corpus from an LST dataset for fine-tuning CLMs is not straightforward, since LST instances often have multiple substitute options (often ranked), creating many choices for verbalizing these instances. We therefore introduce two sampling strategies, described below:

TopSub selects only the top-ranked substitute. By doing so, we aim to capture the most probable and relevant substitute for the given context.

FreqSub exploits the frequency information associated with gold substitutes in LST datasets, where frequency is determined by the number of annotations in agreement for each substitute. These frequencies, gathered during the dataset annotation process, are often overlooked in previous methods. Applying a softmax function to these frequencies creates a probability distribution over the gold substitutes, reflecting their likelihood of selection. We then sample one substitute from this distribution, ensuring the model encounters substitutes in proportion to their data-driven frequencies.

3.4 Prompt engineering

This section outlines the prompt engineering for corpus construction, grounded in integrating contextual information into the templates. From an informational standpoint, we operate under the assumption that enriching prompts with more relevant information leads to improved outcomes. Instead of manual, iterative adjustments, we focus on demonstrating the impact of prompts by contrasting several distinct variants. Examples of each template, filled with a single shared LST instance, can be found in Table 1.

BaseP, shown in Figure 1, is the basic prompt template from Section 3.2. It provides the model only the target word and its context. It serves as the foundation for developing more complex prompts.

InfoP seeks to harness the comprehensive information available in LST data. Traditional approaches often focus on the target word and its immediate context, while LST instances often provide additional details that can be valuable. In InfoP, as exemplified in Figure 2, we incorporate three additional attributes of the target word: its position in the sentence, its part of speech (PoS) tag, and its lemma form. These additions work as enriched contextual cues, guiding the model to produce more appropriate substitutions. It is important to note that these attributes utilized in InfoP are

Instance:	let me begin again.
BaseP:	the "begin" in the sentence "let me begin again." can be substituted with "start".
InfoP:	at position 3 in the sentence, "let me begin again.", the verb "begin", derived from the lemma "begin", can be substituted with "start".
AugP (Train):	at position 3 in the sentence, "let me begin again.", the verb "begin", derived from the lemma "begin", can be substituted with "start", "commence", "open", "bring about", "carry on", "initiate", "introduce", "originate", "restart", "try".
AugP (Test):	at position 3 in the sentence, "let me begin again.", the verb "begin", derived from the lemma "begin", can be best substituted with "start".
ExP (Train):	at position 3 in the sentence, "let me begin again.", the verb "begin", derived from the lemma "begin" with synonyms "commence", "get", "get down", "lead off", "set about", "set out", "start", "start out", can be substituted with "start", "commence", "open", "bring about", "carry on", "initiate", "introduce", "originate", "restart", "try".
ExP (Test):	at position 3 in the sentence, "let me begin again.", the verb "begin", derived from the lemma "begin" with synonyms "commence", "get", "get down", "lead off", "set about", "set out", "start", "start out", can be best substituted with "start".

Table 1: Comparative overview of prompting strategies for a given LST instance. Notably, AugP and ExP utilize distinct prompts for training and inference phases. The masked sentence portion, highlighted in blue, is used for loss calculation during training and autoregressive decoding in testing.

exclusively derived from the LST datasets, without reliance on external resources. Furthermore, as evidenced in Section 5, PromptSub remains flexible, allowing for the incorporation of external knowledge if needed.

AugP is designed to boost the diversity of the generated corpus by further augmenting InfoP. In LST tasks, there is often a need to delineate both the best or "top-1" substitute, and a list of the top 10 substitutes. We therefore specifically embed the term "best" into the prompt, where only the top-ranked gold substitute is presented. To incorporate multiple possible substitutes, we exclude the word "best", instead including the top 10 gold substitutes, as determined by the weighted sampling strategy, following the open quotation mark s_n. This means multiple $y \in \mathbf{y}$ will occupy the s_{n+1} slot, rather than just one; substitutes are separated by a comma followed by a space. During the training phase, the fine-tuning prompt is drawn randomly from templates that either include or exclude the term "best". For inference, potential substitutes are solely sampled using the "best" prompt, the intuition being that this will help the model to produce substitutes that are not only acceptable but optimal. This strategy offers deeper insights into the efficacy of our approach when melded with advanced prompt techniques, such as prompt augmentation.

