DAGA: Data Augmentation with a Generation Approach for Low-resource Tagging Tasks

Bosheng Ding[*12] **Linlin Liu**[*12] **Lidong Bing**[2] **Canasai Kruengkrai**[†3]
Thien Hai Nguyen[2] **Shafiq Joty**[1] **Luo Si**[2] **Chunyan Miao**[1]

[1]Nanyang Technological University, Singapore
[2]DAMO Academy, Alibaba Group [3]National Institute of Informatics, Japan

{bosheng.ding, linlin.liu, l.bing, thienhai.nguyen, luo.si}@alibaba-inc.com
canasai@nii.ac.jp {srjoty, ascymiao}@ntu.edu.sg

data augmentation

- Compared with the above-mentioned downstream tasks like translation and classification, sequence tagging is more fragile when it is confronted with data augmentation noises due to the finer granularity of the (token-level) task.

Labeled Sentence Linearization

- Tags are inserted before the corresponding words during linearization and thus treated as modifiers of these words
- After sentence linearization, we add special tokens [BOS] and [EOS] to the beginning and the end of each sentence, respectively.

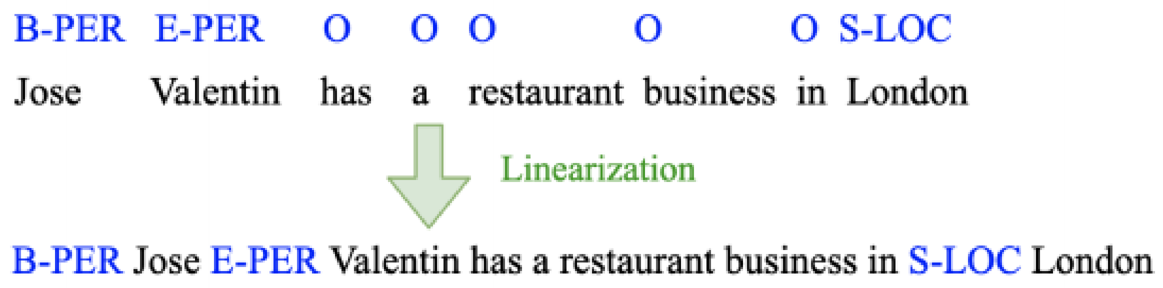

B-PER E-PER O O O O O S-LOC
Jose Valentin has a restaurant business in London
Linearization
B-PER Jose E-PER Valentin has a restaurant business in S-LOC London

Figure 1: An example of labeled sentence linearization. All words and their tags are paired up by inserting tags before (or after) the words (*O* tags removed).

