BANG: Bridging Autoregressive and Non-autoregressive Generation with Large Scale Pretraining

Weizhen Qi[1] *, Yeyun Gong[3]†, Jian Jiao[2]†, Yu Yan[2]†, Dayiheng Liu[4]*†, Weizhu Chen[5]†,
Kewen Tang[2], Houqiang Li[1], Jiusheng Chen[2], Ruofei Zhang[2], Ming Zhou[3], Nan Duan[3]

[1]University of Science and Technology of China, [2]Microsoft

[3]Microsoft Research Asia, [5]Microsoft Dynamics 365 AI, [4]Sichuan University

[1]`weizhen@mail.ustc.edu.cn, lihq@ustc.edu.cn,` [4]`losinuris@gmail.com`
[3]`{yegong, nanduan, mingzhou}@microsoft.com,` [5]`wzchen@microsoft.com`
[2]`{jian.jiao, yyua, Kewen.Tang, jiuchen, bzhang}@microsoft.com`

Abstract

In this paper, we propose **BANG**, a new pretraining model to **B**ridge the gap between **A**utoregressive (AR) and **N**on-autoregressive (NAR) **G**eneration. AR and NAR generation can be uniformly regarded as what extend of previous tokens can be attended to, and BANG bridges AR and NAR generation through designing a novel model structure for large-scale pre-training. A pretrained BANG model can simultaneously support AR, NAR, and semi-NAR generation to meet different requirements. Experiments on question generation (SQuAD 1.1), summarization (XSum), and dialogue (PersonaChat) show that BANG improves NAR and semi-NAR performance significantly as well as attaining comparable performance with strong AR pretrained models. Compared with the semi-NAR strong baselines, BANG achieves absolute improvements of 14.01 and 5.24 in overall scores of SQuAD and XSum, respectively. In addition, BANG achieves absolute improvements of 10.73, 6.39, and 5.90 in overall scores of SQuAD, XSUM, and PersonaChat compared with the NAR strong baselines, respectively. Our code will be made publicly available in the near future[1].

1 Introduction

Various pretraining methods have been successfully applied in autoregressive (AR) natural language generation models (Song et al., 2019; Lewis et al., 2019; Qi et al., 2020; Raffel et al., 2020; Zhang et al., 2019a), and bring significant improvements on tasks such as summarization, question generation, and machine translation.

Although the autoregressive generation method achieves high-quality results in many tasks, its latency is a well-known limitation for online real-time usage. To alleviate this issue, non-autoregressive (NAR) models (Gu et al., 2017; Lee et al., 2018; Ghazvininejad et al., 2019; Raffel et al., 2020; Zhang et al., 2019a) have attracted wide attention. Different from AR models which generate the tokens sequentially, NAR models generate the tokens in parallel. Compared to AR models, NAR models generally comes with a much better inference efficiency, but a worse accuracy. In order to balance latency and accuracy, semi-NAR generation models (Stern et al., 2019; Lee et al., 2018; Gu et al., 2019; Ghazvininejad et al., 2019) are proposed as a hybrid approach.

For AR models, the training strategy of teacher-forcing is commonly used, which uses the golden (**G**) tokens as previous context to predict the next token. For NAR models, [MASK] (a special token) initialization (Ghazvininejad et al., 2019) or other methods like encoder copy (Gu et al., 2017) and posterior distribution approximation (Shu et al., 2020) are used. Though pretraining methods have been widely developed for AR generation, as our best knowledge, no works have been carried out on NAR large-scale pretraining. Besides, it's expensive to pretrain and maintain different pretraining models for different generation requirements.

To benefit NAR from the high accuracy of the AR model and support different generation patterns with the same pretrained model, we propose a new pretraining natural language generation model called BANG, which bridges the gap between AR and NAR sequence-to-sequence generation. We consider AR and NAR generation uniformly as what extend of generated/golden previous tokens can be attended to. BANG is pretrained to predict each token with arbitrary length of previous golden tokens replaced with [MASK]s. With complete previous golden tokens, BANG predicts the next token in an AR manner. With all previous

---

tokens replaced by [MASK], BANG predicts the next token in the NAR manner. With part of previous tokens replaced by [MASK], it benefits NAR generation with curriculum learning (Bengio et al., 2009; Guo et al., 2020) and benefits AR generation with the prediction of future tokens (Qi et al., 2020; Goodman et al., 2020).

In BANG pretraining, we consider the previous context of **G**+[MASK], with arbitrary **G** length and [MASK] length. We propose a new structure named cross-stream visible n-stream self attention to pretrain with different **G**+[MASK] combinations' predicting in parallel. For usage, the pretrained BANG model can be directly loaded to finetune as normal AR or NAR models. BANG can also be finetuned with the pretraining same strategy to support predicting tokens with arbitrary previous golden tokens or [MASK] (semi-NAR inference). With this semi-NAR finetuning, BANG is able to predict the first several tokens one by one as a high-quality sub-sequence hint, then predict the rest tokens simultaneously.

The main contributions of our method are: 1) BANG bridges the gap between AR and NAR by considering arbitrary previous [MASK] length and large-scale pretraining. 2) BANG is pretrained in a high-efficiency cross-stream visible n-stream decoder to realize parallelization. Each token with the arbitrary number of previous tokens replaced with [MASK] is trained to predict simultaneously at each time step. 3) BANG supports NAR, semi-NAR, and AR finetuning to meet different requirements with the same pretrained model structure. 4) We pretrain BANG$_{base}$ with 16GB English language corpora of Wikipedia and BookCorpus, and finetune on 3 popular natural language generation tasks in AR, semi-NAR, and NAR manners, respectively. For NAR and semi-NAR finetuning, BANG achieves significant performance improvement on all the tasks. For AR finetuning, BANG achieves comparable results with strong AR pretrained baselines.