4 Experiments

In this section, we describe our empirical comparison of PromptSub to the top-performing previously published LST methods. After brief descriptions of the benchmark datasets (Section 4.1) and our experimental setup (Section 4.2), we proceed with a comparative analysis of PromptSub and GeneSis, two supervised generative approaches (Section 4.3). We then extend this experiment to include more methods and test of the full suite of datasets (Section 4.4). Unless stated otherwise, we apply PromptSub with FreqSub sampling and AugP for corpus generation, as these settings gave the best performance in our development experiments. Further sensitivity analysis will be presented in Section 5.

4.1 Datasets

LST datasets are few in number and small in size, presenting a challenge for supervised approaches. Thus, we adopt the strategy used by Lacerra et al. (2021b) of merging multiple LST resources for fine-tuning and evaluating on the remainder.

LS07 facilitates comparison with GeneSis, as we can directly compare the results reported by the authors to those we obtain using PromptSub. We carefully follow the dataset construction procedure described in the GeneSis paper.

LS14 includes the CoInCo (Kremer et al., 2014) training set combined with LST and TWSI (Biemann, 2012), as well as a subset of SWORDS (se-

Backbone	Size	Method	best	best-m	oot	oot-m	P@1
bart-large	406M	GeneSis	$19.2_{\pm 0.6}$	$31.1_{\pm 1.5}$	$45.7_{\pm 3.7}$	$60.0_{\pm 4.6}$	$47.9_{\pm 1.1}$
		GeneSis+WN	$20.6_{\pm 0.8}$	$33.2_{\pm 1.7}$	$50.0_{\pm 2.4}$	$65.1_{\pm 2.4}$	$49.2_{\pm 1.9}$
gpt2-medium	345M	PromptSub (ours)	$21.4_{\pm 0.2}$	$35.8_{\pm 0.3}$	$50.5_{\pm 0.2}$	$66.2_{\pm 0.4}$	$50.4_{\pm 0.3}$
		PromptSub+ (ours)	$\mathbf{21.5}_{\pm \mathbf{0.2}}$	$\mathbf{35.9}_{\pm \mathbf{0.4}}$	$\mathbf{51.1}_{\pm \mathbf{0.2}}$	$\mathbf{67.0}_{\pm \mathbf{0.5}}$	$\mathbf{50.7}_{\pm \mathbf{0.3}}$

Table 2: Evaluation results on LS07. For all the metrics, the higher, the better. The best are bolded.

lected to avoid overlap with the CoInCo test set). The dataset is divided into training (90%) and validation (10%) splits. The CoInCo test set is provided for testing.

LS21 follows a similar procedure, but with the SWORDS training set combined with LST and TWSI. A section of CoInCo is added, again ensuring no overlap with the SWORDS test set. The dataset is partitioned into 90% for training and 10% for validation. The original SWORDS test set is preserved for evaluation.

4.2 Experimental setup

Per established practices, we evaluate model performance on LS07 and LS14 using the metrics from the SemEval-2007 task (McCarthy and Navigli, 2007). We use `best` and out-of-ten (`oot`), along with their modal variations `best-m` and `oot-m`, to assess the top-1 and top-10 predictions, respectively. These metrics weight the gold substitutes according to their selection frequency by annotators. For the more recent LS21 dataset, we follow the evaluation protocol developed for the SWORDS benchmark (Lee et al., 2021). We use the F^{10} score, the harmonic mean of precision and recall, for the top 10 predictions against both acceptable ($\mathtt{F^{10}_a}$) and conceivable ($\mathtt{F^{10}_c}$) gold substitutes. SWORDS assigns a score to each substitute to indicate its appropriateness, defining acceptable substitutes as those with scores above 50%, and conceivable substitutes as those with scores above 0%. For thoroughness, we also report a variety of metrics: top-1 precision (`P@1`), top-3 precision (`P@3`), and top-10 recall (`R@10`). Our results, including standard deviations, are averages from five iterations with random seeds 0 to 4.