RNNLM

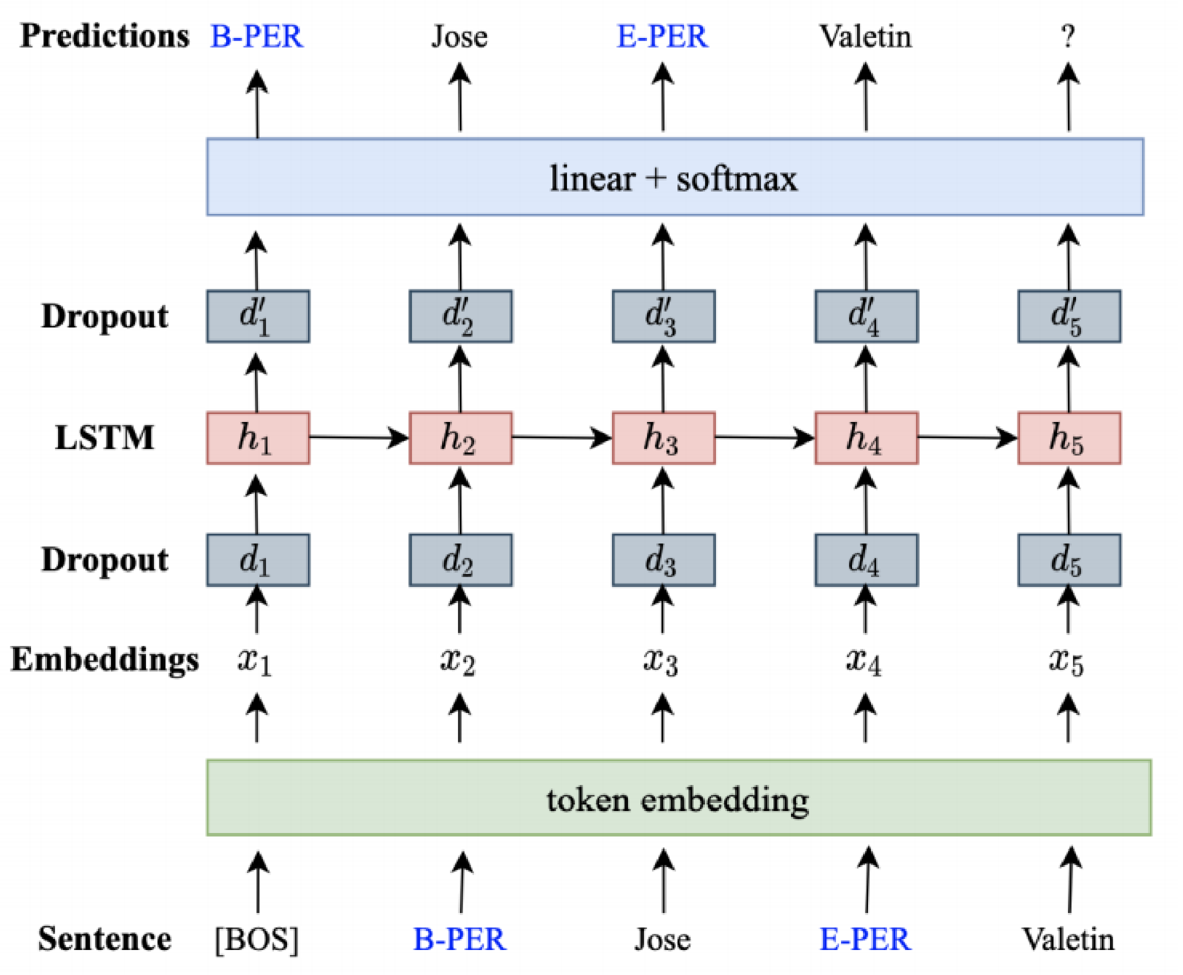


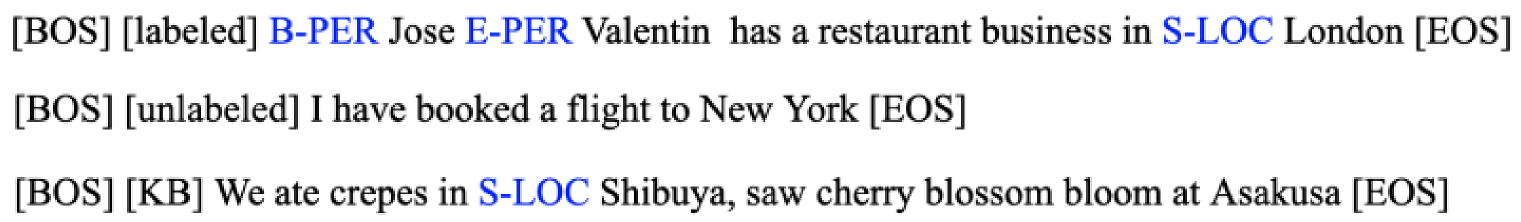
Figure 2: Language model architecture with LSTM.

Post-Processing

- 1) Delete sentences with no tags;
- 2) Delete sentences where all words are [unk];
- 3) Delete sentences with incorrect tag prefix orders (e.g., having E-LOC before B-LOC in NER data);
- 4) Delete sentences that contain same sequences of words but different tags.

Conditional Generation

We prepend one of these condition tags
{[labeled],[unlabeled],[KB]}
at the beginning of each sequence to mark their origin, where KB means the sequence is labeled by matching a knowledge base against the unlabeled data.



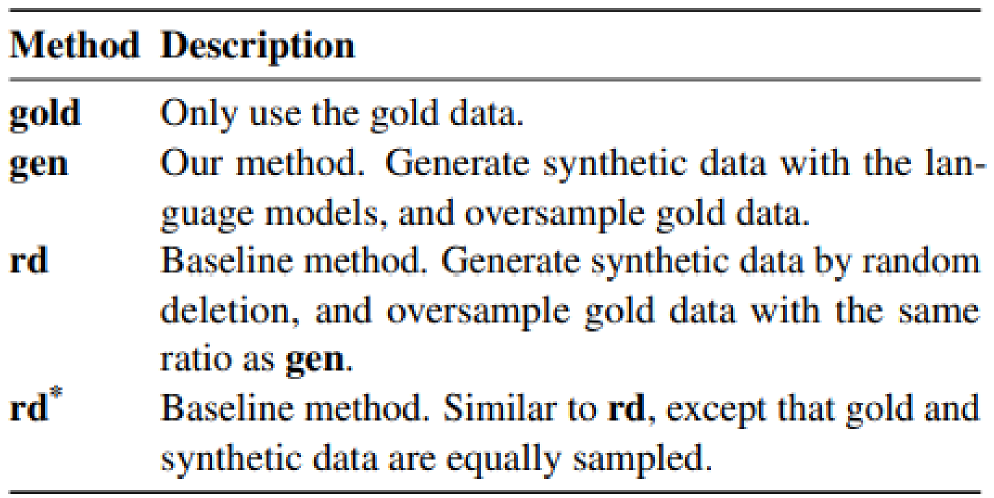
[BOS] [labeled] B-PER Jose E-PER Valentin has a restaurant business in S-LOC London [EOS]

[BOS] [unlabeled] I have booked a flight to New York [EOS]

[BOS] [KB] We ate crepes in S-LOC Shibuya, saw cherry blossom bloom at Asakusa [EOS]

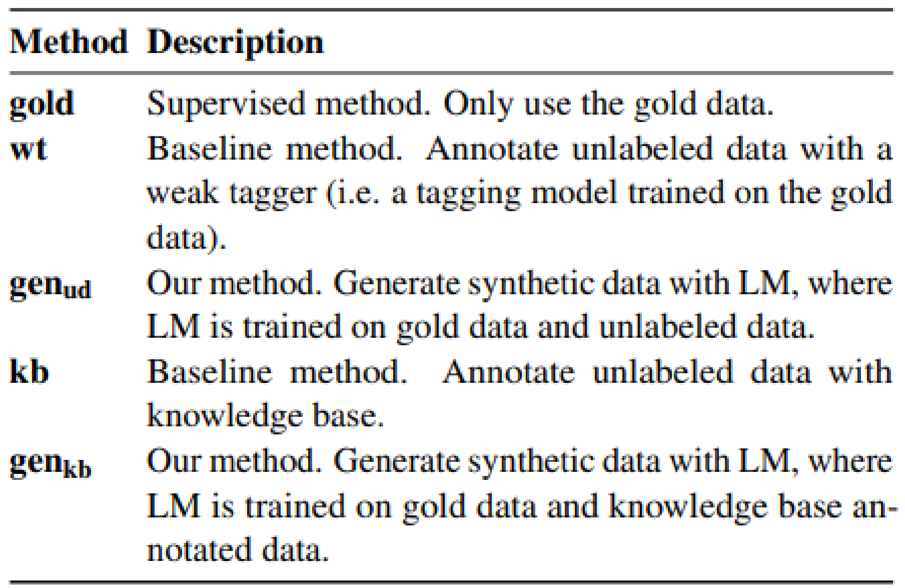
Figure 3: An example of conditional generation. The first sequence is from gold NER data. The second is from unlabeled data, so no labels. The third is labeled by knowledge base matching, where *Asakusa* cannot be labeled due to incomplete knowledge coverage.