2   Model Structure

The motivations of BANG are that 1) BANG model structure should be flexible to support AR, NAR and semi-NAR generation. 2) BANG pretraining task should design a curriculum learning path from AR to NAR to bridge the gap. 3) BANG should complete the extra bridging computation in the pretraining stage in high efficiency with no

extra cost for down-stream AR or NAR finetuning.

To meet these requirements, we propose: 1) Duplicating decoders into different structures by sharing parameters, to be flexible for different generation patterns. 2) BANG language model as a new pretraining task to bridge AR and NAR generation. 3) Cross-stream visible n-stream strategy as a BANG implementation to realize parallelization.

In this section, we firstly introduce fundamental concepts in § 2.1, then BANG model structure in § 2.2.

2.1   Fundamental Concepts

BANG is based on the Transformer Encoder-Decoder structure. The input sequence is encoded into a list of hidden states, and different generation patterns decode them with different decoder structures. In this section, we will describe the fundamental concepts regarding AR, NAR and semi-NAR generation.

2.1.1   AR generation

AR generation follows the conditional probability as:

$$P(Y|X) = \prod_{t=1}^{|Y|} P(y_t|Y_{<t}, X) \qquad (1)$$

From equation 1 we can see that each token y_t in target sequence Y is predicted with the dependency of input sequence X and previous tokens $Y_{<t}$. Vanilla Transformer realize this target by shifting decoder inputs one position, each token attending to its previous tokens to predict the next token. XLNet (Yang et al., 2019) proposes 2-stream strategy to realize training auto-encoding task in AR manner. 2-stream strategy is proposed because that each token is predicted with the initialization of [MASK] rather than previous token, which can not serve as previous contexts for AR generation. To be consistent with other generation pattern, BANG also uses [MASK] initialization for and follows 2-stream strategy to realize AR finetuning parallelization. We take the prediction of y_3 as an example in Figure 1 with the encoder part omitted:

In Figure 1 left part, we can see BANG predicts y_3 with y_0, y_1, y_2 visible as $P(y_3|y_0, y_1, y_2)$. To train predicting each token in parallel, BANG duplicates decoder layers into one main stream and one predicting stream as shown in Figure 1 right part. [MASK] in the predicting stream attends to its previous tokens from main stream to predict corresponding position's target token.
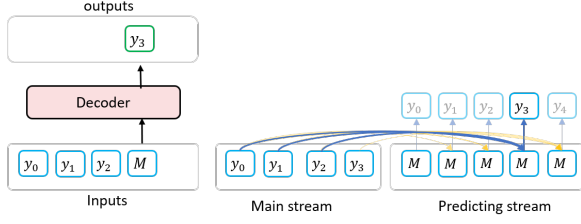
Figure 1: BANG AR generation. The [MASK] can attend to its previous **G** tokens to predict y_3.

2.1.2 NAR generation

NAR generation follows the conditional probability as:

$$P(Y|X) = \prod_{t=1}^{|Y|} P(y_t|X) \qquad (2)$$

From equation 2 we can see that each token y_t in target sequence Y is predicted with the dependency of input sequence X but no dependency of previous tokens. We follow CMLM (Ghazvininejad et al., 2019) or Fairseq version NAT (Ott et al., 2019; Gu et al., 2017) to feed a list of [MASK] tokens as the decoder initialization. We take the prediction of y_3 as an example in Figure 2 with the encoder part omitted.
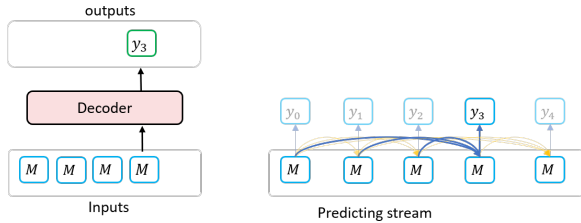

Figure 2: BANG NAR generation. The [MASK] can only attend to its previous [MASK]s to predict y_3.

In Figure 1 left part, we can see BANG predicts y_3 with no context information from target sequence Y but a list of [MASK]s. From Figure 1 right part, a main difference between BANG NAR and CMLM or NAT is that [MASK] in BANG decoder can only attend to its previous [MASK]s rather than every token in the decoder. We design BANG in this attention manner with two reasons: 1) To be consistent with BANG AR and benefit from AR-NAR bridging pretraining. 2) If [MASK] in the decoder has bi-directional attention, the number of fed [MASK] will influence the result, under which condition a length predictor is needed. If the model predicts the wrong target sequence

length, the decoder has to fill all the extra tokens or lacks of enough positions. In BANG, only previous [MASK]s are visible, and the first generated [EOS] token is considered the signal of sentence end token as traditional AR models.

2.1.3 Semi-NAR generation

Semi-NAR generation follows the conditional probability as:

$$P(Y|X) = \prod_{t=1}^{|Y|} P(y_t|Y_{c_t}, X) \qquad (3)$$

In equation 3, Y_{c_t} means visible context for y_t from target sequence Y. Y_{c_t} is designed with different algorithms for different semi-NAR models (Stern et al., 2019; Lee et al., 2018; Ghazvininejad et al., 2019; Gu et al., 2019). For BANG semi-NAR generation, Y_{c_t} is designed as first several tokens in AR manner and the rest tokens in NAR manner. We take the BANG semi-NAR inference procedure as an example in Figure 3 and leave training details to the next section.
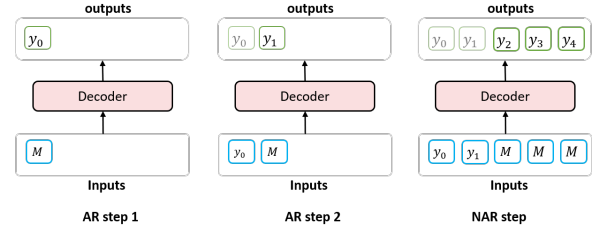
Figure 3: BANG semi-NAR generation. BANG semi-NAR inference supports predicting first several tokens step by step then generating rest tokens at one timestep. Here we present the inference procedure. The training procedure is same to Figure 4.