We utilize GPT-2 (Radford et al., 2019) as our primary CLM. In particular, we use gpt2-medium, except where otherwise specified. This decision stems from constraints related to computational resources and the restrictions on access to more advanced models like GPT-3 (Brown et al., 2020), as well as the desire to compare PromptSub to GeneSis using models with comparable numbers of parameters.

To evaluate the impact of fine-tuning data size on PromptSub, after determining the optimal hyperparameters, we repeat the fine-tuning process on the concatenation of the training and validation sets. We refer to this more fine-tuned variant of PromptSub as PromptSub+. To reiterate, the only distinction between PromptSub and PromptSub+ lies in the training data volume.

To optimize GPU memory utilization on the Nvidia Tesla V100 we use for training, we employ a batch size of 16 with mixed precision training and gradient accumulation. For fine-tuning, we use the AdamW optimizer (Loshchilov and Hutter, 2019) with a learning rate $1e^{-5}$ and an ℓ_2 gradient clipping of 1.0, following Pascanu et al. (2013). To prevent overfitting, we use early stopping with respect to `P@1` on validation for a maximum of 8 epochs (Prechelt, 1998). We set the dropout rate to 0.2, following Srivastava et al. (2014). During inference, we use a beam search with a width of 50, in line with prior methods (Zhou et al., 2019; Lacerra et al., 2021b). All implementations are executed using PyTorch (Paszke et al., 2019), with pre-trained models sourced from the HuggingFace repository (Wolf et al., 2020).

4.3 Experiments on LS07

In evaluating GeneSis, Lacerra et al. (2021b) introduces a set of post-processing steps to the system's output. To ensure a fair comparison, we apply the same post-processing steps to the output of PromptSub. We also test its enhanced variant GeneSis+WN with two extra tricks applied: Fallback strategy (FS) ensures at least ten substitutes are returned by first including previously discarded substitutes and, if necessary, adding more from the vocabulary ranked by cosine similarity to the target, until 10 substitutes are obtained. Vocabulary cut (VC) limits the model to a specified output vocabulary; it discards any generated substitutes outside this vocabulary. It is noteworthy that both FS and

Method	best	best-m	oot	oot-m	P@1
BalAdd	5.6	11.9	20.0	33.8	11.8
SubstituteVector	8.1	17.4	26.7	46.2	–
BERT	14.5	33.9	45.9	69.9	56.3
GeneSis	13.8	30.4	45.6	72.3	58.8
LexSubCon	11.3	23.8	33.6	54.4	41.3
GR-RoBERTa	13.1	28.8	40.9	66.6	48.8
ParaLS	13.8	29.5	41.7	65.6	50.0
ParaLS*	**16.8**	**35.4**	**48.3**	**75.0**	57.8
PromptSub (ours)	14.5	33.1	46.2	72.9	57.7
PromptSub+ (ours)	14.9	33.9	47.0	73.9	**59.5**

Table 3: Evaluation results on LS14. The upper section presents the complete system outcomes, while the lower focuses on the generation step. Results for BalAdd (Melamud et al., 2015b) and SubstituteVector (Melamud et al., 2015a) are sourced from BERT (Zhou et al., 2019). LexSubCon (Michalopoulos et al., 2022), GR-RoBERTa (Lin et al., 2022), ParaLS, and ParaLS* are reported by Qiang et al. (2023). The best are in bold, with the second-best underlined.