Experiments



Method	Description
gold	Only use the gold data.
gen	Our method. Generate synthetic data with the language models, and oversample gold data.
rd	Baseline method. Generate synthetic data by random deletion, and oversample gold data with the same ratio as **gen**.
rd*	Baseline method. Similar to **rd**, except that gold and synthetic data are equally sampled.

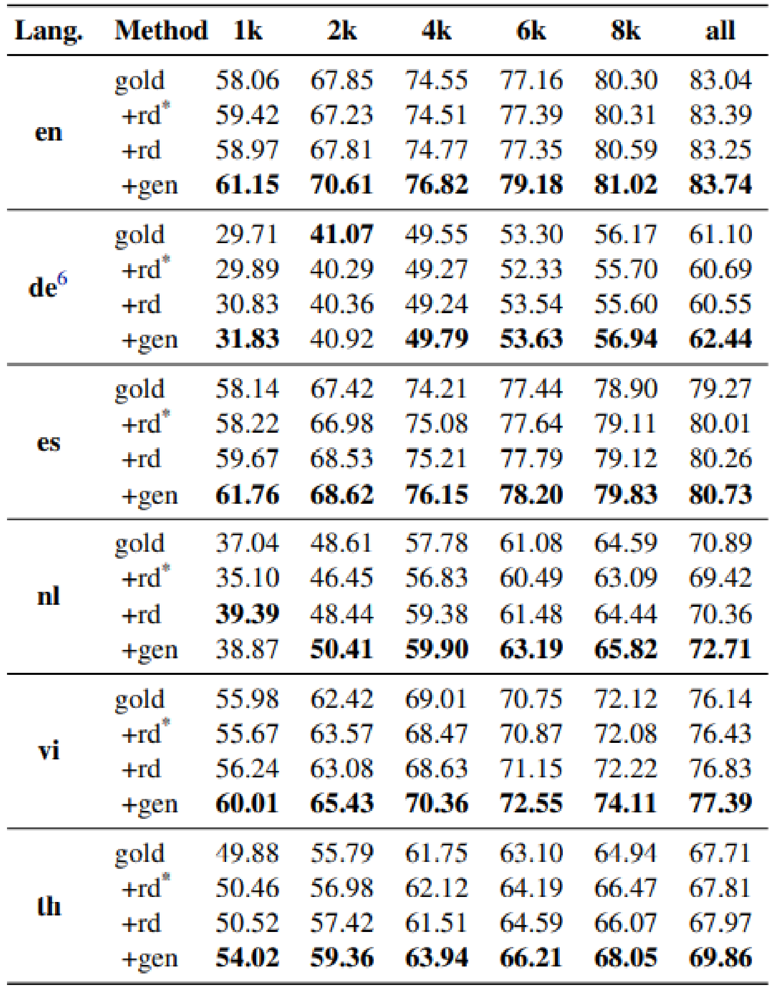
Table 1: Data sources for the supervised setting.



Method	Description
gold	Supervised method. Only use the gold data.
wt	Baseline method. Annotate unlabeled data with a weak tagger (i.e. a tagging model trained on the gold data).
$\textbf{gen}_{\textbf{ud}}$	Our method. Generate synthetic data with LM, where LM is trained on gold data and unlabeled data.
kb	Baseline method. Annotate unlabeled data with knowledge base.
$\textbf{gen}_{\textbf{kb}}$	Our method. Generate synthetic data with LM, where LM is trained on gold data and knowledge base annotated data.