For semi-NAR finetuning, BANG has same model structure and workflow as pretraining which will be thoroughly introduced in the next section § 2.2 and Figure 4. We skip the training details and see the inference procedure in Figure 3. In this figure, we predict first two tokens one by one in AR manner. Then the AR generated sub-sequence is used as a hint for the rest tokens. A list of [MASK]s are appended to be predicted at one time step in NAR manner. With AR inference step zero or max-target-length, it converts into NAR or AR inference. The motivation of this semi-NAR generation pattern is that NAR models always mix up target sequence different expressions. High-quality AR generated sub-sequence can help NAR step about choosing the writing style and other dependencies.

2.2 BANG

In this section, we will introduce BANG model structure and its language model. For one aspect, some works (Qi et al., 2020; Goodman et al., 2020) point that future tokens' prediction will enhance AR generation. For another aspect, some works (Guo et al., 2019) use curriculum learning to gradually replace previous tokens with [MASK] to facilitate NAR generation. Since the intermediate states can help both AR and NAR generation, a main thought for BANG is to naturally bridge AR and NAR generation through large-scale pretraining.

We propose cross-stream visible n-stream structure for BANG decoder. As we introduce in §1, BANG considers arbitrary length of previous tokens replaced with [MASK], as the context of **G**+[MASK]. We duplicate decoder into one main stream and n predicting streams where n equals target sequence length by sharing parameters. Main stream is fed with **G** tokens to be used as golden previous tokens. Predicting streams are fed with [MASK]s to predict corresponding tokens and also serve as previous [MASK] tokens for NAR and semi-NAR generation. For the tokens in the ith predicting stream, their $i-1$ previous tokens are replaced by [MASK] from its previous predicting streams, further previous tokens are **G** tokens from the main stream. Each token in the 1st predicting stream is predicted in AR pattern with all previous **G** tokens visible from main stream. The first tokens in each predicting stream compose the NAR prediction with all previous visible tokens as previous predicting streams' [MASK]s.

Take Figure 4 as an example, we focus on the prediction of y_3 to illustrate how BANG bridges the AR-NAR generation. In the 1st predicting stream, [MASK] for y_3 attends to real y_0, y_1, y_2 from main stream **G** tokens. Each token in 1st predicting stream works same with y_3 and is pretrained on AR manner in parallel. In the 2nd predicting stream, [MASK] for y_3 addends to real y_0, y_1 from main stream **G** token and [MASK] for y_2 from 1st predicting stream. y_2 in 1st predicting stream and y_3 in 2nd predicting stream compose the conditional probability of $P(y_2, y_3|y_0, y_1)$ as shown in the figure left part. We can see the increased difficulty in 2nd predicting stream and the trend from AR to NAR because of more previous tokens replaced with [MASK]. Until the 4-th predicting stream, y_3 is predicted with no previous

G tokens visible. Only [MASK]s from its previous predicting streams are visible for y_3 prediction. We should notice that for each token in the target sequence, arbitrary previous **G** tokens replaced with [MASK] are considered in BANG, and every predicting stream is computed simultaneously. At each time step, each token from target sequence is predicted from AR pattern to NAR pattern simultaneously.

To sum up, the conditional probability of target sequence Y given input sequence X can be described as:

$$P(Y|X) = \prod_{i=1}^{|Y|}\prod_{t=i}^{|Y|} P(y_t|Y_{\leq t-i}, X) \quad (4)$$

For maximum likelihood training with a cross-entropy loss:

$$\begin{aligned} \mathcal{L}_{BANG} &= logP(Y|X) \\ &= \sum_{i=1}^{|Y|}\sum_{t=i}^{|Y|} logP(y_t|Y_{\leq t-i}, X) \end{aligned} \quad (5)$$

Here, the i denotes which predicting stream y_t is. We can understand this equation in different aspects. For equation 5, it represents that in each predicting stream, given the previous tokens $Y_{\leq t-i}$, the ith further token y_t is predicted. Or we can consider this equation by switching sum i and sum t, then it represents given any golden prefix subsequence, all the rest tokens are predicted simultaneously. In these two aspects, BANG represents the future span prediction ability. We can also reformat this equation to show how does BANG bridge AR and NAR generation in equation 6:

$$\begin{aligned} \mathcal{L}_{BANG} &= logP(Y|X) \\ &= \sum_{i=1}^{|Y|}\sum_{t=i}^{|Y|} logP(y_t|Y_{\leq t-i}, X) \\ &= \sum_{t=1}^{|Y|} logP(y_t|Y_{<t}, X) \\ &+ \sum_{i=2}^{|Y|-1}\sum_{t=i+1}^{|Y|} logP(y_t|Y_{\leq t-i}, X) \\ &+ \sum_{t=1}^{|Y|} logP(y_t|X) \end{aligned} \quad (6)$$

In equation 6, we can see the BANG optimizing target is composed by three

Figure 4: BANG pretraining. The prediction of y_3 is used for an example. "M-S" and "P-S" represent "Main stream" and "Predicting stream", respectively. The 1st predicting stream predicts tokens in AR manner. First tokens in each predicting stream compose NAR prediction.

parts: AR part $\sum_{t=1}^{|Y|} logP(y_t|Y_{<t}, X)$, NAR part $\sum_{t=1}^{|Y|} logP(y_t|X)$, bridging part $\sum_{i=2}^{|Y|-1} \sum_{t=i+1}^{|Y|} logP(y_t|Y_{\leq t-i}, X)$. AR part and NAR part directly optimizes down-stream generation pattern. The last part is able to pretrain the model more adequately according to former analysis, and it can also bridge AR and NAR generation and guarantee the ability for semi-NAR generation.