Method	F_a^{10}	F_c^{10}
BERT (Zhou et al., 2019)	17.4	27.5
GeneSis (Lacerra et al., 2021b)	23.3	43.0
GPT-3 (Lee et al., 2021)	22.7	36.3
WordTune (Lee et al., 2021)	23.4	33.2
CALS (Yang et al., 2022)	16.7	28.4
mBERT (Wada et al., 2022)	12.4	22.6
SpanBERT (Wada et al., 2022)	20.9	34.0
MPNet (Wada et al., 2022)	22.0	34.1
XLNet (Wada et al., 2022)	23.3	37.4
ELECTRA (Wada et al., 2022)	23.2	36.7
DeBERTa-V3 (Wada et al., 2022)	24.5	39.9
BART (Wada et al., 2022)	23.5	37.2
ParaLS (Qiang et al., 2023)	23.5	38.6
ParaLS* (Qiang et al., 2023)	24.9	40.1
GPT-3 (Lee et al., 2021)	22.2	34.3
WordTune (Lee et al., 2021)	22.8	32.1
BERT (Wada et al., 2022)	20.7	34.4
BERT-K (Wada et al., 2022)	15.7	24.4
BERT-M (Wada et al., 2022)	10.7	16.5
CILex3 (Seneviratne et al., 2022)	19.9	31.5
ParaLS* (Qiang et al., 2023)	22.8	37.0
PromptSub (ours)	23.2	45.4
PromptSub+ (ours)	**24.0**	**46.4**

Table 4: Evaluation results on LS21. The upper section presents the performance of their complete systems, while the lower section reports that of the generation step only. Results of BERT (Zhou et al., 2019), CALS (Yang et al., 2022), and GPT-3 (Lee et al., 2021) are borrowed from Wada et al. (2022). That of CILex3 (Seneviratne et al., 2022) is reported by Qiang et al. (2023). The best are bolded.

VC rely on external resources such as WordNet (Miller, 1995), while PromptSub does not. However, we still incorporate GeneSis+WN for a thorough comparison.

In Table 2, PromptSub outperforms GeneSis across all metrics. Using gpt2-medium, a model with 15% fewer parameters than the bart-large model used by GeneSis, our PromptSub method yields better results, attaining for example 21.5 in `best` and 50.7 in `P@1`. With both FS and VC enabled, GeneSis+WN is still outperformed by PromptSub, even when the former leverages WordNet for post-processing. The results support the hypothesis that PromptSub can benefit from additional training data, as evidenced by the improvements of PromptSub+ over the standard PromptSub.

Another salient point is the pronounced stability exhibited by PromptSub, evident from the reduced variance we observed across random seeds. For instance, PromptSub shows a variance of 0.2, markedly less than the 3.7 of Genesis, in terms of `oot`. This can be attributed to the fact that PromptSub generates substitutes through greedy sampling from a single-step probability distribution, leading to a more stable and consistent output. In contrast, GeneSis relies on multiple decoding steps, resulting in higher variability across runs.

4.4 Experiments on LS14 and LS21

As detailed in Section 4.3, we follow the same evaluation procedure as in GeneSis to ensure a fair comparison. Other competing systems were tested under different experimental configurations, complicating the comparison. Moreover, existing approaches typically involve multiple stages, such as substitute generation and contextualized reranking. This complicates the isolation and evaluation of the specific impact of PromptSub, which is a single-stage end-to-end generative approach, as opposed to the "pipeline" approaches of the methods we compare to. To address this, we expand our evaluation scope to include LS14 and LS21, emphasizing the substitute generation aspect. In Tables 3 and 4, we compare PromptSub (and PromptSub+) to previously published methods on LS14 and LS21 respectively. Results reported in the second part of each table (below the double horizontal line) evaluate performance after the generation stage, with no post-processing. To further verify the advantages of our PromptSub, we re-implement GeneSis using the same configurations, maintaining gpt2-medium for PromptSub and bart-large for GeneSis.

For results on LS14 (Table 3), PromptSub+

yields competitive performance, ranking first or second on all metrics. A standout observation is the prowess of PromptSub+ in the P@1 metric, where it achieves the top result by a wide margin. We find that this disparity between P@1 and other metrics is attributed to annotator preference induced by the weighted task metrics of SemEval-2007 (McCarthy and Navigli, 2007). This thus suggests that certain methods, such as ParaLS*, may be biased toward the substitutes preferred by annotators.

Turning to LS21 (Table 4), both PromptSub and PromptSub+ outperform prior methods, including GeneSis, setting a new state of the art. Specifically, PromptSub+ achieves an unprecedented F^{10}_{c} of 46.4, surpassing the previous best by almost 10. It also achieves the best F^{10}_{a} at 24.0 using PromptSub+. Notably, despite using gpt2-medium, PromptSub and PromptSub+ are able to outperform GPT-3 by a substantial margin, demonstrating the utility of the knowledge-rich prompting techniques we built into PromptSub. Based on these results, we speculate that, with full access to GPT-3 (or even more powerful models), and additional labeled LST data for fine-tuning, PromptSub could yield even stronger results.