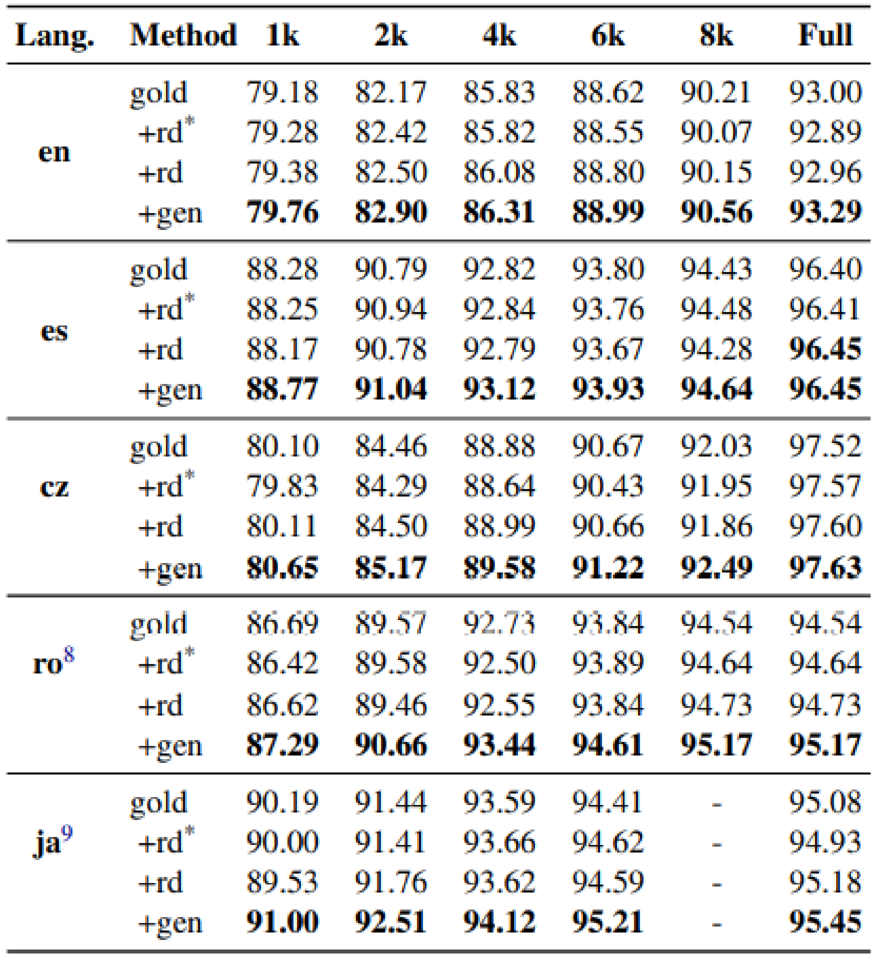
Table 5: Data sources for the semi-supervised setting.

Supervised Experiments



Lang.	Method	1k	2k	4k	6k	8k	all
en	gold	58.06	67.85	74.55	77.16	80.30	83.04
	+rd*	59.42	67.23	74.51	77.39	80.31	83.39
	+rd	58.97	67.81	74.77	77.35	80.59	83.25
	+gen	**61.15**	**70.61**	**76.82**	**79.18**	**81.02**	**83.74**
de[6]	gold	29.71	**41.07**	49.55	53.30	56.17	61.10
	+rd*	29.89	40.29	49.27	52.33	55.70	60.69
	+rd	30.83	40.36	49.24	53.54	55.60	60.55
	+gen	**31.83**	40.92	**49.79**	**53.63**	**56.94**	**62.44**
es	gold	58.14	67.42	74.21	77.44	78.90	79.27
	+rd*	58.22	66.98	75.08	77.64	79.11	80.01
	+rd	59.67	68.53	75.21	77.79	79.12	80.26
	+gen	**61.76**	**68.62**	**76.15**	**78.20**	**79.83**	**80.73**
nl	gold	37.04	48.61	57.78	61.08	64.59	70.89
	+rd*	35.10	46.45	56.83	60.49	63.09	69.42
	+rd	**39.39**	48.44	59.38	61.48	64.44	70.36
	+gen	38.87	**50.41**	**59.90**	**63.19**	**65.82**	**72.71**
vi	gold	55.98	62.42	69.01	70.75	72.12	76.14
	+rd*	55.67	63.57	68.47	70.87	72.08	76.43
	+rd	56.24	63.08	68.63	71.15	72.22	76.83
	+gen	**60.01**	**65.43**	**70.36**	**72.55**	**74.11**	**77.39**
th	gold	49.88	55.79	61.75	63.10	64.94	67.71
	+rd*	50.46	56.98	62.12	64.19	66.47	67.81
	+rd	50.52	57.42	61.51	64.59	66.07	67.97
	+gen	**54.02**	**59.36**	**63.94**	**66.21**	**68.05**	**69.86**

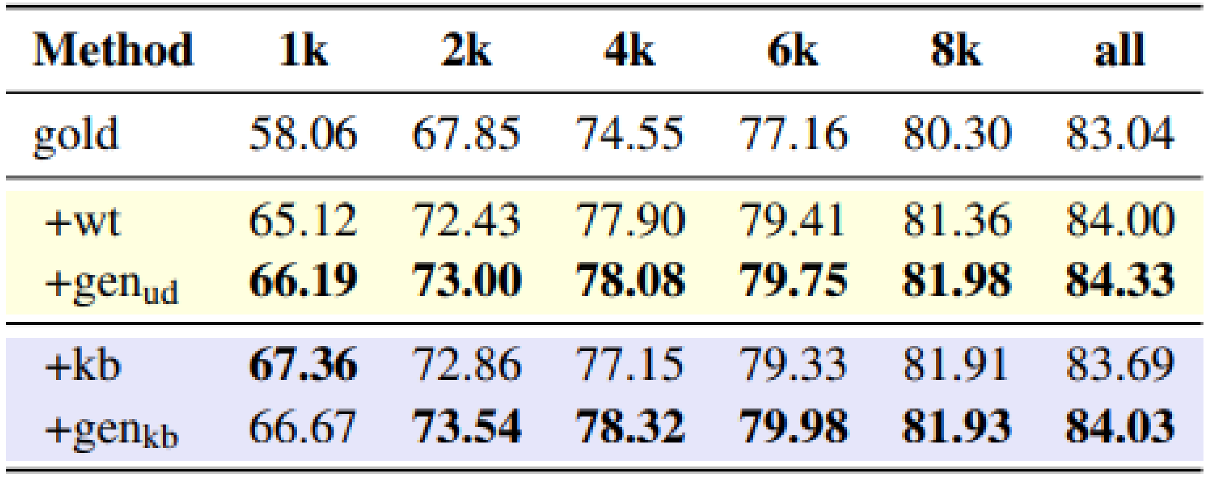
Table 2: Named entity recognition micro F1.