To make sure the GPU memory usage and computation cost acceptable, we adopt block-wise attention calculation. Since only the previous predicting streams are visible, each predicting stream is only concatenated with previous streams to avoid extra computation cost. In each decoder layer, predicting streams are calculated from the first one to the last one, calculated previous streams' k, v vectors are cached and re-used. For cross-stream visible n-stream self-attention calculation, workflow in decoder layer l can be described in Algorithm 1:

Here, function $Linear$ is three linear functions to calculate Q, K, V for self-attention from input hidden states h_l, operation $\oplus$ means appending or concatenating, and function $Attn$ can be described in equation 7:

$$Attn(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_{model}}} + L)V \quad (7)$$

Algorithm 1: Cross-stream visible n-stream self-attention

> **input** : h_l
> **output** : h_{l+1}
> $K_{cache} \leftarrow \emptyset$;
> $V_{cache} \leftarrow \emptyset$;
> $h_{l+1} \leftarrow \emptyset$;
> $Q, K, V = Linear(h_l)$;
> **for** Q_i, K_i, V_i *in split(Q, K, V)* **do**
> > $K_{cache} \leftarrow K_{cache} \oplus K_i$;
> > $V_{cache} \leftarrow V_{cache} \oplus V_i$;
> > $h_{l+1} \leftarrow$
> > $\quad h_{l+1} \oplus Attn(Q_i, K_{cache}, V_{cache})$;
>
> **end**
> return h_{l+1} ;

---

where d_{model} represents model hidden size, L represents relative positional bias and mask matrix to make sure only visible positions can be attended to.

3 Experiments and Results

In this section, we first introduce the experimental setup and datasets in § 3.1, and report the main experimental results in § 3.2. Next, we conduct a comparison between the NAR pretraining and the BANG pretraining in § 3.3.

3.1 Setup and Dataset

3.1.1 Setup

We pretrain a base model using the 16GB corpus (Wikipedia and BookCorpus). The base model has a 6-layer encoder and a 6-layer decoder, with a dimension of 768 in their hidden states.

We use the same dictionary with BERT-uncased (Devlin et al., 2018). Similar to MASS (Song et al., 2019), in each block of an input sentence, we mask up to n continuous tokens for prediction, where n is the same as the number of predicting streams. We mask 15% tokens for every 64 tokens in the input sequence span, which makes the target sequence length in each span as 9. To support both AR and NAR pretraining , BANG has 9 predicting streams. We pretrain BANG$_{base}$ from scratch with a learning rate of 3e-4 for 35 epochs and a batch size of 2048.

For AR generation, we load the BANG pretrained model and finetune it with teacher forcing strategy. We use the AR model **Transformer** (Vaswani et al., 2017) without pretraining, strong pretrained AR models **MASS** (Song et al., 2019), **BART** (Lewis et al., 2019), and **Prophet-Net** (Qi et al., 2020) as our baselines. Most of the AR baseline results are collected from **GLGE** (Liu et al., 2020). BANG AR finetuning hyper-parameters are: learning rate 1e-4, warm up steps of 1000, Adam optimizer, the maximum input and output length of 512, and a label smoothing of 0.1. We finetune BANG on each dataset for 10 epochs and select the best model according to the performance on their dev sets. For inference, we set the beam size as 4, length penalty as 1.0, and batch size as 1 to calculate the latency.

For NAR generation, we load the BANG pretrained model and finetune it with all [MASK] inputs. We use NAT (Gu et al., 2017), iNAT (Lee et al., 2018), CMLM (Ghazvininejad et al., 2019), and LevT (Gu et al., 2019) as the NAR baseline models. For these baselines, we select the outputs from the first iteration as the NAR outputs if they are semi-NAR models. For NAR finetuning experiments, the hyper-parameters are the same as AR finetuning except the number of finetuning epochs, since NAR finetuning need more epochs to converge. We finetune BANG for 50 epochs and save a checkpoint for every 10 epochs. We select the best checkpoint based on the performance on dev set. For inference, we set the no-repeat-ngram hyper-parameter as 2 to merge consecutive same

tokens. Since we consider the first [EOS] token as the end signal rather than predicting the target sentence length, we set the maximal output length as 50, 85, 30 for SQuAD, XSum and PersonaChat, respectively.

For semi-NAR generation, we load the BANG pretrained model and finetune it with multiple predicting streams to simultaneously support AR inference and NAR inference. We set the maximum output length and predicting stream numbers as 30, 30, 40 for SQuAD, PersonaChat, XSum, respectively. In other words, the model produced from multi-stream fine-tuning can simultaneously support AR inference, NAR inference and semi-NAR inference. For semi-NAR, we predict the first n_{ar} tokens using a token-by-token manner sequentially, followed by predicting the last $n_{nar} = n - n_{ar}$ tokens in parallel via a single step. We set n_{ar} as 5, and n_{nar} as 25, 25, 35 for SQuAD, PersonaChat and XSum, respectively. The semi-NAR strategy in BANG is quite flexible to support different sequential and parallel combinations, and we leave it as future work for further exploration. For semi-NAR baselines, we choose InsT (Stern et al., 2019), iNAT (Lee et al., 2018), CMLM (Ghazvininejad et al., 2019), LevT (Gu et al., 2019) and set the maximum iteration steps as 10 (one decoding followed by up to nine iterative refinements).

For all downstream tasks, we use 8 NVIDIA Tesla V100 GPUs for finetuning and one single V100 GPU for inference. All the experiments are conducted on the Fairseq (Ott et al., 2019) v0.9.0 codebase and we use the built-in time statistics function to calculate the per-sample inference latency.

3.1.2 Datasets

We conduct experiments on following three popular generation benchmarks:

XSum (Narayan et al., 2018) contains 227K online article and single sentence summary pairs from the British Broadcasting Corporation (BBC). The average input and output lengths are 358.5 and 21.1.