4.5 Error examples

In this section, we discuss the most frequent types of errors made by our method.

The first such category that we identified involves instances where the substitutes provided by annotators include phrases rather than single words. For example, one test instance from LS21 has the context "That is why I cannot take payment"; the target word *take* is annotated with substitutions including *accept* and *ask for*. While *accept* is a single element of the vocabulary, *ask for* is a phrase that models trained predominantly to predict single-word substitutes may not generate.

Besides, we observed some potential omissions in the datasets. One example involves substituting the target word *voice* in the context "How should I reply? Her voice had grown quiet". The top prediction of PromptSub, *sound*, is not among the provided substitutes, and so is considered incorrect. However, the annotations include *talk*, *utterance*, and *tongue*, which are, arguably, less suitable as substitutes than *sound*. This highlights the need for benchmarks which are more comprehensive, and which have more consistent criteria for substitutes.

Method	Backbone	LS14				LS21	
		best	best-m	oot	oot-m	F^{10}_{a}	F^{10}_{c}
PromptSub	gpt2	13.8	31.7	43.8	68.8	22.1	42.6
	gpt2-medium	14.5	33.1	46.2	72.9	23.2	45.4
	gpt2-large	14.7	34.5	46.2	72.4	23.8	46.7
PromptSub+	gpt2	14.1	32.4	44.3	69.4	22.8	44.3
	gpt2-medium	14.9	33.9	**47.0**	**73.9**	**24.0**	46.4
	gpt2-large	**15.1**	**34.9**	46.8	72.9	23.8	**47.6**

Table 5: Analysis results of PromptSub on LS14 and LS21, showing the impact of varying model capacity.

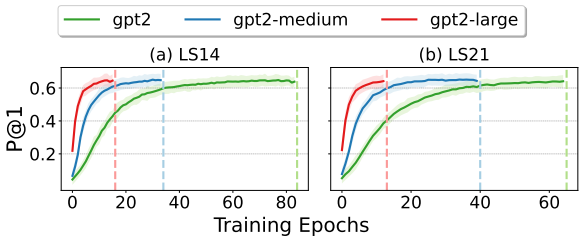

Figure 3: Learning curve of PromptSub for various model sizes on LS14 and LS21 validation sets. Vertical dotted lines indicate the last training epoch before early stopping.

5 Analysis

We now present a sensitivity analysis of our method. We measure the impact of various aspects of our experimental setup, including training size, model capacity, sampling strategy, prompt engineering, and external knowledge. We maintain the same experimental setup, modifying one aspect of our methods to observe the resulting performance change on LS21. The random seed is held constant at 0.

Training size Regarding training data size, we have introduced PromptSub+, a variant of PromptSub, that includes the validation set in its training data. Across all experiments, PromptSub+ consistently outperforms PromptSub on the test data, demonstrating its ability to benefit from additional data. This finding underscores the challenge posed by limited data resources in most existing LST benchmarks, which affects the broader application of PromptSub and other supervised methods (Lacerra et al., 2021a,b).

Model capacity We tested three GPT2 model sizes to measure the impact of model capacity (i.e. number of parameters). As reported in Table 5, the results demonstrate the trend of improved performance, across most evaluation metrics, as we scale from gpt2 to gpt2-medium, then to gpt2-large. Figure 3 depicts the learning curve in relation to model capacity, showing a drop in the number of

Sampling	F_a^{10}	F_c^{10}	P@1	P@3	R@10
TopSub	20.9	39.9	69.4	57.8	40.1
FreqSub	**22.0**	**42.3**	**71.0**	**60.7**	**42.5**

Table 6: Analysis results of gpt2 under PromptSub on LS21, obtained by varying the sampling strategy to fill in the prompt template with label substitutes.

