# POINTER: Constrained Progressive Text Generation via Insertion-based Generative Pre-training

**Yizhe Zhang**[1]* **Guoyin Wang**[2]*† **Chunyuan Li**[1]
**Zhe Gan**[1] **Chris Brockett**[1] **Bill Dolan**[1]
[1]Microsoft Research, Redmond, WA, USA
[2]Amazon Alexa AI, Seattle, WA, USA
{yizzhang,chunyl,zhe.gan,chrisbkt,billdol}@microsoft.com, guoyiwan@amazon.com

## Abstract

Large-scale pre-trained language models, such as BERT and GPT-2, have achieved excellent performance in language representation learning and free-form text generation. However, these models cannot be directly employed to generate text under specified lexical constraints. To address this challenge, we present POINTER[1], a simple yet novel insertion-based approach for hard-constrained text generation. The proposed method operates by progressively inserting new tokens between existing tokens in a parallel manner. This procedure is recursively applied until a sequence is completed. The resulting coarse-to-fine hierarchy makes the generation process intuitive and interpretable. We pre-train our model with the proposed progressive insertion-based objective on a 12GB Wikipedia dataset, and fine-tune it on downstream hard-constrained generation tasks. Non-autoregressive decoding yields an empirically logarithmic time complexity during inference time. Experimental results on both News and Yelp datasets demonstrate that POINTER achieves state-of-the-art performance on constrained text generation. We released the pre-trained models and the source code to facilitate future research [2].

## 1 Introduction

Real-world editorial assistant applications must often generate text under specified lexical constraints, for example, convert a meeting note with key phrases into a concrete meeting summary, recast a user-input search query as a fluent sentence, generate a conversational response using grounding facts (Mou et al., 2016), or create a story using a pre-specified set of keywords (Fan et al., 2018; Yao et al., 2019; Donahue et al., 2020).

Generating text under specific lexical constraints is challenging. *Constrained* text generation broadly falls into two categories, depending on whether inclusion of specified keywords in the output is mandatory. In *soft-constrained* generation (Qin et al., 2019; Tang et al., 2019), keyword-text pairs are typically first constructed (sometimes along with other conditioning information), and a conditional text generation model is trained to capture their co-occurrence, so that the model learns to incorporate the constrained keywords into the generated text. While *soft-constrained* models are easy to design, even remedied by soft enforcing algorithms such as attention and copy mechanisms (Bahdanau et al., 2015; Gu et al., 2016; Chen et al., 2019a), keywords are still apt to be lost during generation, especially when multiple weakly correlated keywords must be included.

*Hard-constrained* generation (Hokamp and Liu, 2017; Post and Vilar, 2018; Hu et al., 2019; Miao et al., 2019; Welleck et al., 2019), on the other hand, requires that all the lexical constraints be present in the output sentence. This approach typically involves sophisticated design of network architectures. Hokamp and Liu (2017) construct a lexical-constrained grid beam search decoding algorithm to incorporate constraints. However, Hu et al. (2019) observe that a naive implementation of this algorithm has a high running time complexity. Miao et al. (2019) introduces a sampling-based conditional generation method, where the constraints are first placed in a template, then words in a random position are either inserted, deleted or updated under a Metropolis-Hastings-like scheme. However, individually sampling each token results in slow convergence, as the joint distribution of all the tokens in a sentence is highly correlated. Welleck et al. (2019) propose a tree-based text generation scheme, where a token is first generated in an arbitrary position, and then the model recursively

*These authors contributed equally to this work.
†Work was done while Guoyin was at Microsoft.
[1]**PrOgressive INsertion-based TransformER**
[2]https://github.com/dreasysnail/POINTER



*Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 8649–8670,
November 16–20, 2020. 

| Stage | Generated text sequence |
|---|---|
| 0 ($X^0$) | sources sees structure perfectly |
| 1 ($X^1$) | sources **company** sees **change** structure perfectly **legal** |
| 2 ($X^2$) | sources **suggested** company sees **reason** change **tax** structure **which** perfectly legal **.** |
| 3 ($X^3$) | **my** sources **have** suggested **the** company sees **no** reason **to** change **its** tax structure **,** which **are** perfectly legal . |
| 4 ($X^4$) | my sources have suggested the company sees no reason to change its tax structure , which are perfectly legal . |

Table 1: Example of the progressive generation process with multiple stages from the POINTER model. Words in **blue** indicate newly generated words at the current stage. $X^i$ denotes the generated partial sentence at Stage $i$. $X^4$ and $X^3$ are the same indicates the end of the generation process. Interestingly, our method allows informative words (*e.g.*, *company, change*) generated before the non-informative words (*e.g.*, *the, to*) generated at the end.

generates words to its left and right, yielding a binary tree. However, the constructed tree may not reflect the progressive hierarchy/granularity from high-level concepts to low-level details. Further, the time complexity of generating a sentence is $\mathcal{O}(n)$, like standard auto-regressive methods.

Motivated by the above, we propose a novel non-autoregressive model for hard-constrained text generation, called POINTER (**PrO**gressive **IN**sertion-based **T**ransform**ER**). As illustrated in Table 1, generation of words in POINTER is *progressive*, and *iterative*. Given lexical constraints, POINTER first generates high-level words (*e.g.*, nouns, verbs and adjectives) that bridge the keyword constraints, then these words are used as pivoting points at which to insert details of finer granularity. This process iterates until a sentence is finally completed by adding the least informative words (typically pronouns and prepositions).

Due to the resemblance to the masked language modeling (MLM) objective, BERT(Devlin et al., 2019) can be naturally utilized for initialization. Further, we perform large-scale pre-training on a large Wikipedia corpus to obtain a pre-trained POINTER model that which can be readily fine-tuned on specific downstream tasks.

The main contributions of this paper are summarized as follows. ($i$) We present POINTER, a novel insertion-based Transformer model for hard-constrained text generation. Compared with previous work, POINTER allows long-term control over generation due to the top-down progressive structure, and enjoys a significant reduction over emperical time complexity from $\mathcal{O}(n)$ to $\mathcal{O}(\log n)$ at best. ($ii$) Large-scale pre-training and novel beam search algorithms are proposed to further boost performance. ($iii$) We develop a novel beam search algorithm customized to our approach, further improving the generation quality. ($iv$) Experiments on several datasets across different domains (including News and Yelp) demonstrates the superiority of POINTER over strong baselines. Our approach is simple to understand and implement, yet powerful, and can be leveraged as a building block for future research.

## 2 Related Work

**Language Model Pre-training** Large-scale pre-trained language models, such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), XLNet (Yang et al., 2019), Text-to-text Transformer (Raffel et al., 2019) and ELECTRA (Clark et al., 2020), have achieved great success on natural language understanding benchmarks. GPT-2 (Radford et al., 2018) first demonstrates great potential for leveraging Transformer models in generating realistic text. MASS (Song et al., 2019) and BART (Lewis et al., 2019) propose methods for sequence-to-sequence pre-training. UniLM (Dong et al., 2019) unifies the generation and understanding tasks within a single pre-training scheme. DialoGPT (Zhang et al., 2020) and MEENA (Adiwardana et al., 2020) focus on open-domain conversations. CTRL (Keskar et al., 2019) and Grover (Zellers et al., 2019) guide text generation with pre-defined control codes. In addition, recent work has also investigated how to leverage BERT for conditional text generation (Chen et al., 2019b; Mansimov et al., 2019; Li et al., 2020). To the best of our knowledge, ours is the first large-scale pre-training work for hard-constrained text generation.

**Non-autoregressive Generation** Many attempts have been made to use non-autoregressive models for text generation tasks. For neural machine translation, the promise of such methods mostly lies in their decoding efficiency. For example, Gu et al. (2018) employs a non-autoregressive decoder that generates all the tokens simultaneously. Generation can be further refined with a post-processing step to remedy the conditional independence of the parallel decoding process (Lee et al., 2018; Ghazvininejad et al., 2019; Ma et al., 2019; Sun

et al., 2019; Kasai et al., 2020). Deconvolutional decoders (Zhang et al., 2017; Wu et al., 2019) have also been studied for title generation and machine translation. The Insertion Transformer (Stern et al., 2019; Gu et al., 2019; Chan et al., 2019) is a partially autoregressive model that predicts both insertion positions and tokens, and is trained to maximize the entropy over all valid insertions, providing fast inference while maintaining good performance. Our POINTER model hybridizes the BERT and Insertion Transformer models, inheriting the advantages of both, and generates text in a progressive coarse-to-fine manner.

# 3 Method

## 3.1 Model Overview

Let $X = \{x_0, x_1, \cdots, x_T\}$ denote a sequence of discrete tokens, where each token $x_t \in V$, and $V$ is a finite vocabulary set. For the hard-constrained text generation task, the goal is to generate a complete text sequence $X$, given a set of key words $\hat{X}$ as constraints, where the key words have to be exactly included in the final generated sequence with the same order.

Let us denote the lexical constraints as $X^0 = \hat{X}$. The generation procedure of our method can be formulated as a (progressive) sequence of $K$ stages: $S = \{X^0, X^1, \cdots, X^{K-1}, X^K\}$, such that for each $k \in \{1, \ldots, K\}$, $X^{k-1}$ is a sub-sequence of $X^k$. The following stage can be perceived as a finer-resolution text sequence compared to the preceding stage. $X^K$ is the final generation, under the condition that the iterative procedure is converged (*i.e.*, $X^{K-1} = X^K$).

Table 1 shows an example of our progressive text generation process. Starting from the lexical constraints ($X_0$), at each stage, the algorithm inserts tokens progressively to formulate the target sequence. At each step, at most one new token can be generated between two existing tokens. Formally, we propose to factorize the distribution according to the *importance* (defined later) of each token:

$$p(X) = p(X^0) \prod_{k=1}^{K} p(X^k | X^{k-1}) \qquad (1)$$

where $p(X^k|X^{k-1}) = \prod_{x \in X^k - X^{k-1}} p(x|X^{k-1})$. The more important tokens that form the skeleton of the sentence, such as nouns and verbs, appear in earlier stages, and the auxiliary tokens, such as articles and prepositions, are generated at the later stages. In contrast, the autoregressive model factorizes the joint distribution of $X$ in a standard left-to-right manner, *i.e.*, $p(X) = p(x_0) \prod_{t=1}^{T} p(x_t|x_{<t})$, ignoring the word importance. Though the Insertion Transformer (Stern et al., 2019) attempts to implement the progressive generation agenda in (1), it does not directly address how to train the model to generate important tokens first.