SQuAD 1.1 (Rajpurkar et al., 2016) is a dataset created for machine reading comprehension. After preprocessing, the dataset contains 98K ⟨answer, passage, question⟩ data triples. Input is formatted as ⟨answer [SEP] passage⟩ following GLGE. The average input and output lengths are 149.4 and 11.5, respectively.

PersonaChat (Zhang et al., 2018) is a dataset cre-

| Pattern | Methods | SQuAD 1.1 | | | | Latency |
		ROUGE-L	BLEU-4	METEOR	OVERALL	ms/Sample
Semi-NAR	InsT (Stern et al., 2019)	29.98	2.34	8.15	13.49 (+0.00)	67.61 (4.3x)
	iNAT (Lee et al., 2018)	32.34	3.16	9.18	14.89 (+1.40)	31.59 (2.0x)
	CMLM (Ghazvininejad et al., 2019)	29.60	3.89	9.70	14.40 (+0.91)	106.84 (6.8x)
	LevT (Gu et al., 2019)	30.81	2.68	9.40	14.30 (+0.81)	116.41 (7.4x)
	BANG$_{base}$	**47.39**	**17.62**	**21.69**	**28.90 (+15.41)**	111.11 (7.1x)
NAR	NAT (Gu et al., 2017)	31.51	2.46	8.86	14.28 (+0.02)	17.11 (1.1x)
	iNAT (Lee et al., 2018)	32.44	2.33	8.84	14.54 (+0.28)	16.52 (1.1x)
	CMLM (Ghazvininejad et al., 2019)	31.58	2.51	8.85	14.31 (+0.05)	16.41 (1.0x)
	LevT (Gu et al., 2019)	31.38	2.27	9.14	14.26 (+0.00)	27.52 (1.8x)
	BANG$_{base}$	**44.07**	**12.75**	**18.99**	**25.27 (+11.01)**	**15.69 (1.0x)**
AR	Transformer (Vaswani et al., 2017)	30.73	4.80	10.93	15.49 (+0.00)	233.10 (14.9x)
	MASS$_{base}$ (Song et al., 2019)	**49.48**	20.16	**24.41**	31.35 (+15.86)	N/A
	BART$_{base}$ (Lewis et al., 2019)	42.55	17.08	23.19	27.61 (+12.12)	N/A
	ProphetNet$_{base}$ (Qi et al., 2020)	48.00	19.58	23.94	30.51 (+15.02)	N/A
	BANG$_{base}$	49.32	**21.40**	24.25	**31.66 (+16.17)**	N/A

Table 1: Results of semi-NAR, NAR and AR on SQuAD 1.1.

| Pattern | Methods | XSum | | | | Latency |
		ROUGE-1	ROUGE-2	ROUGE-L	OVERALL	ms/Sample
Semi-NAR	InsT (Stern et al., 2019)	17.65	5.18	16.05	12.96 (+0.00)	63.37 (4.0x)
	iNAT (Lee et al., 2018)	26.95	6.88	22.43	18.75 (+5.79)	31.27 (2.0x)
	CMLM (Ghazvininejad et al., 2019)	29.12	7.70	23.04	19.95 (+6.99)	113.64 (7.1x)
	LevT (Gu et al., 2019)	25.33	7.40	21.48	18.07 (+5.11)	101.01 (6.3x)
	BANG$_{base}$	**34.71**	**11.71**	**29.16**	**25.19 (+12.23)**	109.77 (6.9x)
NAR	NAT (Gu et al., 2017)	24.04	3.88	20.32	16.08 (+0.22)	17.47 (1.1x)
	iNAT (Lee et al., 2018)	24.02	3.99	20.36	16.12 (+0.26)	16.94 (1.1x)
	CMLM (Ghazvininejad et al., 2019)	23.82	3.60	20.15	15.86 (+0.00)	16.88 (1.1x)
	LevT (Gu et al., 2019)	24.75	4.18	20.87	16.60 (+0.74)	27.72 (1.7x)
	BANG$_{base}$	**32.59**	**8.98**	**27.41**	**22.99 (+7.13)**	**15.97 (1.0x)**
AR	Transformer (Vaswani et al., 2017)	30.57	10.47	24.22	21.76 (+0.00)	364.96 (22.9x)
	MASS$_{base}$ (Song et al., 2019)	39.70	17.24	31.91	29.62 (+7.86)	N/A
	BART$_{base}$ (Lewis et al., 2019)	38.79	16.16	30.61	28.52 (+6.76)	N/A
	ProphetNet$_{base}$ (Qi et al., 2020)	39.89	17.12	32.07	29.69 (+7.93)	N/A
	BANG$_{base}$	**41.09**	**18.37**	**33.22**	**30.89 (+9.13)**	N/A

Table 2: Results of semi-NAR, NAR and AR on XSum.

ated for multi-turn conversation with personalizing profiles . After preprocessing, the dataset contains 150k ⟨persona profile description text, conversation history, response⟩ data triples. Input is formatted as ⟨profile [SEP] conversation history⟩ following GLGE. The average input and output lengths are 120.8 and 11.8, respectively.

3.2 Main Results

We present the results for question generation task in Table 1, summarization task in Table 2, and dialog task in Table 3. BANG achieves significantly performance improvements on all tasks consistently for both the NAR and semi-NAR settings. Compared with the best semi-NAR baselines, BANG achieves absolute improvements of 14.01 and 5.24 in overall scores of SQuAD and XSum, respectively. In addition, BANG achieves absolute improvements of 10.73, 6.39, and 5.90 in overall scores of SQuAD, XSUM, and PersonaChat compared with the best NAR baselines, respectively.

This clearly demonstrates the effectiveness of the BANG pretraining. NAR, or semi-NAR results based on BANG are even comparable or better than the AR generation methods without pretraining.