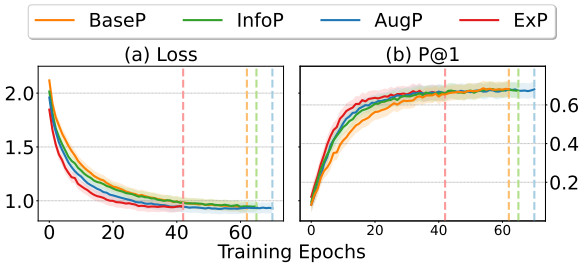

Figure 4: Training dynamics of gpt2 under PromptSub, showing the average loss (a) and P@1 (b) across different prompt templates on the validation set of LS21. Vertical dotted lines mark the early stopping epochs.

Prompt	F_a^{10}	F_c^{10}	P@1	P@3	R@10
BaseP	21.1	37.6	72.7	58.3	38.6
InfoP	20.7	38.4	72.3	59.0	39.5
AugP	**22.1**	42.2	71.9	62.0	**43.6**
ExP	22.0	**42.3**	**73.0**	**62.6**	43.4

Table 7: Analysis results of gpt2 under PromptSub on LS21, obtained by varying the prompt templates.

training epochs before early stopping is triggered. It also becomes apparent that larger models are more prone to overfitting the training set. This trend again reflects the challenge posed by limited data resources in LST, particularly when deploying PLMs in scale.

Sampling strategy In Section 3.3, we considered two different sampling techniques, TopSub and FreqSub. The results obtained by our method with each sampling strategy are presented in Table 6. We observe a constant improvement from TopSub to FreqSub, hence its usage in our principal experiments. These results support that FreqSub successfully addresses the one-to-many mapping issue during corpus generation and facilitates the generation of more accurate and diverse substitutes.

Prompt engineering We next quantify the impact of different prompt templates (Section 3.4) on PromptSub. Table 7 shows that InfoP generally outperforms BaseP, validating the value of extra contextualized cues. AugP outperforms both, aligning with our expectations as the information provided to the language model by AugP is a superset of what InfoP provides. This comparison effectively serves as an ablation study, showcasing the significance of incorporating additional knowledge into prompts. Interestingly, although augmented prompt templates are not used during inference, their inclusion in the training phase still leads to noticeable performance improvements.

External knowledge To validate the efficacy of incorporating external knowledge in PromptSub, we introduce a new prompting strategy, ExP, as a simple form of retrieval-augmented generation (RAG; Lewis et al., 2020b). Building upon AugP, ExP utilizes WordNet as an external knowledge base, retrieving WordNet synsets for the word to be substituted, and which share the same part of speech. These synsets are integrated into the prompt templates as descriptions, following a similar approach to that used for other information. Comparison with AugP in Table 7 reveals the superiority of ExP in P@1 and P@3, indicating that high-quality substitutes are more likely to be near the top of the list produced by PromptSub. Training results in Figure 4 also demonstrate the advantages of ExP, with lower loss, higher P@1, and earlier convergence. These results indicate the potential benefits of grounding PromptSub on external sources of knowledge through RAG.

6 Conclusion

We have presented PromptSub, a framework for recasting LST as CLM, which overcomes the limitations of earlier methods by bridging the gap between pre-training and fine-tuning. PromptSub is flexible and extensible: it allows for variations in the prompt template, facilitating the inclusion of additional knowledge; further analysis reveals the potential for further improvement through scaling up model capacity and data size, applying prompt engineering, and retrieving external knowledge via RAG. Our extensive experiments found that PromptSub consistently outperforms the previous generative approach, GeneSis, on LS07, and establishes a new overall state of the art. As the first attempt to fine-tune decoder-only PLMs for LST, our work highlights the broader applicability of PLMs to semantic tasks.

Limitations

While PromptSub is a significant step forward in LST, it is not without its limitations. Firstly, its effectiveness is limited by the quality and diversity of its training data, a common challenge in supervised methods. This is particularly relevant given the data scarcity in LST, restricting our ability to scale with data extension. Furthermore, PromptSub has not been tested with the latest PLMs due to limited computing resources and closed-source constraints. The computational demands for fine-tuning large-scale language models may limit its practicality, especially in resource-constrained environments.