## 3.2 Data Preparation

Designing a loss function so that $(i)$ generating an important token first and $(ii)$ generating more tokens at each stage that would yield a lower loss would be complicated. Instead, we prepare data in a form that eases model training.

The construction of data-instance pairs reverses the generation process. We construct pairs of text sequences at adjacent stages, *i.e.*, $(X^{k-1}, X^k)$, as the model input. Therefore, each training instance $X$ is broken into a consecutive series of pairs: $(X^0, X^1), \cdots, (X^{K-1}, X^K)$, where $K$ is the number of such pairs. At each iteration, the algorithm masks out a proportion of existing tokens $X^k$ to yield a sub-sequence $X^{k-1}$, creating a training instance pair $(X^{k-1}, X^k)$. This procedure is iterated until only less than $c$ ($c$ is small) tokens are left.

Two properties are desired when constructing data instances: $(i)$ important tokens should appear in an earlier stage , so that the generation follows a progressive manner; $(ii)$ the number of stages $K$ is small, thus the generation is fast at inference time.

**Token Importance Scoring** We consider three different schemes to assess the importance score of a token: term frequency-inverse document frequency (TF-IDF), part-of-speech (POS) tagging, and Yet-Another-Keyword-Extractor (YAKE) (Campos et al., 2018, 2020). The TF-IDF score provides the uniqueness and local enrichment evaluation of a token at a corpus level. POS tagging indicates the role of a token at a sequence level. We explicitly assign noun or verb tokens a higher POS tagging score than tokens from other categories. YAKE is a commonly used unsupervised automatic keyword extraction method that relies on statistical features extracted from single documents to select the most important keywords (Campos et al., 2020). YAKE is good at extracting common key words, but relatively weak at extracting special nouns (*e.g.*, names), and does not provide any importance level for non-keyword tokens. Therefore, we combine the above three metrics for token importance scor-

ing. Specifically, the overall score $\alpha_t$ of a token $x_t$ is defined as $\alpha_t = \alpha_t^{\text{TF-IDF}} + \alpha_t^{\text{POS}} + \alpha_t^{\text{YAKE}}$, where $\alpha_t^{\text{TF-IDF}}$, $\alpha_t^{\text{POS}}$ and $\alpha_t^{\text{YAKE}}$ represent the TF-IDF, POS tagging and YAKE scores (each is rescaled to $[0, 1]$), respectively.

Additionally, stop words are manually assigned a low importance score. If a token appears several times in a sequence, the latter occurrences are assigned a decayed importance score to prevent the model from generating the same token multiple times in one step at inference time. We note that our choice of components of the importance score is heuristic. It would be better to obtain an unbiased/oracle assessment of importance, which we leave for future work.

**DP-based Data Pair Construction** Since we leverage the Insertion-based Transformer, which allows at most one new token to be generated between each two existing tokens, sentence length at most doubles at each iteration. Consequently, the optimal number of iterations $K$ is $\log(T)$, where $T$ is the length of the sequence. Therefore, generation efficiency can be optimized by encouraging more tokens to be discarded during each masking step when preparing the data. However, masking positional interleaving tokens ignores token importance, and thus loses the property of progressive planning from high-level concepts to low-level details at inference time. In practice, sequences generated by such an approach can be less semantically consistent as less important tokens occasionally steer generation towards random content.

We design an approach to mask the sequence by considering both token importance and efficiency using dynamic programming (DP). To accommodate the nature of insertion-based generation, the masking procedure is under the constraint that no consecutive tokens can be masked at the same stage. Under such a condition, we score each token and select a subset of tokens that add up to the highest score (all scores are positive). This allows the algorithm to adaptively choose as many high scored tokens as possible to mask.

Formally, as an integer linear programming problem (Richards and How, 2002), the objective is to find an optimal masking pattern $\Phi = \{\phi_1, \cdots, \phi_T\}$, where $\phi_t \in \{0, 1\}$, and $\phi_t = 1$ represents discarding the corresponding token $x_t$, and $\phi_t = 0$ indicates $x_t$ remains. For a sequence $X'$, the objective can be formulated as:

$$\max \sum_{t=1}^{T} \phi_t(\alpha_{\max} - \alpha_t), \quad \text{s.t.} \quad \phi_t \phi_{t+1} \neq 1\,, \forall t \tag{2}$$

where $\alpha_{\max} = \max_t\{\alpha_t\}$. Though solving Eq. (2) is computationally expensive, one can resort to an analogous problem for a solution, the so-called *House Robbery Problem*, a variant of *Maximum Subarray Problem* (Bentley, 1984), where a professional burglar plans to rob houses along a street and tries to maximize the outcome, but cannot break into two adjacent houses without triggering an alarm. This can be solved using dynamic programming (Bellman, 1954) (also known as *Kadane's algorithm* (Gries, 1982)) as shown in Algorithm 1.

**Algorithm 1** DP-based Data Pair Construction.

**Input:** A sequence of discrete tokens $X = \{x_1 \cdots, x_T\}$ and its corresponding score list $\{\alpha_{\max} - \alpha_1, \cdots, \alpha_{\max} - \alpha_T\}$
**Output:** Masking pattern $\Phi = \{\phi_1, \cdots, \phi_T\}$
**Initialization:** Accumulating scores $s_1 \leftarrow \alpha_{\max} - \alpha_1$ and $s_2 \leftarrow \max(\alpha_{\max} - \alpha_1, \alpha_{\max} - \alpha_2)$; position tracker $p_1 \leftarrow -\inf$ and $p_2 \leftarrow -\inf$; $\Phi$ = 0
**while** $t \leq T$ **do**
    $s_t \leftarrow \max(s_{t-2} + \alpha_{\max} - \alpha_t, s_{t-1})$
    **if** $s_t = s_{t-1}$ **then** $p_t \leftarrow t - 1$
    **else** $p_t \leftarrow t - 2$
    **end if**
    $t \leftarrow t + 1$
**end while**
**if** $s_T = s_{T-1}$ **then** $t \leftarrow T - 1$
**else** $t \leftarrow T - 2$, $\phi_T \leftarrow 1$
**end if**
**while** $t \geq 1$ **do**
    $\phi_t \leftarrow 1$,$t \leftarrow p_t$
**end while**

### 3.3 Model Training

**Stage-wise Insertion Prediction** With all the data-instance pairs $(X^{k-1}, X^k)$ created as described above as the model input, we optimize the following objective:

$$\begin{aligned} \mathcal{L} &= -\log p(X^k | X^{k-1}) \\ &= -\sum_{x \in X^+} \log p(x | \Phi^{k-1}, X^{k-1}) p(\Phi^{k-1} | X^{k-1})\,, \end{aligned} \tag{3}$$

where $X^+ \triangleq X^k - X^{k-1}$, and $\Phi^{k-1}$ denotes an indicator vector in the $k$-th stage, representing whether an insertion operation is applied in a slot.

As illustrated in Figure 1, while the MLM objective in BERT only predicts the token of a masked

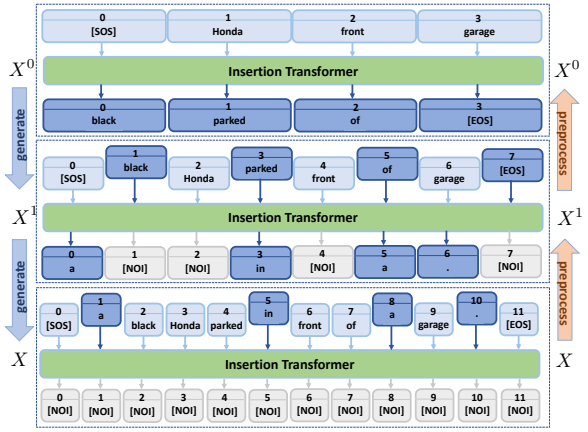


Figure 1: Illustration of the generation process ($X^0 \rightarrow X$) of the proposed POINTER model. At each stage, the Insertion Transformer module generates either a regular token or a special [NOI] token for each gap between two existing tokens. The generation stops when all the gaps predict [NOI]. The data preparation process reverses the above generative process.

placeholder, our objective comprises both ($i$) likelihood of an insertion indicator for each slot (between two existing tokens), and ($ii$) the likelihood of each new token conditioning on the activated slot. To handle this case, we expand the vocabulary with a special no-insertion token [NOI]. During inference time, the model can predict either a token from the vocabulary to insert, or an [NOI] token indicating no new token will be inserted at a certain slot at the current stage. By utilizing this special token, the two objectives are merged. Note that the same insertion transformer module is re-used at different stages. We empirically observed that the model can learn to insert different words at different stages; it presumably learns from the completion level (how discontinuous the context is) of the current context sequence to roughly estimate the progress up to that point.

During inference time, once in a stage ($X^k$), all the slots predict [NOI] for the next stage, the generation procedure is converged and $X^k$ is the final output sequence. Note that to account for this final stage $X^k$, during data preparation we incorporate an $(X, N)$ pair for each sentence in the training data, where $N$ denotes a sequence of [NOI] with the same length of $X$. To enable the model to insert at the beginning and end of the sequence, an [SOS] token and an [EOS] token are added in the beginning and at the end of each sentence, respectively.

In light of the similarity with the MLM objective, we use BERT model to initialize the Insertion Transformer module.

**Large-scale Pre-training** In order to provide a general large-scale pretrained model that can benefit various downstream tasks with fine-tuning, we train a model on the massive publicly available English Wiki dataset, which covers a wide range of topics. The Wiki dataset is first preprocessed according to Sec. 3.2. We then initialize the model with BERT, and perform model training on the processed data using our training objective (3). After pre-training, the model can be used to generate an appropriate sentence with open-domain keyword constraints, in a tone that represents the Wiki style. In order to adapt the pre-trained model to a new domain (*e.g.*, News and Yelp reviews), the pre-trained model is further fine-tuned on new datasets, which empirically demonstrates better performance than training the model on the target domain alone.