Lang.	Method	1k	2k	4k	6k	8k	Full
en	gold	79.18	82.17	85.83	88.62	90.21	93.00
	+rd*	79.28	82.42	85.82	88.55	90.07	92.89
	+rd	79.38	82.50	86.08	88.80	90.15	92.96
	+gen	**79.76**	**82.90**	**86.31**	**88.99**	**90.56**	**93.29**
es	gold	88.28	90.79	92.82	93.80	94.43	96.40
	+rd*	88.25	90.94	92.84	93.76	94.48	96.41
	+rd	88.17	90.78	92.79	93.67	94.28	**96.45**
	+gen	**88.77**	**91.04**	**93.12**	**93.93**	**94.64**	**96.45**
cz	gold	80.10	84.46	88.88	90.67	92.03	97.52
	+rd*	79.83	84.29	88.64	90.43	91.95	97.57
	+rd	80.11	84.50	88.99	90.66	91.86	97.60
	+gen	**80.65**	**85.17**	**89.58**	**91.22**	**92.49**	**97.63**
ro[8]	gold	86.69	89.57	92.73	93.84	94.54	94.54
	+rd*	86.42	89.58	92.50	93.89	94.64	94.64
	+rd	86.62	89.46	92.55	93.84	94.73	94.73
	+gen	**87.29**	**90.66**	**93.44**	**94.61**	**95.17**	**95.17**
ja[9]	gold	90.19	91.44	93.59	94.41	-	95.08
	+rd*	90.00	91.41	93.66	94.62	-	94.93
	+rd	89.53	91.76	93.62	94.59	-	95.18
	+gen	**91.00**	**92.51**	**94.12**	**95.21**	-	**95.45**

Table 3: POS tagging accuracy.

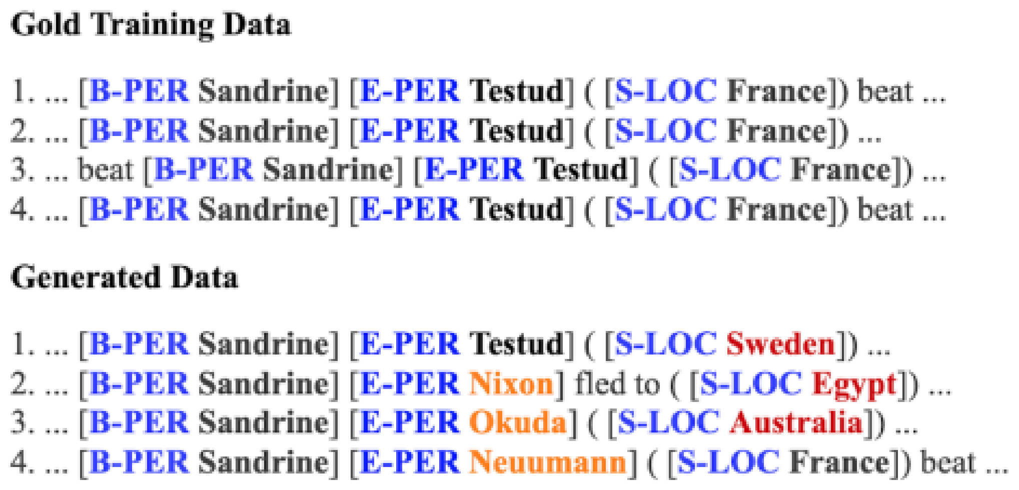
Conditional Generation



Method	**1k**	**2k**	**4k**	**6k**	**8k**	**all**
gold	58.06	67.85	74.55	77.16	80.30	83.04
+wt	65.12	72.43	77.90	79.41	81.36	84.00
+gen$_{\text{ud}}$	**66.19**	**73.00**	**78.08**	**79.75**	**81.98**	**84.33**
+kb	**67.36**	72.86	77.15	79.33	81.91	83.69
+gen$_{\text{kb}}$	66.67	**73.54**	**78.32**	**79.98**	**81.93**	**84.03**

Table 6: Semi-supervised NER F1.

A Closer Look at Synthetic Data



Gold Training Data

1. ... [**B-PER Sandrine**] [**E-PER Testud**] ([**S-LOC France**]) beat ...
2. ... [**B-PER Sandrine**] [**E-PER Testud**] ([**S-LOC France**]) ...
3. ... beat [**B-PER Sandrine**] [**E-PER Testud**] ([**S-LOC France**]) ...
4. ... [**B-PER Sandrine**] [**E-PER Testud**] ([**S-LOC France**]) beat ...

Generated Data

1. ... [**B-PER Sandrine**] [**E-PER Testud**] ([**S-LOC Sweden**]) ...
2. ... [**B-PER Sandrine**] [**E-PER Nixon**] fled to ([**S-LOC Egypt**]) ...
3. ... [**B-PER Sandrine**] [**E-PER Okuda**] ([**S-LOC Australia**]) ...
4. ... [**B-PER Sandrine**] [**E-PER Neuumann**] ([**S-LOC France**]) beat ...

Figure 4: An illustration of diversity of generated data. The name "*Sandrine*" in the gold training data always pairs up with "*Testud*" in sentences.