Acknowledgements

This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), and the Alberta Machine Intelligence Institute (Amii).

References

Nikolay Arefyev, Boris Sheludko, Alexander Podolskiy, and Alexander Panchenko. 2020. Always keep your target in mind: Studying semantics and improving performance of neural lexical substitution. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1242–1255, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Dennis Aumiller and Michael Gertz. 2022. UniHD at TSAR-2022 shared task: Is compute all we need for lexical simplification? In *Proceedings of the Workshop on Text Simplification, Accessibility, and Readability (TSAR-2022)*, pages 251–258, Abu Dhabi, United Arab Emirates (Virtual). Association for Computational Linguistics.

Chris Biemann. 2012. Turk bootstrap word sense inventory 2.0: A large-scale resource for lexical substitution. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 4038–4042, Istanbul, Turkey. European Language Resources Association (ELRA).

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.

Dallas Card. 2023. Substitution-based semantic change detection using contextual embeddings. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 590–602, Toronto, Canada. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Matan Eyal, Shoval Sadde, Hillel Taub-Tabib, and Yoav Goldberg. 2022. Large scale substitution-based word sense induction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4738–4752, Dublin, Ireland. Association for Computational Linguistics.

Aina Garí Soler, Anne Cocos, Marianna Apidianaki, and Chris Callison-Burch. 2019. A comparison of context-sensitive models for lexical substitution. In *Proceedings of the 13th International Conference on Computational Semantics - Long Papers*, pages 271–282, Gothenburg, Sweden. Association for Computational Linguistics.

Samer Hassan, Andras Csomai, Carmen Banea, Ravi Sinha, and Rada Mihalcea. 2007. UNT: SubFinder: Combining knowledge sources for automatic lexical substitution. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 410–413, Prague, Czech Republic. Association for Computational Linguistics.

Bradley Hauer and Grzegorz Kondrak. 2023. Taxonomy of problems in lexical semantics. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 9833–9844, Toronto, Canada.

Gerold Hintz and Chris Biemann. 2016. Language transfer learning for supervised lexical substitution. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 118–129, Berlin, Germany. Association for Computational Linguistics.

Bairu Hou, Fanchao Qi, Yuan Zang, Xurui Zhang, Zhiyuan Liu, and Maosong Sun. 2020. Try to substitute: An unsupervised Chinese word sense disambiguation method based on HowNet. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1752–1757, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Gerhard Kremer, Katrin Erk, Sebastian Padó, and Stefan Thater. 2014. What substitutes tell us - analysis of an "all-words" lexical substitution corpus. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 540–549, Gothenburg, Sweden. Association for Computational Linguistics.

Caterina Lacerra, Tommaso Pasini, Rocco Tripodi, and Roberto Navigli. 2021a. Alasca: an automated approach for large-scale lexical substitution. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 3836–3842. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Caterina Lacerra, Rocco Tripodi, and Roberto Navigli. 2021b. GeneSis: A Generative Approach to Substitutes in Context. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10810–10823, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Mina Lee, Chris Donahue, Robin Jia, Alexander Iyabor, and Percy Liang. 2021. Swords: A benchmark for lexical substitution with improved data coverage and quality. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4362–4379, Online. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020b. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Dianqi Li, Yizhe Zhang, Hao Peng, Liqun Chen, Chris Brockett, Ming-Ting Sun, and Bill Dolan. 2021. Contextualized perturbation for textual adversarial attack. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5053–5069, Online. Association for Computational Linguistics.

Yu Lin, Zhecheng An, Peihao Wu, and Zejun Ma. 2022. Improving contextual representation with gloss regularized pre-training. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 907–920, Seattle, United States. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Diana McCarthy. 2002. Lexical substitution as a task for WSD evaluation. In *Proceedings of the ACL-02 Workshop on Word Sense Disambiguation: Recent Successes and Future Directions*, pages 089–115. Association for Computational Linguistics.

Diana McCarthy and Roberto Navigli. 2007. SemEval-2007 task 10: English lexical substitution task. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 48–53, Prague, Czech Republic. Association for Computational Linguistics.