### 3.4 Inference

During inference time, starting from the given lexical constraint $X^0$, the proposed model generates text stage-by-stage using greedy search or top-K sampling (Fan et al., 2018), by applying the Insertion Transformer module repeatedly until no additional token is generated. If a [NOI] token is generated, it is deleted at the next round.

**Inner-Layer Beam Search** According to (3), all new tokens are simultaneously generated based on the existing tokens at the previous stage. Despite of being fully parallel, like BERT (Yang et al., 2019) and NAT (Ghazvininejad et al., 2019; Kasai et al., 2020) this approach suffers from a conditional independence problem in which the predicted tokens are conditional-independently generated and are agnostic of each other. This can result in generating repeating or inconsistent new tokens at each generation round.[3]

To address this weak-dependency issue, we perform a modified beam search algorithm for decoding. Specifically, at stage $k$, suppose the existing tokens from last stage are $X^{k-1} = \{x_1^{k-1}, \cdots, x_{T_{k-1}}^{k-1}\}$, where $T_{k-1}$ is the length of $X^{k-1}$. For predicting next stage $X^k$, there will be $T_{k-1}$ available slots. A naive approach to perform beam search would be to maintain a priority queue of top $B$ candidate token series predictions when moving from the leftmost slot to the rightmost slot. At the $t$-th move, the priority queue contains top $B$ sequences

[3]For example, from an existing token "*and*", the model generates "*clean and clean*".

for existing predicted tokens: $(s_1^{(b)}, \cdots, s_{t-1}^{(b)})$, where $s_i^{(b)}$ denotes the predicted token for the $i$-th slot in the $b$-th ($b \in \{1, \cdots, B\}$) sequence. The model then evaluates the likelihood of each item (including [NOI]) in the vocabulary for the slot $s_t$, by computing the likelihood of $(s_1^{(b)}, x_1^{k-1}, \cdots, s_{t-1}^{(b)}, x_{t-1}^{k-1}, s_t, x_t^{k-1}, [\texttt{NOI}], \cdots, [\texttt{NOI}], x_{T_{k-1}}^{k-1})$. This is followed by a ranking step to select the top $B$ most likely series among the $VB$ series to grow. However, such a naive approach is expensive, as the runtime complexity takes $\mathcal{O}(TBV)$ evaluations.

Instead, we approximate the search by constraining it in a narrow band. We design a customized beam search algorithm for our model, called *inner-layer beam search* (ILBS). This method applies an approximate local beam search at each iteration to find the optimal stage-wise decoding. At the $t$-th slot, ILBS first generates top $B$ token candidates by applying one evaluation step based on existing generation. Prediction is limited to these top $B$ token candidates, and thus the beam search procedure as described above is applied on the narrow band of $B$ instead of the full vocabulary $V$. This reduces the computation to $\mathcal{O}(TB^2)$.

## 4 Experiments

We evaluate the POINTER model on constrained text generation over News and Yelp datasets. Details of the datasets and experimental results are provided in the following sub-sections. The pre-trained models and the source code are available at Github [4].

### 4.1 Experimental Setup

**Datasets and Pre-processing** We evaluate our model on two datasets. The *EMNLP2017 WMT News dataset*[5] contains 268,586 sentences, and we randomly pick 10k sentences as the validation set, and 1k sentences as the test set. The *Yelp English review dataset* is from Cho et al. (2018), which contains 160k training examples, 10k validation examples and 1k test examples. These two datasets vary in sentence length and domain, enabling the assessment of our model in different scenarios.

The English Wikipedia dataset we used for pre-training is first pre-processed into a set of natural sentences, with maximum sequence length of 64 tokens, which results in 1.99 million sentences for model training in total (12.6 GB raw text). On average, each sentence contains 27.4 tokens.

For inference, we extract the testing lexical constraints for all the compared methods using the 3rd party extracting tool YAKE[6]. The maximum length of the lexical constraints we used for News and Yelp is set to 4 and 7, respectively, to account the average length for News ($27.9 \approx 4 \times 2^3$) and Yelp ($50.3 \approx 7 \times 2^3$), as we would hope the generation can be done within 4 stages.

**Baselines** We compare our model with two state-of-the-art methods for hard-constrained text generation: ($i$) Non-Monotonic Sequential Text Generation (NMSTG) (Welleck et al., 2019), and ($ii$) Constrained Sentence Generation by Metropolis-Hastings Sampling (CGMH) (Miao et al., 2019). We also compared with an autoregressive soft-constraint baseline(Gao et al., 2020). Note that the Insertion Transformer (Stern et al., 2019) focuses on machine translation rather than hard-constrained generation task, and therefore is not considered for comparison. Other methods based on grid beam search typically have long inference time, and they only operate on the inference stage; these are also excluded from comparison. For all compared system, we use the default settings suggested by the authors, the models are trained until the evaluation loss does not decrease. More details are provided in the Appendix.

**Experiment Setups** We employ the tokenizer and model architecture from BERT-base and BERT-large models for all the tasks. BERT models are used as our model initialization. Each model is trained until the validation loss is no longer decreasing. We use a learning rate of 3e-5 without any warming-up schedule for all the training procedures. The optimization algorithm is Adam (Kingma and Ba, 2015). We pre-train our model on the Wiki dataset for 2-4 epochs, and fine-tune on the News and Yelp datasets for around 10 epochs.

**Evaluation Metrics** Following Zhang et al. (2020), we perform automatic evaluation using commonly adopted text generation metrics, including BLEU (Papineni et al., 2002), METEOR (Lavie and Agarwal, 2007), and NIST (Doddington, 2002). Following (Kann et al., 2018), to assess the coherence of generated sentences, we also report the perplexity over the test set using pre-trained GPT-2

[4] https://github.com/dreasysnail/POINTER

[5] http://www.statmt.org/wmt17/

[6] https://github.com/LIAAD/yake

| News dataset Method | NIST N-2 | NIST N-4 | BLEU B-2 | BLEU B-4 | METEOR | Entropy E-4 | Dist D-1 | Dist D-2 | PPL. | Avg. Len. |
|---|---|---|---|---|---|---|---|---|---|---|
| CGMH | 1.60 | 1.61 | 7.09% | 1.61% | 12.55% | 9.32 | **16.60%** | **70.55%** | 189.1 | 14.29 |
| NMSTG | 2.70 | 2.70 | 10.67% | 1.58% | 13.56% | 10.10 | 11.09% | 65.96% | 171.0 | 27.85 |
| Greedy (base) | 2.90 | 2.80 | 12.13% | 1.63% | 15.66% | **10.41** | 5.89% | 39.42% | 97.1 | 47.40 |
| Greedy (+Wiki,base) | 3.04 | 3.06 | 13.01% | 2.51% | **16.38%** | 10.22 | 11.10% | 57.78% | 56.7 | 31.32 |
| ILBS (+Wiki,base) | 3.20 | 3.22 | 14.00% | 2.99% | 15.71% | 9.86 | 13.17% | 61.22% | 66.4 | 22.59 |
| Greedy (+Wiki, large) | **3.28** | **3.30** | **14.04%** | **3.04%** | 15.90% | 10.09 | 12.23% | 60.86% | **54.7** | 27.99 |
| Human oracle | - | - | - | - | - | 10.05 | 11.80% | 62.44% | 47.4 | 27.85 |

| Yelp dataset Method | NIST N-2 | NIST N-4 | BLEU B-2 | BLEU B-4 | METEOR | Entropy E-4 | Dist D-1 | Dist D-2 | PPL. | Avg. Len. |
|---|---|---|---|---|---|---|---|---|---|---|
| CGMH | 0.50 | 0.51 | 4.53% | 1.45% | 11.87% | 9.48 | **12.18%** | **57.10%** | 207.2 | 16.70 |
| NMSTG | 1.11 | 1.12 | 10.06% | 1.92% | 13.88% | 10.09 | 8.39% | 50.80% | 326.4 | 27.92 |
| Greedy (base) | 2.15 | 2.15 | 11.48% | 2.16% | **17.12%** | **11.00** | 4.19% | 31.42% | 99.5 | 87.30 |
| Greedy (+Wiki,base) | 3.27 | 3.30 | 15.63% | 3.32% | 16.14% | 10.64 | 7.51% | 46.12% | 71.9 | 48.22 |
| ILBS (+Wiki,base) | 3.34 | 3.38 | 16.68% | 3.65% | 15.57% | 10.44 | 9.43% | 50.66% | 61.0 | 35.18 |
| Greedy (+Wiki, large) | **3.49** | **3.53** | **16.78%** | **3.79%** | 16.69% | 10.56 | 6.94% | 41.2% | **55.5** | 48.05 |
| Human oracle | - | - | - | - | - | 10.70 | 10.67% | 52.57% | 55.4 | 50.36 |

Table 2: Automatic evaluation results on the News (upper) and Yelp (lower) dataset. ILBS denotes beam search. "+Wiki" denotes fine-tuning on the Wiki-pretrained model. "base/large" represents the greedy generation from a based(110M)/large(340M) model. "Human" represents the held-out human reference.

medium (large) model[7]. We use Entropy (Zhang et al., 2018) and Dist-n (Li et al., 2016) to evaluate lexical diversity.

| Keywords | **estate pay stay policy** |
|---|---|
| CGMH | an economic **estate** developer that could **pay** for it is that a **stay policy** . |
| NMSTG | as **estate** owners , they cannot **pay** for households for hundreds of middle - income property , buyers **stay** in retail **policy** . |
| POINTER (Greedy, base) | if you buy new buildings from real **estate** company, you may have to **pay** down a mortgage and **stay** with the **policy** for financial reasons . |
| POINTER (ILBS, base) | but no matter what foreign buyers do , real **estate** agents will have to **pay** a small fee to **stay** consistent with the **policy** . |
| POINTER (Greedy, Large) | but it would also be required for **estate** agents , who must **pay** a larger amount of cash but **stay** with the same **policy** for all other assets . |

Table 3: Generated examples from the News dataset.

| Keywords | **joint great food great drinks greater staff** |
|---|---|
| CGMH | very cool **joint** with **great food** , **great drinks** and even **greater staff** . ! . |
| NMSTG | awesome **joint** . **great** service. **great food** great **drinks**. good to **greater** and great **staff**! |
| POINTER (Greedy, base) | my favorite local **joint** around old town. **great** atmosphere, amazing **food**, delicious and delicious coffee, **great** wine selection and delicious cold **drinks**, oh and maybe even a **greater** patio space and energetic front desk **staff**. |
| POINTER (ILBS, base) | the best breakfast **joint** in charlotte . **great** service and amazing **food** . they have **great** selection of **drinks** that suits the **greater** aesthetic of the **staff** . |
| POINTER (Greedy, Large) | this is the new modern breakfast **joint** to be found around the area . **great** atmosphere , central location and excellent **food** . nice variety of selections . **great** selection of local craft beers , good **drinks** . quite cheap unless you ask for **greater** price . very friendly patio and fun **staff** . love it ! |

Table 4: Generated examples from the Yelp dataset.