From Table 1, we can see that via BANG pretraining, the results on semi-NAR and NAR generation are significantly improved. Meanwhile, the results on AR generation via BANG pretraining are comparable with strong pretrained baselines. On the other hand, both BANG's inference outperforms Transformer AR baselines in both NAR and semi-NAR categories. For example, the per sample inference latency of the BANG is 15.69ms while that of the Transformer model is 133.10ms, which is 14.9 times slower. On the other hand, in terms of the inference speed, we can see that BANG is slightly better than other NAR generation models. This is mainly due to the fact that BANG NAR uses the first [EOS] token to stop the generation, without the need of length prediction.

Pattern	Methods	PersonaChat					Latency
		B-1	B-2	D-1	D-2	OVERALL	ms/Sample
Semi-NAR	InsT (Stern et al., 2019)	12.63	9.43	0.1	0.3	5.62 (+0.00)	65.27 (4.4x)
	iNAT (Lee et al., 2018)	41.17	32.13	0.1	1.1	18.63 (+13.01)	43.25 (2.9x)
	CMLM (Ghazvininejad et al., 2019)	**44.38**	**35.18**	0.1	0.8	20.12 (+14.50)	105.82 (7.1x)
	LevT (Gu et al., 2019)	24.89	18.94	0.1	0.6	11.13 (+5.51)	80.26 (5.4x)
	BANG$_{base}$	39.82	30.72	**1.9**	**14.2**	**21.66 (+16.04)**	109.17 (7.3x)
NAR	NAT (Gu et al., 2017)	31.53	24.17	0.1	0.8	14.15 (+2.20)	17.86 (1.2x)
	iNAT (Lee et al., 2018)	30.56	23.38	0.1	0.7	13.69 (+1.74)	16.40 (1.1x)
	CMLM (Ghazvininejad et al., 2019)	**31.44**	**24.06**	0.1	0.6	14.05 (+2.10)	16.26 (1.1x)
	LevT (Gu et al., 2019)	26.92	20.47	0.0	0.4	11.95 (+0.00)	27.56 (1.9x)
	BANG$_{base}$	31.11	23.90	**2.5**	**22.7**	20.05 (+8.10)	**14.89 (1.0x)**
AR	Transformer (Vaswani et al., 2017)	38.34	33.60	0.2	0.7	18.21 (+0.00)	204.91 (13.8x)
	MASS$_{base}$ (Song et al., 2019)	41.06	35.75	1.4	6.9	21.28 (+3.07)	N/A
	BART$_{base}$ (Lewis et al., 2019)	**47.60**	**39.36**	1.1	6.1	**23.54(+5.33)**	N/A
	ProphetNet$_{base}$ (Qi et al., 2020)	46.00	38.40	1.3	7.3	23.25 (+5.04)	N/A
	BANG$_{base}$	45.77	35.54	**1.4**	**8.4**	22.78 (+4.57)	N/A

Table 3: Results of semi-NAR, NAR and AR on Dialog. B-1: BLEU-1, B-2: BLEU-2, D-1: Distinct-1. D-2: Distinct-2. D-1 and D-2 are multiplied by 100 according to GLGE.

Similar to the Question Generation tasks, the improvement on the XSum in Table 2 is significant. In all the three categories Semi-NAR, NAR, and AR, BANG can outperform all the existing baselines consistently. Meanwhile, BANG's NAR and semi-NAR results are better than AR generation models without pre-training. Meanwhile, we observe an interesting result when study both the SQuAD and XSum together: BANG semi-NAR consistently outperforms NAR in both tasks, while baseline models show that semi-NAR via iterative refinements can help XSum summarization but often hurt SQuAD question generation.

For the results on PersonaChat in Table 3, we focus on the dialog outputs diversity Distinct-1 and Distinct-2 metrics. For dialog generation, the outputs are very open where the outputs diversity is deeply concerned to avoid boring, and useless dialog responses. We can see the D-1 and D-2 of models without pretraining are very low, which means even the inputs are different, their outputs are highly overlapped or just repetitive. For NAR baseline models, we observe the outputs are often composed of "i, a, an, the" and punctuation, which may share some same parts with target sequence to achieve a reasonable B-1 and B-2 results, but lacking of meaningful responses. For semi-NAR baseline models, we observe the iterative refinement can help produce a fluent response. However, the generated response are often deviated from context with a low accuracy and unrelated to the dialog context. Baseline AR models have high B-1 and B-2 scores via producing some common phrases like "i have", "have a", "a lot", but meaningless dialog responses. For BANG, we observe meaningful response and high diversity in the response, which make it achieve a good diversity score consistently in all three categories: the high-speed NAR generation, the high-performance AR generation, and the semi-NAR generation. We list more examples in appendix § 6 for a detailed illustration.

3.3 BANG vs. NAR

Notice that NAR baseline models in § 3.2 are not pretrained. Here we provide a NAR pretrained model to be compared with BANG. NAR pretraining setting is the same as BANG pretraining, except that we replace the cross-stream visible multistream decoder with a single decoder filled with [MASK] to make it the same as NAR finetuning.

Pretrain Strategy	ROUGE-L	BLEU-4	METEOR
No-pretrain	30.69	2.07	8.22
NAR	41.15	11.38	17.63
BANG	44.07	12.75	18.99

Table 4: NAR results on SQuAD 1.1 question generation with different pretraining strategy.

Pretrain Strategy	ROUGE-1	ROUGE-2	ROUGE-L
No-Pretrain	23.86	4.17	20.12
NAR	30.43	7.66	25.50
BANG	32.50	8.92	27.35

Table 5: NAR results on XSum summarization with different pretraining strategy.

The NAR results after fine-tuning the NAR pretrained model and BANG pretrained model are shown in both Table 4 and Table 5. First, pretraining obviously improves NAR results significantly and consistently in both tasks. Second,

after the same NAR fine-tuning, the BANG pretrained model consistently outperforms the vanilla NAR pretrained model. This clearly demonstrates that the proposed pretraining strategy in BANG via bridging both AR and NAR is critical to achieve a better performance in NAR generation.