Oren Melamud, Ido Dagan, and Jacob Goldberger. 2015a. Modeling word meaning in context with substitute vectors. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 472–482, Denver, Colorado. Association for Computational Linguistics.

Oren Melamud, Omer Levy, and Ido Dagan. 2015b. A simple word embedding model for lexical substitution. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 1–7, Denver, Colorado. Association for Computational Linguistics.

George Michalopoulos, Ian McKillop, Alexander Wong, and Helen Chen. 2022. LexSubCon: Integrating knowledge from lexical resources into contextual embeddings for lexical substitution. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1226–1236, Dublin, Ireland. Association for Computational Linguistics.

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

Talgat Omarov and Grzegorz Kondrak. 2023. Grounding the lexical substitution task in entailment. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 2854–2869, Toronto, Canada. Association for Computational Linguistics.

OpenAI. 2023. ChatGPT. `https://openai.com/chatgpt`. Version 4.0.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, page III–1310–III–1318. JMLR.org.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca

Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Lutz Prechelt. 1998. Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer.

Jipeng Qiang, Kang Liu, Yun Li, Yunhao Yuan, and Yi Zhu. 2023. ParaLS: Lexical substitution via pretrained paraphraser. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3731–3746, Toronto, Canada. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. Technical report, OpenAI.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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.

Sandaru Seneviratne, Elena Daskalaki, Artem Lenskiy, and Hanna Suominen. 2022. CILex: An investigation of context information for lexical substitution methods. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4124–4135, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

György Szarvas, Chris Biemann, and Iryna Gurevych. 2013a. Supervised all-words lexical substitution using delexicalized features. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1131–1141, Atlanta, Georgia. Association for Computational Linguistics.

György Szarvas, Róbert Busa-Fekete, and Eyke Hüllermeier. 2013b. Learning to rank lexical substitutions. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1926–1932, Seattle, Washington, USA. Association for Computational Linguistics.

Takashi Wada, Timothy Baldwin, Yuji Matsumoto, and Jey Han Lau. 2022. Unsupervised lexical substitution with decontextualised embeddings. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4172–4185, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Xi Yang, Jie Zhang, Kejiang Chen, Weiming Zhang, Zehua Ma, Feng Wang, and Nenghai Yu. 2022. Tracing text provenance via context-aware lexical substitution. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(10):11613–11621.

Wangchunshu Zhou, Tao Ge, Ke Xu, Furu Wei, and Ming Zhou. 2019. BERT-based lexical substitution. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3368–3373, Florence, Italy. Association for Computational Linguistics.

A PromptSub vs. T5

Our approach distinguishes itself from T5 (Raffel et al., 2020) in several key aspects:

Architecture. Unlike the encoder-decoder framework of T5, PromptSub leverages a decoder-only model to reframe LST as CLM, taking advantage of its inherent strengths in generating text.

Prompting. T5 utilizes a short prefix to specify each task. One example is "cola sentence: " for the CoLA dataset. In contrast, PromptSub employs in-context placeholder prompts that not only verbalize raw LST data instances but also provide a descriptive context for CLM.

Method. The text-to-text format has inherent limitations, thus, aside from GeneSis, there has been no effective method to address LST within this framework. PromptSub, however, offers a fresh perspective and demonstrates a new solution.

Task. PromptSub has successfully adapted causal language models to LST, a domain where, to the best of our knowledge, T5 has not been demonstrated to operate.

Performance. Empirical evidence show that PromptSub outperforms GeneSis, which takes BART as the backbone. Given that GeneSis uses an encoder-decoder framework akin to T5, it stands to reason that PromptSub could extend its advantages over approaches that merely transition from BART to T5.

B MLM & CLM

Consider the following example illustrating the direct application of MLM and CLM to LST:

- Sentence: I live in a beautiful house .
- MLM: I live in a [MASK] house .
- CLM: I live in a [MASK] [MASK] [MASK]

The target word (i.e., "beautiful") is masked for the model to predict it, potentially leading to a substitute (e.g., "big") that fits the context but does not preserve the original sentence semantics due to the absence of the target word information.