## 4.2 Experimental Results

**News Generation** We first conduct experiments on the News dataset to generate sentences from 4 lexical constraints. Quantitative results are summarized in Table 2 (upper). Some qualitative examples including the progressive generations at each stage are provided in Table 3 and Appendix B. POINTER is able to take full advantage of BERT initialization and Wiki pre-training to improve relevance scores (NIST, BLEU and METEOR). Leveraging the ILBS or using a larger model further improves most automatic metrics we evaluated [8]. For diversity scores, as CGMH is a sampling-based method in nature, it achieves the highest Dist-n scores (even surpasses human score). We observed that the length of generated sentences, the diversity scores and the GPT-2 perplexity from POINTER are close to human oracle.

**Yelp Generation** We further evaluate our method

[7] https://github.com/openai/gpt-2

[8] The ILBS for larger models performs similarly to greedy decoding, and thus is omitted from comparison

| **Semantics**: *A and B, which is more semantically meaningful and consistent?* | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| News dataset | | | | | Yelp dataset | | | | |
| System A | | Neutral | System B | | System A | | Neutral | System B | |
| POINTER(base) | **60.9%** | 17.4% | 21.8% | CGMH | POINTER(base) | **59.8%** | 17.3% | 23.0% | CGMH |
| POINTER(base) | **55.2%** | 21.7% | 23.1% | NMSTG | POINTER(base) | **57.5%** | 23.0% | 19.6% | NMSTG |
| POINTER(base) | 21.7% | 21.4% | **56.9%** | Human | POINTER(base) | 26.8% | 25.9% | **47.3%** | Human |
| **Fluency**: *A and B, which is more grammatical and fluent?* | | | | | | | | | |
| News dataset | | | | | Yelp dataset | | | | |
| System A | | Neutral | System B | | System A | | Neutral | System B | |
| POINTER(base) | **57.7%** | 19.9% | 22.4% | CGMH | POINTER(base) | **54.2%** | 20.0% | 25.8% | CGMH |
| POINTER(base) | **52.7%** | 24.1% | 23.2% | NMSTG | POINTER(base) | **59.0%** | 22.8% | 18.2% | NMSTG |
| POINTER(base) | 16.6% | 20.0% | **63.4%** | Human | POINTER(base) | 24.0% | 26.1% | **49.9%** | Human |
| **Informativeness**: *A and B, which is more informative?* | | | | | | | | | |
| News dataset | | | | | Yelp dataset | | | | |
| System A | | Neutral | System B | | System A | | Neutral | System B | |
| POINTER(base) | **70.4%** | 12.8% | 16.8 % | CGMH | POINTER(base) | **69.9%** | 10.9% | 19.3 % | CGMH |
| POINTER(base) | **57.7%** | 18.7% | 23.6% | NMSTG | POINTER(base) | **65.2%** | 18.1% | 16.7% | NMSTG |
| POINTER(base) | 31.7% | 19.0% | **49.4%** | Human | POINTER(base) | 32.8% | 19.0% | **48.2%** | Human |

Table 5: **Human Evaluation** on two datasets for semantic consistency, fluency and informativeness, showing preferences (%) for our POINTER(base) model vis-à-vis baselines and real human responses. Numbers in bold indicate the most preferred systems. Differences in mean preferences are statistically significant at $p \leq 0.00001$.

on the Yelp dataset, where the goal is to generate a long-form text from more constraints. Generating a longer piece of text with more lexical constraints is generally more challenging, since the model needs to capture the long-term dependency structure from the text, and effectively conjure up with a plan to realize the generation. Results of automatic evaluation are provided in Table 2 (lower). Generated examples are shown in Table 4 and Appendix C. Generally, the generation from our model effectively considers all the lexical constraints, and is semantically more coherent and grammatically more fluent, compared with the baseline methods. The automatic evaluation results is generally consistent with the observations from News dataset, with an exception that Dist-n scores is much lower than the human Dist-n scores. Compared with greedy approach, at a cost of efficiency, ILBS is typically more concise and contains less repeated information, a defect the greedy approach occasionally suffers (e.g., Table 4, "delicious and delicious").

For both datasets, most of the generations converges with in 4 stages. We perform additional experiments on zero-shot generation from the pre-trained model on both datasets, to test the versatility of pre-training. The generated sentences, albeit Wiki-like, are relatively fluent and coherent (see examples in Appendix B and C), and yield relatively high relevance scores (see Appendix E for details). Interestingly, less informative constraints are able to be expanded to coherent sentences. Given the constraint is to from, our model generates "it is oriented to its east, but from the west".

The autoregressive soft-constraint baseline(Gao et al., 2020) has no guarantee that it will cover all keywords in the given order, thus we omit it in the Table 2. For this baseline, the percentage of keywords that appear in the outputs are 57% and 43% for News and Yelp datasets, respectively. With the similar model size (117M), this baselines performance is worse than ours approach in automatic metrics for News dataset (BLEU4: 2.99 $\rightarrow$ 1.74; NIST4: 3.22 $\rightarrow$ 1.10; METEOR: 16% $\rightarrow$ 9%; DIST2: 61% $\rightarrow$ 58%; PPL: 66 $\rightarrow$ 84). The performance gap in Yelp dataset is even larger due to more lexical constraints.

**Human Evaluation** Using a public crowdsourcing platform (UHRS), we conducted a human evaluation of 400 randomly sampled outputs (out of 1k test set) of CGMH, NMSTG and our base and large models with greedy decoding. Systems were paired and each pair of system outputs was randomly presented (in random order) to 5 crowdsourced judges , who ranked the outputs pairwise for coherence, informativeness and fluency using

| Model | Training | Inference |
|---|---|---|
| CGMH | 4382 toks/s | 33h |
| NMSTG | 357 toks/s | 487s |
| POINTER | 5096 toks/s | 94s |

Table 6: Speed comparison. "toks/s" represents tokens per second. Inference time is computed on 1000 test examples. POINTER uses (greedy, base)

a 5-point Likert-like scale. The human evaluation template is provided in Appendix G. The overall judge preferences for fluency, informativeness and semantic coherence are presented as percentages of the total "vote" in Table 5. P-values are all p¡0.00001 (line 721), computed using 10000 bootstrap replications. For inter-annotator agreement, Krippendorff's alpha is 0.23 on the News dataset and 0.18 on the Yelp dataset. Despite the noise, the judgments show a strong across-the-board preference for POINTER(base) over the two baseline systems on all categories. A clear preference for the human ground truth over our method is also observed. The base and large models show comparable human judge preferences on the News dataset, while human judges clearly prefer the large model on Yelp data (see Appendix D for more details).

**Running-time Comparison** One of the motivations of this work is that at each stage the generation can be parallel, leading to a significant reduction in training and inference. We compare the model training time and the inference decoding time of all the methods on the Yelp dataset, and summarize the results in Table 6. The evaluation is based on a single Nvidia V100 GPU. Training time for CGMH and POINTER is relatively fast, while NMSTG processes fewer tokens per second since it needs to generate a tree-like structure for each sentence. With respect to inference time, CGMH is slow, as it typically needs hundreds of sampling iterations to decode one sentence.

We note there is no theoretical guarantee of $\mathcal{O}(\log N)$ time complexity for our method. However, our approach encourages filling as many slots as possible at each stage, which permits enables the model to achieve an empirical $\mathcal{O}(\log N)$ speed. In our experiment 98% of generations end within 4 stages.

Note that our method in Table 6 uses greedy decoding. ILBS is around 20 times slower than greedy. The large model is around 3 times slower than the base model.

## 5 Conclusion

We have presented POINTER, a simple yet powerful approach to generating text from a given set of lexical constraints in a non-autoregressive manner. The proposed method leverages a large-scale pre-trained model (such as BERT initialization and our insertion-based pre-training on Wikipedia) to generate text in a progressive manner using an insertion-based Transformer. Both automatic and human evaluation demonstrate the effectiveness of POINTER. In future work, we hope to leverage sentence structure, such as the use of constituency parsing, to further enhance the design of the progressive hierarchy. Our model can be also extended to allow inflected/variant forms and arbitrary ordering of given lexical constraints.

## References


Daniel Adiwardana, Minh-Thang Luong, David R So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, et al. 2020. Towards a human-like open-domain chatbot. *arXiv preprint arXiv:2001.09977*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Richard Bellman. 1954. The theory of dynamic programming. Technical report, Rand corp santa monica ca.

Jon Bentley. 1984. Programming pearls: algorithm design techniques. *Communications of the ACM*, 27(9):865–873.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Jorge, Célia Nunes, and Adam Jatowt. 2020. Yake! keyword extraction from single documents using multiple local features. *Information Sciences*.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Mário Jorge, Célia Nunes, and Adam Jatowt. 2018. Yake! collection-independent automatic keyword extractor. In *European Conference on Information Retrieval*.

William Chan, Nikita Kitaev, Kelvin Guu, Mitchell Stern, and Jakob Uszkoreit. 2019. Kermit: Generative insertion-based modeling for sequences. *arXiv preprint arXiv:1906.01604*.

Liqun Chen, Yizhe Zhang, Ruiyi Zhang, Chenyang Tao, Zhe Gan, Haichao Zhang, Bai Li, Dinghan Shen, Changyou Chen, and Lawrence Carin. 2019a. Improving sequence-to-sequence learning via optimal transport. In *ICLR*.