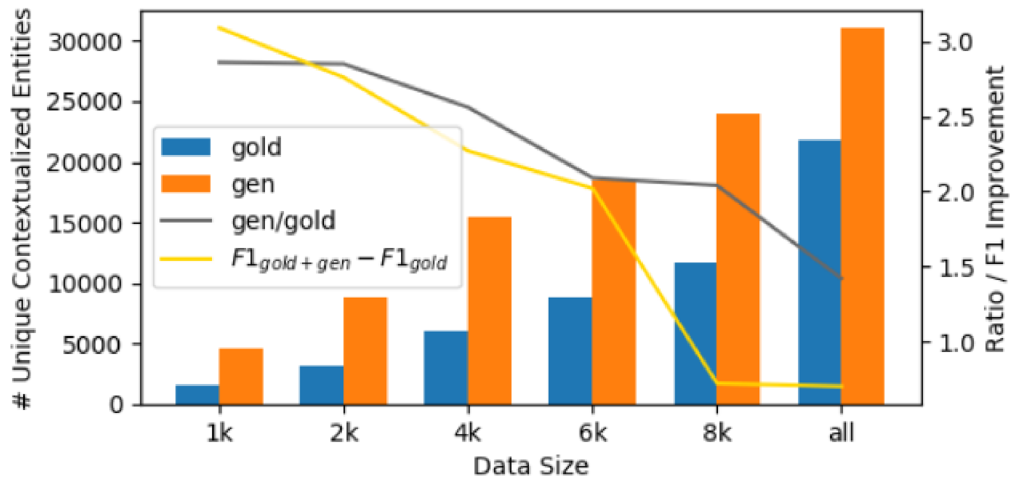


Figure 5: Statistics of unique contextualized entities.

Tell Me How to Ask Again: Question Data Augmentation with Controllable Rewriting in Continuous Space

Dayiheng Liu[♠][*] **Yeyun Gong**[†] **Jie Fu**[♢] **Yu Yan**[‡]
Jiusheng Chen[‡] **Jiancheng Lv**[♠] **Nan Duan**[†] **Ming Zhou**[†]
[♠]College of Computer Science, Sichuan University [†]Microsoft Research Asia
[♢] Mila [‡]Microsoft
losinuris@gmail.com

CRQDA

- we propose a new QDA method called Controllable Rewriting based Question Data Augmentation (CRQDA), which can generate both new context-relevant answerable questions and unanswerable questions
- There are two components of CRQDA: (i) A Transformer-based autoencoder whose encoder maps the question into a latent representation. Then its decoder reconstructs the question from the latent representation. (ii) A MRC model, which is pre-trained on the original dataset.

Overview

- Given a question q from the original dataset D, we first map the question q into a continuous embedding space
- Then we revise the question embeddings by gradient-based optimization with the guidance of the MRC model
- Finally, the revised question embeddings are inputted to the Transformer-based autoencoder to generate a new question data.

MODEL

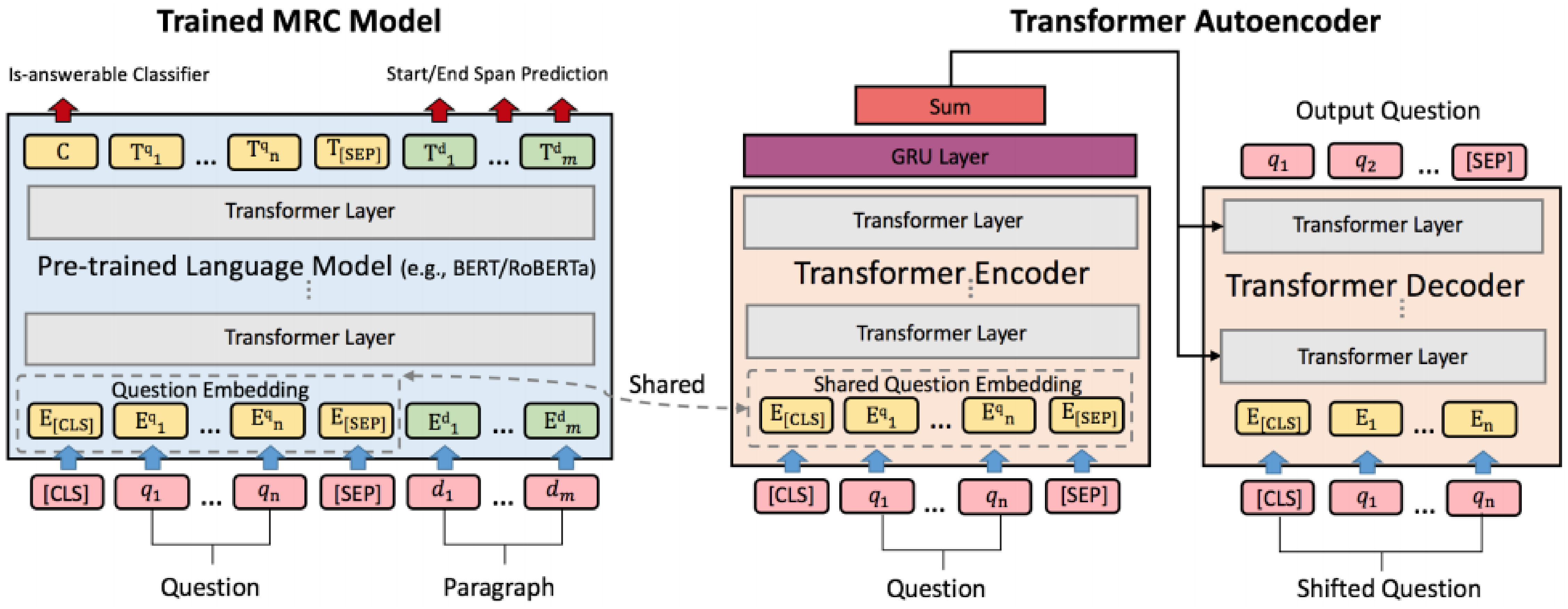


Figure 1: The architecture of CRQDA.