4   Related Work

AR models have been developed for a long time. Recent works show that pretraining on large scale text will guarantee a consistent improvements on downstream generation tasks. GPT series work (Radford et al., 2018, 2019; Brown et al., 2020) pretrains decoders with the task of next token prediction and convert different downstream tasks into language models. MASS (Song et al., 2019) masks continuous words' spans from input sentences to predict. BART (Lewis et al., 2019) uses denoising task to pretrain Transformer. Prophet-Net (Qi et al., 2020) deploys future tokens' prediction to enhance AR generation ability. Dialog-GPT (Zhang et al., 2019b) is pretrained for conversational response generation. XLNet (Yang et al., 2019) utilizes AR pretraining for downstream natural language understanding tasks, and we also borrow 2-stream strategy from XLNet for BANG cross-stream visible n-stream decoder.

NAR and semi-NAR models are proposed to accelerate natural language generation. NAT (Gu et al., 2017) is proposed as a NAR generation model to decode the whole target sequence at one time step. iNAT (Lee et al., 2018) refines outputs with multi-turn post-processing. InsT (Stern et al., 2019) predicts inserting positions and inserting tokens at each iteration. CMLM (Ghazvininejad et al., 2019) firstly predict all target words with NAR generation and maskout-regenerate low confidence words. LevT (Gu et al., 2019) considers two basic operations insertion and deletion at each iteration.

Some work propose to use AR generation facilitate NAR generation and some work propose future tokens' prediction to facilitate AR generation. Curriculum learning (Bengio et al., 2009) is used to benefit NAR generation from AR generation such as Guo et al. (2020). Future tokens' prediction is used to benefit AR generation from NAR generation such as Qi et al. (2020); Goodman et al. (2020).

BANG is benefited from these different NAR and AR generation models to support NAR, AR,

semi-NAR generation.

5   Conclusion

We propose a new natural language generation pretraining model named BANG. BANG bridges NAR and AR generation with cross-stream visible n-stream strategy and large scale pretraining. BANG supports NAR, AR and semi-NAR generation with the same pretrained model. Experiments show that BANG can significant improve the NAR and semi-NAR generation performance, and provide comparable results with strong AR pretraining models. BANG shows that NAR and semi-NAR generation can be applied to general natural language generation tasks with acceptable performance. BANG is powerful and flexible to support more diverse semi-NAR generation strategies and finetuning strategies, which we leave as future work.

References

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. *arXiv preprint arXiv:1904.09324*.

Sebastian Goodman, Nan Ding, and Radu Soricut. 2020. Teaforn: Teacher-forcing with n-grams. *arXiv preprint arXiv:2010.03494*.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2017. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. In *Advances in Neural Information Processing Systems*, pages 11181–11191.

Junliang Guo, Xu Tan, Linli Xu, Tao Qin, Enhong Chen, and Tie-Yan Liu. 2019. Fine-tuning by curriculum learning for non-autoregressive neural machine translation. *arXiv preprint arXiv:1911.08717*.

Junliang Guo, Xu Tan, Linli Xu, Tao Qin, Enhong Chen, and Tie-Yan Liu. 2020. Fine-tuning by curriculum learning for non-autoregressive neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7839–7846.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. *arXiv preprint arXiv:1802.06901*.

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*.

Dayiheng Liu, Yu Yan, Yeyun Gong, Weizhen Qi, Hang Zhang, Jian Jiao, Weizhu Chen, Jie Fu, Linjun Shou, Ming Gong, et al. 2020. Glge: A new general language generation evaluation benchmark. *arXiv preprint arXiv:2011.11928*.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *EMNLP*, pages 1797–1807.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv:1904.01038*.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training. *arXiv preprint arXiv:2001.04063*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

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.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *EMNLP*, pages 2383–2392.

Raphael Shu, Jason Lee, Hideki Nakayama, and Kyunghyun Cho. 2020. Latent-variable non-autoregressive neural machine translation with deterministic inference using a delta posterior. In *AAAI*, pages 8846–8853.

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*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J Liu. 2019a. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. *arXiv preprint arXiv:1912.08777*.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *ACL*, pages 2204–2213.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2019b. Dialogpt: Large-scale generative pre-training for conversational response generation. *arXiv preprint arXiv:1911.00536*.

6 Appendix Case Study

In this section, we choose two samples from SQuAD question generation and XSum summarization to help illustrate how BANG helps NAR and semi-NAR generation in Table 6. We also provide more examples for PersonaChat in Table 7. We choose NAT and CMLM as NAR and semi-NAR baselines, respectively, since NAT is the first NAR generation model and CMLM has the best performance according to the experimental results in § 3.2.

For results on SQuAD in Table 6, we can see that SQuAD question generation is easier than XSum with more fluent NAR generation because of the shorter output length and the question sentence format. Baseline NAR output sentence structures seem reasonable but the described items are wrong.

SQuAD 1.1	
Input	Forbes [SEP] A self - described " modern - day feminist " , Beyoncé creates songs that are often characterized by themes of love , relationships , and monogamy , as well as female sexuality and empowerment. ...... Forbes magazine also listed her as the most powerful female musician of 2015 .
Golden	which magazine declared her the most dominant woman musician ?
NAT	where is the music of s in called ?
CMLM NAR	what is the is the music in music ?
CMLM semi-NAR	what is the name of the most popular music ?
BANG NAR	who magazine listed her as the most powerful musician ?
BANG semi-NAR	who listed her as the most powerful female musician of 2015 ?
BANG AR	which magazine listed beyonce as the most powerful female musician in 2015 ?
XSum	
Input	She became Kenya's first high-profile athlete to fail a test, when she tested positive for performance-enhancing drugs in September.Jeptoo, 33, says she may have been prescribed some banned substances at a local hospital after a road accident.She has become the 45th Kenyan athlete to have failed a doping test. ...... She has won the previous three Boston and two Chicago marathons and also previously won the Stockholm, Paris, Milan and Lisbon marathons.
Golden	kenya's rita jeptoo, winner of the boston and chicago marathons, has been banned for two years after failing a drugs test.
NAT	kenya kenyan s kenya - kenya has , the a doping .
CMLM NAR	kenyan s je -oo has banned , the - doping .
CMLM semi-NAR	kenyan olympic gold medallist laura jefioo has been banned from the country of an anti - doping .
BANG NAR	kenyan marathon runner rita jeptoo has been given for two year ban after failing doping athletics .
BANG semi-NAR	kenyan marathon runner kiba jeptoo has been banned for two years for failing doping .
BANG AR	kenya ' s world marathon champion lydia jeptoo has been banned for two years by athletics kenya for failing a doping test .