Yen-Chun Chen, Zhe Gan, Yu Cheng, Jingzhou Liu, and Jingjing Liu. 2019b. Distilling the knowledge of bert for text generation. *arXiv preprint arXiv:1911.03829*.

Woon Sang Cho, Pengchuan Zhang, Yizhe Zhang, Xiujun Li, Michel Galley, Chris Brockett, Mengdi Wang, and Jianfeng Gao. 2018. Towards coherent and cohesive long-form text generation. *arXiv preprint arXiv:1811.00511*.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*.

Chris Donahue, Mina Lee, and Percy Liang. 2020. Enabling language models to fill in the blanks. *arXiv preprint arXiv:2005.05339*.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *NeurIPS*.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *ACL*.

Xiang Gao, Michel Galley, and Bill Dolan. 2020. Mixingboard: a knowledgeable stylized integrated text generation platform. In *ACL, system demonstration*.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. In *EMNLP*.

David Gries. 1982. A note on a standard strategy for developing loop invariants and loops. *Science of Computer Programming*.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *ICLR*.

Jiatao Gu, Qi Liu, and Kyunghyun Cho. 2019. Insertion-based decoding with automatically inferred generation order. *TACL*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*.

Chris Hokamp and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid beam search. In *ACL*.

J Edward Hu, Huda Khayrallah, Ryan Culkin, Patrick Xia, Tongfei Chen, Matt Post, and Benjamin Van Durme. 2019. Improved lexically constrained decoding for translation and monolingual rewriting. In *NAACL*.

Katharina Kann, Sascha Rothe, and Katja Filippova. 2018. Sentence-level fluency evaluation: References help, but can be spared! *arXiv preprint arXiv:1809.08731*.

Jungo Kasai, James Cross, Marjan Ghazvininejad, and Jiatao Gu. 2020. Parallel machine translation with disentangled context transformer. *arXiv preprint arXiv:2001.05136*.

Nitish Shirish Keskar, Bryan McCann, Lav Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL - A Conditional Transformer Language Model for Controllable Generation. *arXiv preprint arXiv:1909.05858*.

D. Kingma and J. Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Alon Lavie and Abhaya Agarwal. 2007. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. *EMNLP*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Dianqi Li, Yizhe Zhang, Hao Peng, Liqun Chen, Chris Brockett, Ming-Ting Sun, and Bill Dolan. 2020. Contextualized perturbation for textual adversarial attack. *arXiv preprint arXiv:2009.07502*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *NAACL*.

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*.

Xuezhe Ma, Chunting Zhou, Xian Li, Graham Neubig, and Eduard Hovy. 2019. Flowseq: Non-autoregressive conditional sequence generation with generative flow. *arXiv preprint arXiv:1909.02480*.

Elman Mansimov, Alex Wang, and Kyunghyun Cho. 2019. A generalized framework of sequence generation with application to undirected sequence models. *arXiv preprint arXiv:1905.12790*.

Ning Miao, Hao Zhou, Lili Mou, Rui Yan, and Lei Li. 2019. Cgmh: Constrained sentence generation by metropolis-hastings sampling. In *AAAI*.

Lili Mou, Yiping Song, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2016. Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation. *arXiv preprint arXiv:1607.00970*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Matt Post and David Vilar. 2018. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. In *NAACL*.

Lianhui Qin, Michel Galley, Chris Brockett, Xiaodong Liu, Xiang Gao, Bill Dolan, Yejin Choi, and Jianfeng Gao. 2019. Conversing by reading: Contentful neural conversation with on-demand machine reading. In *ACL*.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. 2018. Language models are unsupervised multitask learners. Technical report, OpenAI.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Arthur Richards and Jonathan P How. 2002. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *Proceedings of American Control Conference*.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. *arXiv preprint arXiv:1905.02450*.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations. *arXiv preprint arXiv:1902.03249*.

Zhiqing Sun, Zhuohan Li, Haoqing Wang, Di He, Zi Lin, and Zhihong Deng. 2019. Fast structured decoding for sequence models. In *NeurIPS*.

Jianheng Tang, Tiancheng Zhao, Chenyan Xiong, Xiaodan Liang, Eric P. Xing, and Zhiting Hu. 2019. Target-guided open-domain conversation. In *ACL*.

Sean Welleck, Kianté Brantley, Hal Daumé III, and Kyunghyun Cho. 2019. Non-monotonic sequential text generation. *arXiv preprint arXiv:1902.02192*.

Felix Wu, Angela Fan, Alexei Baevski, Yann N Dauphin, and Michael Auli. 2019. Pay less attention with lightweight and dynamic convolutions. *arXiv preprint arXiv:1901.10430*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*.

Lili Yao, Nanyun Peng, Ralph Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2019. Plan-and-write: Towards better automatic storytelling. In *AAAI*, volume 33, pages 7378–7385.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. In *NeurIPS*.

Yizhe Zhang, Michel Galley, Jianfeng Gao, Zhe Gan, Xiujun Li, Chris Brockett, and Bill Dolan. 2018. Generating informative and diverse conversational responses via adversarial information maximization. In *NeurIPS*.

Yizhe Zhang, Dinghan Shen, Guoyin Wang, Zhe Gan, Ricardo Henao, and Lawrence Carin. 2017. Deconvolutional paragraph representation learning. In *NeurIPS*.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2020. Dialogpt: Large-scale generative pre-training for conversational response generation. In *ACL (system demonstration)*.

# Appendix

## A Baseline and Experimental Details

For NMSTG, we first convert the lexical constraints into a prefix sub-tree, and then sample a sentence to complete the sub-tree. We use the default settings suggested by the authors, and use an LSTM with hidden size of 1024 as the text generator, and select the best performed variants (*annealed*) as our baseline. For CGMH, we use their default setting, which uses an LSTM with hidden size of 300, and set the vocabulary size as 50k. Both models are trained until the evaluation loss does not decrease. During inference, we run CGMH for 500 iterations with default hyperparameters.

For experiment setup, we employ the tokenizer from BERT, and use WordPiece Embeddings (Wu et al., 2016) with a 30k token vocabulary for all the tasks. A special no-insertion token [NOI] is added to the vocabulary. We utilize the BERT-base and BERT-large models with 12 self-attention layers and 768 hidden dimensions as our model initialization. Each model is trained until there is no progress on the validation loss. We use a learning rate of 3e-5 without any warming-up schedule for all the training procedures. The optimization algorithm is Adam (Kingma and Ba, 2015). We pretrain our model on the Wiki dataset for 2 epochs, and fine-tune on the News and Yelp datasets for around 10 epochs.

## B Additional Generated Examples for News Dataset

We provide two examples on News dataset for how the model progressively generates the sentences in Table 7. All the generations are from the POINTER large model using greedy decoding.

In this section, we also provide some additional examples from the 1k news test data.

| Stage | Generated text sequence |
|---|---|
| 0 ($X^0$) | aware negative immediately sites |
| 1 ($X^1$) | **if** aware **posts** negative **should** immediately **any** sites **posts** |
| 2 ($X^2$) | **would** if **user** aware **that** posts **have** negative **impact** should immediately **related** any **these** sites **remove** posts |
| 3 ($X^3$) | **this** would **prefer** if **the** user **is** aware that **the** posts have **a** negative impact **and** should **be** immediately related **to** any **of** these sites **and** remove **those** posts . |

| Stage | Generated text sequence |
|---|---|
| 0 ($X^0$) | estate pay stay policy |
| 1 ($X^1$) | **also** estate **agents** pay **amount** stay **same** policy assets |
| 2 ($X^2$) | **it** also **required** estate agents **who** pay **same** amount **cash** stay **with** same policy **all** assets |
| 3 ($X^3$) | **but** it **would** also **be** required **for** estate agents **,** who **must** pay **the** same amount **of** cash **but** stay with **the** same policy **for** all **other** assets . |

Table 7: Example of the progressive generation process with multiple stages from the POINTER model. New additions at each stage are marked as **blue**.

| Keywords | aware negative immediately sites |
|---|---|
| ORACLE | where we become **aware** of any accounts that may be **negative** , we **immediately** contact companies such as Instagram , although we have no control over what they allow on their **sites** . |
| CGMH | not even **aware** of **negative** events including video events **immediately** at stations , Facebook **sites**. |
| NMSTG | health providers in a country for England are **aware** of small health systems - and not non - health care but all **negative** is **immediately** treated by heads of businesses and departments in the **sites** . |
| POINTER (Greedy, base) | ‘ if users are **aware** of the **negative** impact of blocking , how can they so **immediately** ban these **sites** ? ’ the researchers wrote . |
| POINTER (ILBS, base) | if the users are **aware** of or the **negative** messages , they can **immediately** be transferred to other **sites** . |
| POINTER (Greedy, Large) | this would prefer if the user is **aware** that the posts have a **negative** impact and should be **immediately** related to any of these **sites** and remove those posts . |
| Wiki zero-shot | he is not **aware** of the **negative** , and will **immediately** go to the positive **sites** . |

| Keywords | children fault left charge |
|---|---|
| ORACLE | my relationship with my **children** was seriously affected as they were told time and again that everything was my **fault** , they were even **left** ' in **charge** ' of me if my wife went out of the house . |
| CGMH | his two **children** are the rare **fault** that **left** the police **charge** |
| NMSTG | but despite **children** from hospitals to last one by **fault** backing this month , there have arrived as Mr Hunt has been **left charge** . |
| POINTER (Greedy, base) | but i found that these **children** were not at school however this was not their **fault** , and if so they were **left** without a parent in **charge** . |
| POINTER (ILBS, base) | but my lovely wife and **children** consider that it is not our own **fault** and we should not be **left** alone in **charge** . |
| POINTER (Greedy, Large) | i said to my **children** : it ' s not his **fault** the parents **left** him ; the parents should be in **charge** of him . |
| Wiki zero-shot | but for the **children** who are not at a **fault** , they are **left** behind on the **charge** . |

| Keywords | managers cut costs million |
|---|---|
| ORACLE | he was the third of four **managers** sent in to **cut costs** and deal with the city ' s $ 13 **million** deficit . |
| CGMH | the **managers** , who tried to **cut** off their **costs** , added 20 **million** euros |
| NMSTG | business **managers cut** demand for more expensive **costs** in 2017 - by October - is around 5 **million** 8 per cent , and has fallen by 0 . 3 per cent in January and 2017 . |
| POINTER (Greedy, base) | under one of its general **managers** , the firm had already **cut** its annual operating **costs** from $ 13 . 5 **million** to six million euros . |
| POINTER (ILBS, base) | and last month , the **managers** announced that it had **cut** its operating **costs** by $ 30 **million** . |
| POINTER (Greedy, Large) | the biggest expense is for the **managers** , where it plans to **cut** their annual management **costs** from $ 18 . 5 **million** to $ 12 million . |
| Wiki zero-shot | but then he and all of his **managers** agreed to **cut** off all of the operating **costs** by about 1 **million** . |

| Keywords | estate pay stay policy |
|---|---|
| ORACLE | how many people on the **estate** does he think will be affected by the new **pay** - to - **stay policy** ? |
| CGMH | an economic **estate** developer that could **pay** for it is that a **stay policy** |
| NMSTG | as **estate** owners , they cannot **pay** for households for hundreds of middle - income property , buyers **stay** in retail **policy** . |
| POINTER (Greedy, base) | if you buy new buildings from real **estate** company, you may have to **pay** down a mortgage and **stay** with the **policy** for financial reasons . |
| POINTER (ILBS, base) | but no matter what foreign buyers do , real **estate** agents will have to **pay** a small fee to **stay** consistent with the **policy** . |
| POINTER (Greedy, Large) | but it would also be required for **estate** agents , who must **pay** a larger amount of cash but **stay** with the same **policy** for all other assets . |
| Wiki zero-shot | however , his real **estate** agent agreed to **pay** him for the **stay** under the same **policy** . |

| Keywords | looked report realized wife |
|---|---|
| ORACLE | i **looked** at the **report** and saw her name , and that's when I **realized** it was my ex-**wife** . |
| CGMH | he **looked** at the **report** and said he **realized** that if his **wife** Jane |
| NMSTG | i **looked** at my **report** about before I **realized** I return to travel holidays but - it doesn ' t haven ' t made anything like my **wife** . |
| POINTER (Greedy, base) | when i turned and **looked** at a file **report** from the airport and **realized** it was not my **wife** and daughter . |
| POINTER (ILBS, base) | when i turned around and **looked** down at the pictures from the **report** , i **realized** that it was my **wife** . |
| POINTER (Greedy, Large) | however , when they **looked** at the details of the **report** about this murder , they quickly **realized** that the suspect was not with his **wife** or his partner . |
| Wiki zero-shot | but when he **looked** up at the **report** , he **realized** that it was not his **wife** . |

| Keywords | time claim tax year |
|---|---|
| ORACLE | walker says there is still **time** to **claim** this higher protection if you haven ' t already as the deadline is the end of the 2016 / 2017 **tax year** . |
| CGMH | " two states , one - **time** voters can **claim** a federal **tax year** |
| NMSTG | this **time** they had three to **claim** of an equal **tax** and 34 women at which indicated they should leave that over the **year** of 16 . |
| POINTER (Greedy, base) | it is the very first **time** in history that trump will ever **claim** over $ 400 million in federal income **tax** that he had held last **year** , the same report says . |
| POINTER (ILBS, base) | is this the very first **time** someone has to **claim** federal income **tax** twice in a single **year** ? |
| POINTER (Greedy, Large) | this is not for the first **time** that the scottish government was able to **claim tax** cuts of thousands of pounds a **year** to pay . |
| Wiki zero-shot | but at the **time** , the **claim** was that the same sales **tax** that was from the previous fiscal **year** . |

| Keywords | made year resolution managed |
|---|---|
| ORACLE | i once **made** this my new **year** ' s **resolution** , and it is the only one that I ' ve actually ever **managed** to keep . |
| CGMH | indeed , as he **made** up the previous **year** , the GOP **resolution** was **managed** |
| NMSTG | while additional sanctions had been issued last week **made** a **year** from the latest **resolution** , Russia ' s Russian ministers have but have **managed** . |
| POINTER (Greedy, base) | no progress has been **made** in syria since the security council started a **year** ago , when a **resolution** expressed confidence that moscow **managed** to save aleppo . |
| POINTER (ILBS, base) | and the enormous progress we have **made** over the last **year** is to bring about a **resolution** that has not been **managed** . |
| POINTER (Greedy, Large) | the obama administration , which **made** a similar call earlier this **year** and has also voted against a **resolution** to crack down on the funding , **managed** to recover it . |
| Wiki zero-shot | but despite all the same changes **made** both in both the previous fiscal **year** , and by the un **resolution** itself , only the federal government **managed** ... |

| Keywords | model years big drama |
|---|---|
| ORACLE | the former **model** said : " I haven ' t seen him in so many **years** , I can ' t make a **big drama** out of it . " |
| CGMH | the " **model** " continues , like many **years** of sexual and **big drama** going |
| NMSTG | after **model** two **years** and did it like , could we already get bigger than others in a **big drama** ? |
| POINTER (Greedy, base) | but i am a good role **model** , who has been around for 10 **years** now , and that is a **big** example of what i can do in **drama** on screen . |
| POINTER (ILBS, base) | but the young actress and **model** , for 15 **years** , made a very **big** impact on the **drama** . |
| POINTER (Greedy, Large) | i have seen the different **model** she recommends of over **years** , but it ' s no **big** change in the **drama** after all . |
| Wiki zero-shot | she was a **model** actress for many **years** and was a **big** star in the **drama** . |

| Keywords | club believed centre window |
|---|---|
| ORACLE | the **club** are **believed** to be keen on bringing in cover at **centre** - back during the current transfer **window** , with a loan move most likely . |
| CGMH | the **club** has also been **believed** that more than a new **centre** - up **window** |
| NMSTG | one **club believed** it was not clear that the **centre** would hold place on the **window** until there were no cases that they had heard or had the decision disappeared . |
| POINTER (Greedy, base) | he had been talking to the **club** since he is **believed** to have reached the **centre** spot in the queue before the january transfer **window** was suspended . |
| POINTER (ILBS, base) | when he left his old **club** , chelsea , he was **believed** to be at the **centre** of the transfer **window** . |
| POINTER (Greedy, Large) | the striker has remained at the **club** at the weekend and is increasingly **believed** to be available as a **centre** of the club during the summer transfer **window** until january 2016 . |
| Wiki zero-shot | during his first **club** as manager he was widely **believed** to be at the **centre** forward in the january transfer **window** . |

| | |
|---|---|
| Keywords | great past decade city |
| ORACLE | it ' s been a **great** time , the **past decade** or so , to be the mayor of a major capital **city** . |
| CGMH | the great past decade is that so much of a new home city |
| NMSTG | i like to thank you for me and I ' ve wanted it to grow in every **great past decade** over the **city** , a very amazing time . |
| POINTER (Greedy, base) | this is one of the **great** cities that he have visited in the **past** two **decade** , the kansas **city** , missouri , he says . |
| POINTER (ILBS, base) | you don ' t feel as **great** as you ' ve been in the **past decade** in a major **city** . |
| POINTER (Greedy, Large) | there has been a lot of **great** work here in the **past** few years within more than a **decade** , done for the **city** , he says . |
| Wiki zero-shot | there was a **great** success in the **past** during the last **decade** for the **city** . |

## C Additional Generated Examples for Yelp Dataset

We provide two examples on Yelp dataset for how the model progressively generates the sentences in Table 8. All the generations are from the POINTER large model using greedy decoding.

We also provide some additional examples from the Yelp test set. The results includes keywords, human oracle, CGMH, NMSTG and our models. For our models, we include POINTER base and large models with greedy decoding and base model with ILBS. The large model with ILBS is time consuming so we omit them from the comparison.

| Stage | Generated text sequence |
|---|---|
| 0 ($X^0$) | delicious love mole rice back |
| 1 ($X^1$) | **restaurant** delicious **authentic** love **dish** mole **beans** rice **definitely** back **!** |
| 2 ($X^2$) | **new** restaurant **so** delicious **fresh** authentic **.** love **mexican** dish **called** mole **with** beans **and** rice **we** definitely **coming** back **more** ! |
| 3 ($X^3$) | **this** new restaurant **is** so delicious **,** fresh **and** authentic **tasting** . **i** love **the** mexican **style** dish **,** called **the** mole **,** with **black** beans **,** and **white** rice **.** we **will** definitely **be** coming back **for** more ! |

| Stage | Generated text sequence |
|---|---|
| 0 ($X^0$) | joint great food great drinks greater staff |
| 1 ($X^1$) | **new** joint **around** great **location** food **variety** great **craft** drinks **unless** greater **friendly** staff **!** |
| 2 ($X^2$) | **is** new **breakfast** joint **be** around **area** great **,** location **excellent** food **nice** variety **selections** great **of** craft **,** drinks **quite** unless **ask** greater **.** friendly **and** staff **love** ! |
| 3 ($X^3$) | **this** is **the** new **modern** breakfast joint **to** be **found** around **the** area **.** great **atmosphere** , **central** location **and** excellent food **.** nice variety **of** selections **.** great **selection** of **local** craft **beers** , **good** drinks . quite **cheap** unless **you** ask **for** greater **price** . **very** friendly **patio** and **fun** staff **.** love **it** ! |

Table 8: Example of the progressive generation process with multiple stages from the POINTER model. New additions at each stage are marked as **blue**.