Table 6: Case study for different NAR, AR and semi-NAR models on SQuAD 1.1 and XSum.

We remark that with iterative refinements the results become more fluent, but the modification is often only based on the first generated outputs. For example, the example continues to describe the wrong item "music" rather than "musician". Besides, although CMLM semi-NAR outputs are fluent with refinements, the ability of understanding the task target is still weak and heavily relies on the output in the first iteration. On the contrary, BANG is able to raise proper question on correct object. Although BANG NAR generation mixes "who listed" and "which magazine listed" into "who magazine listed" because of the weakness of NAR generation ability, BANG semi-NAR generation fixes that problem into "who listed her as the" to serve as a high-quality sub-sequence hint. In the AR generation result, BANG even finds "her" represents "beyonce" from the input context.

For results on XSum in Table 6, we can see that baseline NAR models nearly fail in generating meaningful sentences. Baseline NAR results are composed with key phrases and key words while BANG NAR generation are fluent with insignificant errors. CMLM semi-NAR results are fluent and close to target sequence via iterative modifi-

cation. BANG semi-NAR output has a mistake at the end of the sentence where "failing doping" should be "failing a doping test". BANG AR output contains nearly all the important details. A common problem for all the outputs is about the runner name "jeptoo". In XSum training samples, they often come with a complete names composed with the first name and the second name. Models learn to generate complete human names. In this given input, only the second name "jeptoo" is given and the models have no idea about her first name. All the models fake a first name for her to generate a complete description, which shows that natural language generation models have a bias on the training data and may fabricate some details with maximum likelihood.

For results on PersonaChat in Table 7, we can see though the baseline models have high B-1 B-2 evaluation results in Table 3, they fail to generate meaningful dialog responses. NAR baseline outputs are composed with common words while semi-NAR baseline outputs are composed with common sentences. Although the common words, phrases, sentences have n-gram overlap with target to have high B-1 and B-2 scores, they are hard to compose

meaningful responses. On the contrary, no matter what the generation pattern is, BANG models seek to generate diverse and meaningful responses.

Input	i have two dogs and one cat . ...[SEP] ... do you have pets ? no i do not , do you ? i have two dogs and one cat . ... [SEP] ... nice , where do you live ? i resign in north dakota i like to make crafts . ... [SEP] ... i live in a small town in ohio . i work in a factory .... [SEP] ... i live in a small town in ohio . so we are semi close neighbors . i like to dance at the club ....[SEP] hi there , how are you today ? i just had surgery . ...[SEP] ... i am great ! just got home from working with dogs all day ! i like to dance at the club . ... [SEP] ... oh you have dogs ? i have two chow chow . i just had surgery .... [SEP] ... i have dogs and i get to train them at work , too ! i like to dance at the club .... [SEP] ... i just had a surgery not long ago .
Golden	yes . two dogs and a cat . they are my babies . i live in texas . i love riding my bike here . so we are semi close neighbors . seems like it . have you been to the rock hall ? i am great ! just got home from working with dogs all day ! oh you have dogs ? i have two chow chow . i love dogs ! i have dogs and i get to train them at work , too ! i wish i could take my dogs out . i just had a surgery not long ago . i hope you feel better soon . what do you like to do for fun ?
NAT	i is , the ? yes , i a cat that is . do you any pets ? that is . do you any pets ? i am good , how are you ? i am i . just . i . my dogs . i , i my you . i , i . i . dogs . i is i a dogs .
CMLM NAR	i , you a you ? i , i . i you a you . i is i . i . i the you . i is i . i the live . i am doing , how are you ? i am i . you ? i ! . i do a you . i ! . i . you dogs . i ! . i . i a dogs .
CMLM semi-NAR	i have a few years , i have a lot of my dog . i have a few dogs , i have a lot of them . i have a lot of dogs . i have a lot of them . i have a lot of dogs . i have a lot of them . i am good , how are you ? i am doing well , i am doing well . how are you ? i am a teacher . i am a big teacher . but i for a lot . i am a good teacher . i am looking for a living . what do you work ? i am a teacher . i just am a lot job for work a lot .
BANG NAR	yes . i have and a cat . is i live in north dakota . i ride bike . maybe you like ? i is close . you about ? . i am doing great thanks ! oh nice ! i have dogs have big dogs . yes ! dogs ! my sweet ! that is are dogs ! i got to school surgery , surgery that sounds bad ?
BANG semi-NAR	yes , i have two dogs and a cat . i live in north carolina . i ride bike . i have never been there . i homes there that is good . do you have any hobbies ? i am good . just got back from . oh , i love dogs ! i have two dogs . i have two ! i have a sweet tooth ! that is awesome ! i have not work i school i am so sorry to hear that . what you do ?
BANG AR	yes , i have two dogs and a cat . i live in california . i love riding my bike there . do you have any hobbies ? i like to make crafts . yes , we are . what do you do for a living ? i am great ! just got back from dance practice . you ? that sounds like a lot of fun . i have two dogs . yes , i do ! i have a big sweet tooth . what kind of food do you like ? what kind of work do you do ? oh , i am sorry to hear that . what kind of surgery ?

Table 7: Case study for different NAR, AR and semi-NAR models on PersonaChat.