| | |
|---|---|
| Keywords | service perfect delicious service awesome good place |
| ORACLE | yummy excellent **service** . ordered the carne asada medium rare . it was **perfect** . and **delicious** . their customer **service** was **awesome** . they were so friendly and made sure all was **good** . i definitely recommend this **place** . |
| CGMH | great **service** **perfect** food and **delicious** **service** . **awesome** place and **good** **place** !. |
| NMSTG | **service** was **perfect** , **delicious** and great **service** **awesome** service **good** food . this **place** will go back . |
| POINTER (Greedy, base) | excellent food , great **service** , really nice atmosphere , **perfect** amount of spring rolls , **delicious** especially the chicken and eel . the **service** was very friendly and the prices are **awesome** too . for a female who loves **good** japanese restaurant , this is definitely your **place** ! |
| POINTER (ILBS, base) | from the food to **service** . the foods are **perfect** , they were **delicious** . and **service** is beyond expectation . christina was **awesome** , so many **good** things about this **place** . |
| POINTER (Greedy, Large) | absolutely loved the food and very friendly **service** . i had the chicken , it was cooked **perfect** and the seafood pasta was thick and **delicious** and not too heavy though . our **service** guy at the front bar was so **awesome** , he made sure we had a **good** time . would definitely recommend to try this **place** to anyone ! |
| Wiki zero-shot | he said the **service** was **perfect** , and **delicious** , and the **service** that is **awesome** , and very **good** in its **place** . |

| | |
|---|---|
| Keywords | good drinks love clients tighter great service |
| ORACLE | **great** atmosphere , good food and **drinks** . i **love** coming here in the fall to spring to meet with **clients** . their inside is a little small and makes summer a bit **tighter** , but still a **great** staff with excellent **service** . |
| CGMH | **good** **drinks** . i **love** how out **clients** are **tighter** . **great** customer **service** . |
| NMSTG | such **good** place with i love the mushroom **drinks** . the menu they **love** the **clients** . and **tighter** out the menu are **great** **service** . |
| POINTER (Greedy, base) | this place is **good** . they have a wide variety of **drinks** . this really fits your taste . **love** the cozy bar that allows **clients** to be able to fit very tightly and **tighter** , better blending with the crowd . **great** coffee , reasonable prices , and friendly **service** ! |
| POINTER (ILBS, base) | nice place , with **good** vibe . nice mix of **drinks** and intimate space . what i really **love** about was there were so more mature **clients** , and they can fit in a **tighter** timeline . overall , **great** atmosphere and excellent **service** . |
| POINTER (Greedy, Large) | really like this place . has a **good** dj , good atmosphere and cool **drinks** and quite nice lounge area . i **love** this idea of having fun on your **clients** and rubbing your feet to stand up **tighter** than other ones . **great** variety of drinks and pretty quick **service** at the bar ! |
| Wiki zero-shot | she is a **good** at **drinks** , and in **love** for him and all his **clients** , and he enjoys a **tighter** schedule and has a **great** food and a generous **service** . |

| | |
|---|---|
| Keywords | joint great food great drinks greater staff |
| ORACLE | apteka is seriously all around the best vegan **joint** in the burgh . **great food** , **great drinks** , **greater staff** . |
| CGMH | very cool **joint** with **great food** , **great drinks** and even **greater staff** . ! |
| NMSTG | awesome **joint** . **great** service . **great food** great **drinks** . good to **greater** and great **staff** ! |
| POINTER (Greedy, base) | my favorite local **joint** around old town . **great** atmosphere , amazing **food** , delicious and delicious coffee , **great** wine selection and delicious cold **drinks** , oh and maybe even a **greater** patio space and energetic front desk **staff** . |
| POINTER (ILBS, base) | the best breakfast **joint** in charlotte . **great** service and amazing **food** . they have **great** selection of **drinks** that suits the **greater** aesthetic of the **staff** . |
| POINTER (Greedy, Large) | this is the new modern breakfast **joint** to be found around the area . **great** atmosphere , central location and excellent **food** . nice variety of selections . **great** selection of local craft beers , good **drinks** . quite cheap unless you ask for **greater** price . very friendly patio and fun **staff** . love it ! |
| Wiki zero-shot | it is a joint owner of the **great** society of irish **food** , and the **great** britain and soft **drinks** , and the **greater** britain and its **staff** . |

| | |
|---|---|
| Keywords | service polite professional affordable work safe tree |
| ORACLE | aron's tree **service** were very **polite** and **professional** . they are very **affordable** . they arrived a little early and got right to **work** . they were quick and **safe** . they cleaned up and hauled out the **tree** trimmings . i highly recommend them . |
| CGMH | excellent customer **service** , **polite** , **professional** , and **affordable work** , **safe** bike **tree** . |
| NMSTG | excellent food and **service** and are amazing service and **polite** and **professional** . **affordable** it **work** out **safe** on sun **tree** ! |
| POINTER (Greedy, base) | amazing customer **service** . so **polite** , and very **professional** , and very **affordable** . such great **work** done at the **safe** end of a **tree** . |
| POINTER (ILBS, base) | excellent customer **service** , very **polite** , and very **professional** . honest and **affordable** pricing . i will definitely get the **work** done here for the **safe** parts of my **tree** . |
| POINTER (Greedy, Large) | diane provides customers with great customer **service** . technician mike was very **polite** and helpful . clean facility , very **professional** , and always responsive . quick and **affordable** as well . i had very nice **work** done . we have now found someone **safe** . thank you big two buck **tree** shrub care ! |
| Wiki zero-shot | customer **service** should be more **polite** , and more **professional** , and more **affordable** , and will **work** in a **safe** place under the family **tree** . |

| | |
|---|---|
| Keywords | hesitate give customers chicken rice decent list |
| ORACLE | i **hesitate** to **give** them the five stars they deserve because they have a really small dining area and more **customers** , selfishly , would complicate things for me . **chicken** panang is quite good with a superb brown **rice** . **decent** wine **list** . after three visits the wait staff remembered what i like ( complicated ) and always get the order right . |
| CGMH | they **hesitate** to **give customers** their **chicken** fried **rice** and a **decent** wine **list** . |
| NMSTG | they **hesitate** to an wonderful time to **give** it about a table , love the **customers chicken rice** and dishes seafood and **decent** at the **list** . |
| POINTER (Greedy, base) | i just did not even **hesitate** to admit , i should **give** credit cards to my **customers** here . the beijing **chicken** and fried **rice** were spot on , a **decent** side on my favorite **list** . |
| POINTER (ILBS, base) | i don't have to **hesitate** that they should **give** five stars . i will be one of their repeat **customers** . like the basil **chicken** and basil fried **rice** , it was **decent** on my **list** . |
| POINTER (Greedy, Large) | service is very slow , don ' t **hesitate** to tell manager to **give** some feedbacks as their job is to take care of their **customers** . had the vegetable medley soup and **chicken** . both were cooked well . the garlic **rice** did not have the vegetable and was fairly **decent** . they are changing the flavor and **list** of menu items . |
| Wiki zero-shot | he did not **hesitate** himself to **give** it to his **customers** , such as **chicken** , and steamed **rice** , a very **decent** item on the **list** . |

| | |
|---|---|
| Keywords | good potential bad maintained replaced dirty disgusting |
| ORACLE | has **good potential** but very **bad maintained** . the padding is done , needs to be **replaced** , holes everywhere . so are those huge flowers or what ever those are . ripped . very **dirty** too . there was a a very dirty towel laying on the floor **disgusting** . please the city of vegas come and clean it ! |
| CGMH | **good potential** but **bad** service. not **maintained** . it **replaced** a **dirty** box . **disgusting** . |
| NMSTG | do a **good** price . not like the and **potential bad maintained** has disgusting . **replaced** been , **dirty** and **disgusting** . |
| POINTER (Greedy, base) | the food was very **good** . it really has more **potential** maybe , but it smells really **bad** . its not very well **maintained** either . trash cans were **replaced** only when they were **dirty** . the floors were utterly **disgusting** . |
| POINTER (ILBS, base) | the food is really **good** . this location has **potential** to be pretty **bad** and not very well **maintained** when it was **replaced** , its super **dirty** , just plain **disgusting** . |
| POINTER (Greedy, Large) | this gym is not so **good** . overall it has a lot of **potential** for being better but it is too **bad** that it is not clean and un **maintained** and towels are in desperate need to be **replaced** regularly . the floors are very **dirty** and the higher floors have become filthy **disgusting** when i visited here . |
| Wiki zero-shot | it is **good** it has no **potential** , and the **bad** taste can be **maintained** until they are **replaced** by a **dirty** , and **disgusting** one . |

| | |
|---|---|
| Keywords | love animal style long line expected quick |
| ORACLE | who doesn t **love** in and out . **animal** style is a must . **long line** but **expected** , it goes **quick** anyways so don t let that discourage you . |
| CGMH | **love** this place . **animal style** food . **long line** than **expected** for **quick** . |
| NMSTG | **love animal** chicken . it was **style long** a bit so good . the **line** is it was even on on a time and we **expected** to go but **quick** . |
| POINTER (Greedy, base) | great little breakfast spot . i **love** having the double with **animal style** fries and protein style etc . have a super **long** wait **line** , but its just as **expected** and it always moves pretty **quick** too . |
| POINTER (ILBS, base) | y all you just gotta **love** about this place is the double **animal style** and protein style . it was a **long line** , but i **expected** it to be **quick** . |
| POINTER (Greedy, Large) | great burger and good price . i **love** that they have non chain locations . i like the **animal style** fries too . have to wait **long** as there is always traffic but the **line** can be much shorter than i had **expected** and they are always send out pretty **quick** . very impressed ! |
| Wiki zero-shot | he also has **love** with the **animal** and his **style** , and was **long** as the finish **line** , and was **expected** to be **quick** . |

| | |
|---|---|
| Keywords | great great service happy found close home |
| ORACLE | **great** sushi and **great service** . i m really **happy** to have **found** a good sushi place so **close** to **home** ! |
| CGMH | **great** price and **great** customer **service** . very **happy** that i **found** this place **close** to my **home** . |
| NMSTG | **great** food and **great service** . a **happy** and **found** a year in **close** for them . keep them **home** here . |
| POINTER (Greedy, base) | amazing food . **great** quality food . **great** prices and friendly **service** staff . so **happy** and surprised to have finally **found** such a wonderful nail salon so **close** to my work and **home** . |
| POINTER (ILBS, base) | this is just **great** food . **great** food and wonderful **service** . very **happy** to have finally **found** a chinese restaurant **close** to my **home** . |
| POINTER (Greedy, Large) | wow . i have been here twice . **great** times here . food always has been **great** and the customer **service** was wonderful . i am very **happy** that we finally **found** our regular pad thai restaurant that is **close** to where we work now and our **home** . pleasantly surprised ! |
| Wiki zero-shot | he was a **great** teacher and a **great** love of the **service** he was very **happy** , and he **found** himself in the **close** to his **home** . |

## D Additional Human Evaluation information and Results

There were 145 judges in all: 5 judges evaluated each pair of outputs to be reasonably robust against spamming. P-values are all p¡0.00001 (line 721), computed using 10000 bootstrap replications. Judges were lightly screened by our organization for multiple screening tasks.

We present the additional human evaluation results on POINTER large model vs base model in table 11. In general, for the news dataset the results are mixed. For the yelp dataset, the large model

| Method | NIST N-2 | NIST N-4 | BLEU B-2 | BLEU B-4 | METEOR | Entropy E-4 | Dist D-1 | Dist D-2 | PPL | Avg Len |
|---|---|---|---|---|---|---|---|---|---|---|
| Greedy (+Wiki) | 3.04 | 3.06 | 13.01% | 2.51% | **16.38%** | 10.22 | 11.10% | 57.78% | **56.7** | 31.32 |
| ILBS (+Wiki) | 3.20 | 3.22 | 14.00% | 2.99% | 15.71% | 9.86 | 13.17% | 61.22% | 66.4 | 22.59 |
| Greedy (+Wiki,L) | **3.28** | **3.30** | **14.04%** | **3.04%** | 15.90% | 10.09 | 12.23% | 60.86% | **54.7** | 27.99 |
| Wiki zero-shot | 2.80 | 2.82 | 11.38% | 1.84% | 15.12% | 9.73 | 14.33% | 53.97% | 62.9 | 20.68 |
| Human | - | - | - | - | - | 10.05 | 11.80% | 62.44% | 47.4 | 27.85 |

Table 9: Additional evaluation results on the News dataset. ILBS denotes beam search. "+Wiki" denotes fine-tuning on the Wiki-pretrained model. "Human" represents the held-out human reference. "Wiki zero-shot" represents zero-shot generation from the pre-trained model.

| Method | NIST N-2 | NIST N-4 | BLEU B-2 | BLEU B-4 | METEOR | Entropy E-4 | Dist D-1 | Dist D-2 | PPL | Avg Len |
|---|---|---|---|---|---|---|---|---|---|---|
| Greedy (+Wiki) | 3.27 | 3.30 | 15.63% | 3.32% | 16.14% | 10.64 | 7.51% | 46.12% | 71.9 | 48.22 |
| ILBS (+Wiki) | 3.34 | 3.38 | 16.68% | 3.65% | 15.57% | 10.44 | 9.43% | 50.66% | 61.0 | 35.18 |
| Large (+Wiki) | **3.49** | **3.53** | **16.78%** | **3.79%** | 16.69% | 10.56 | 6.94% | 41.2% | **55.5** | 48.05 |
| Wiki zero-shot | 0.86 | 0.87 | 8.56% | 1.30% | 12.85% | 9.90 | 10.09% | 41.97% | 62.9 | 26.80 |
| Human | - | - | - | - | - | 10.70 | 10.67% | 52.57% | 55.4 | 50.36 |

Table 10: Additional evaluation results on the Yelp dataset. ILBS denotes beam search. "+Wiki" denotes fine-tuning on the Wiki-pretrained model. "Human" represents the held-out human reference. "Wiki zero-shot" represents zero-shot generation from the pre-trained model.

| **Informativeness**: *A and B, which is more semantically meaningful and consistent?* | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| News dataset | | | | | Yelp dataset | | | | |
| System A | | Neutral | System B | | System A | | Neutral | System B | |
| POINTER(large) | 35.4% | 27.7% | **36.9 %** | POINTER(base) | POINTER(large) | **41.4%** | 26.6% | 32.1 % | POINTER(base) *** |
| POINTER(large) | 20.3% | 22.7% | **57.1**% | Human *** | POINTER(large) | 27.2% | 24.4% | **48.5**% | Human *** |
| **Fluency**: *A and B, which is more grammatical and fluent?* | | | | | | | | | |
| News dataset | | | | | Yelp dataset | | | | |
| System A | | Neutral | System B | | System A | | Neutral | System B | |
| POINTER(large) | **38.4%** | 28.5% | 33.2 % | POINTER(base) | POINTER(large) | **41.1%** | 28.1% | 30.8 % | POINTER(base) *** |
| POINTER(large) | 16.7% | 15.8% | **67.5**% | Human *** | POINTER(large) | 27.1% | 21.9% | **51.1**% | Human *** |
| **Informativeness**: *A and B, which is more informative?* | | | | | | | | | |
| News dataset | | | | | Yelp dataset | | | | |
| System A | | Neutral | System B | | System A | | Neutral | System B | |
| POINTER(large) | 32.1% | 27.6% | **40.4 %** | POINTER(base) | POINTER(large) | **41.6%** | 25.0 % | 33.4 % | POINTER(base) *** |
| POINTER(large) | 31.9% | 17.1% | **51.0**% | Human *** | POINTER(large) | 35.9% | 14.7% | **49.4**% | Human *** |

Table 11: **Human Evaluation** on two datasets for semantic consistency, fluency and informativeness, showing preferences (%) for our POINTER(large) model vis-a-vis POINTER(base) model and real human responses. Numbers in bold indicate the most preferred systems. Significant differences ($p \leq 0.001$) are indicated as ***.

wins with a large margin. All results are still far away from the human oracle in all three aspects.

## E Additional Automatic Evaluation Results

We provide the full evaluation result data including Wikipedia zero-shot learning results in Table 9 and Table 10. Note that zero-shot generations from Wikipedia pre-trained model yield the lowest perplexity, presumably because the Wikipedia dataset is large enough so that the model trained on it can learn language variability, thus delivering fluent generated results.

## F Inference Details

During inference time, we use a decaying schedule to discourage the model from generating non-interesting tokens, including [NOI] and some other special tokens, punctuation and stop words. To do this, we use a decay multiplier $\eta$ on the logits of

these tokens before computing the softmax. The $\eta$ is set to be $\eta = \min(0.5 + \lambda * s)$, where $s$ is the current stage and $\lambda$ is an annealing hyper-parameter. In most of the experiments, $\lambda$ is set at $0.5$

## G Human Evaluation Template

See Figure 2 for human evaluation template

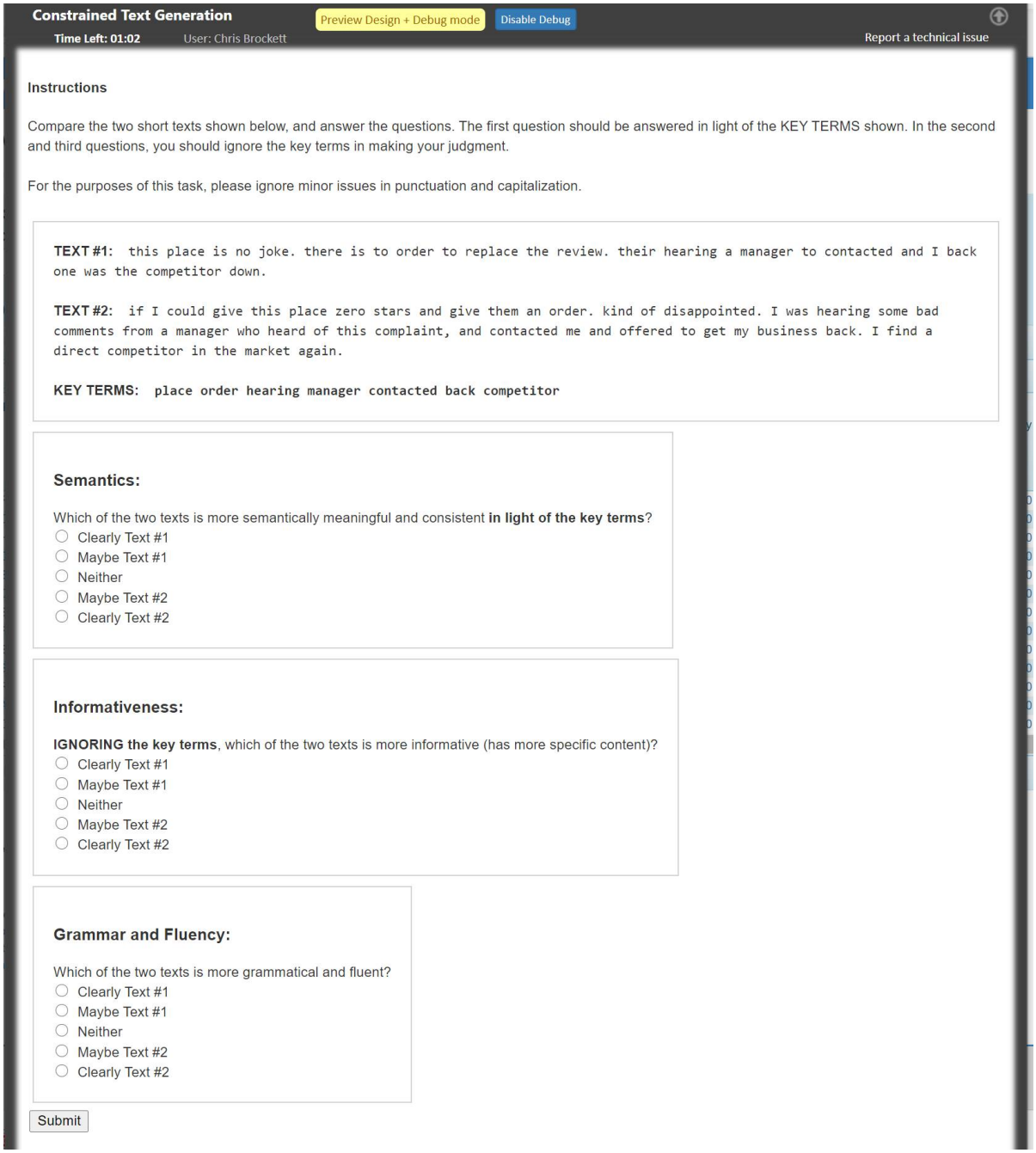


Figure 2: Human evaluation template.