Problem Formulation

5-tuple data: (q, d, s, e, t), where $|\mathcal{D}|$ is the data size, $q = \{q_1, ..., q_n\}$ is a tokenized question with length n, $d = \{d_1, ..., d_m\}$ is a tokenized paragraph with length m, $s, e \in \{0, 1, ..., m-1\}$ are inclusive indices pointing to the start and end of the answer span, and $t \in \{0, 1\}$ represents whether the question q is answerable or unanswerable with d. Given a data tuple (q, d, s, e, t), we aim to rewrite q to a new answerable or unanswerable question q' and obtain a new data tuple (q', d, s, e, t') that fulfills certain requirements: (*i*) The generated answerable question can be answered with the answer span (s, e) with d, while the generated unanswerable question cannot be answered with d. (*ii*) The generated question should be relevant to the original question q and paragraph d. (*iii*) The augmented dataset $\mathcal{D}'$ should be able to further improve the performance of the MRC models.

For the data tuple (q, d, s, e, t), the total loss of MRC model can be written as

$$\begin{aligned}\mathcal{L}_{\text{mrc}} &= \lambda \mathcal{L}_a(t) + \mathcal{L}_s(s) + \mathcal{L}_e(e), \\ &= -\lambda \log P_a(t) - \log P_s(s) - \log P_e(e),\end{aligned} \tag{5}$$

where λ is a hyper-parameter.

Pre-trained Language Model based MRC Model

Following Devlin et al. (2018), given a data tuple (q, d, s, e, t), we concatenate a "[CLS]" token, the tokenized question q with length n, a "[SEP]" token, the tokenized paragraph d with length m, and a final "[SEP]" token. We feed the resulting sequence into the BERT model. The question q and paragraph d are first mapped into two sequence of embeddings:

$$\mathbf{E}^q, \mathbf{E}^d = \mathrm{BertEmbedding}(q, d), \qquad (1)$$

$\mathbf{E}^q$ and $\mathbf{E}^d$ are further fed into BERT layers which consist of multiple Transformer layers (Vaswani et al., 2017) to obtain the final hidden representations $\{\mathbf{C}, \mathbf{T}^q_1, ..., \mathbf{T}^q_n, \mathbf{T}_{[SEP]}, \mathbf{T}^d_1, ..., \mathbf{T}^d_m\}$ as shown in Figure 1. The representation vector $\mathbf{C} \in \mathbb{R}^h$ corresponding to the first input token ([CLS]) are fed into a binary classification layer to output the probability of whether the question is answerable:

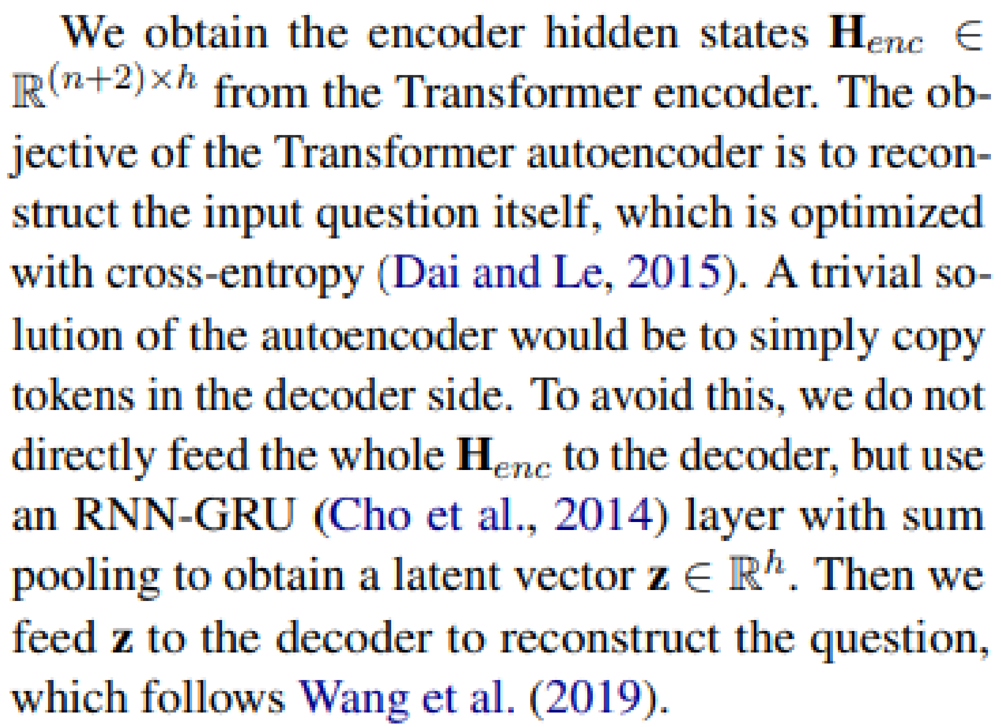
$$P_a(\text{is-answerable}) = \mathrm{Sigmoid}(\mathbf{C}\mathbf{W}_c^T + \mathbf{b}_c), \quad (2)$$

where $\mathbf{W}_c \in \mathbb{R}^{2\times h}$ and $\mathbf{b}_c \in \mathbb{R}^2$ are trainable parameters. The final hidden representations of paragraph $\{\mathbf{T}^d_1, ..., \mathbf{T}^d_m\} \in \mathbb{R}^{m\times h}$ are inputted into two classifier layer to output the probability of the start position and the end position of the answer span:

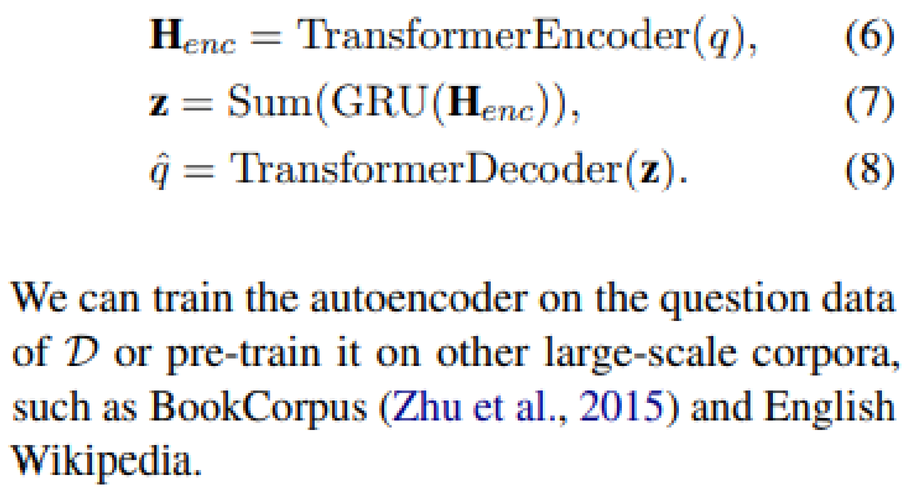
$$P_s(i =< \text{start} >) = \mathrm{Sigmoid}(\mathbf{T}^d_i \mathbf{W}_s^T + b_s), \quad (3)$$
$$P_e(i =< \text{end} >) = \mathrm{Sigmoid}(\mathbf{T}^d_i \mathbf{W}_e^T + b_e), \quad (4)$$

where $\mathbf{W}_s \in \mathbb{R}^{1\times h}$, $\mathbf{W}_e \in \mathbb{R}^{1\times h}$, $b_s \in \mathbb{R}^1$, and $b_e \in \mathbb{R}^1$ are trainable parameters.

Transformer-based Autoencoder



We obtain the encoder hidden states $\mathbf{H}_{enc} \in \mathbb{R}^{(n+2)\times h}$ from the Transformer encoder. The objective of the Transformer autoencoder is to reconstruct the input question itself, which is optimized with cross-entropy (Dai and Le, 2015). A trivial solution of the autoencoder would be to simply copy tokens in the decoder side. To avoid this, we do not directly feed the whole $\mathbf{H}_{enc}$ to the decoder, but use an RNN-GRU (Cho et al., 2014) layer with sum pooling to obtain a latent vector $\mathbf{z} \in \mathbb{R}^{h}$. Then we feed $\mathbf{z}$ to the decoder to reconstruct the question, which follows Wang et al. (2019).



$$\mathbf{H}_{enc} = \mathrm{TransformerEncoder}(q), \tag{6}$$

$$\mathbf{z} = \mathrm{Sum}(\mathrm{GRU}(\mathbf{H}_{enc})), \tag{7}$$

$$\hat{q} = \mathrm{TransformerDecoder}(\mathbf{z}). \tag{8}$$

We can train the autoencoder on the question data of $\mathcal{D}$ or pre-train it on other large-scale corpora, such as BookCorpus (Zhu et al., 2015) and English Wikipedia.

Rewriting Question with Gradient-based Optimization

- Given an answerable question q, the goals of the rewriting are:
- (I) the revised question embedding should make the pre-trained MRC model predict the question from answerable to unanswerable with the paragraph d;
- (II) The modification size of Eq should be adaptive to prevent the revision of Eq from falling into local optimum.
- (III) The revised question $\hat{q}$ should be similar to the original q, which helps to improve the robustness of the model.

Rewriting Question with Gradient-based Optimization

For goal-(I), we take the label $t' = 0$, which denotes the label of question is unanswerable, to calculate the loss $\mathcal{L}_a(t')$ and the gradient of $\mathbf{E}^q$ by the pre-trained MRC model (see the red line in Figure 2). We iteratively revise $\mathbf{E}^q$ with gradients from the pre-trained MRC model until the MRC model predicts the question is unanswerable with the revised $\mathbf{E}^{q'}$ as its input, which means the $P_a(t'|\mathbf{E}^{q'})$ is large than a threshold β_t. Note that here we use the gradient to only modify $\mathbf{E}^q$, and all the model parameters during rewriting process are fixed. The process of each iteration can be written as:

$$\mathbf{E}^{q'} = \mathbf{E}^q - \eta(\nabla_{\mathbf{E}^q}\mathcal{L}_a(t')), \tag{9}$$

where η is the step size. Similarly, we can revise the $\mathbf{E}^q$ of a data tuple (q, d, s, e, t) to generate a new answerable question whose answer is still the original answer span (s, e) as follows:

$$\mathbf{E}^{q'} = \mathbf{E}^q - \eta\left(\nabla_{\mathbf{E}^q}(\lambda\mathcal{L}_a(t) + \mathcal{L}_s(s) + \mathcal{L}_e(e))\right). \tag{10}$$

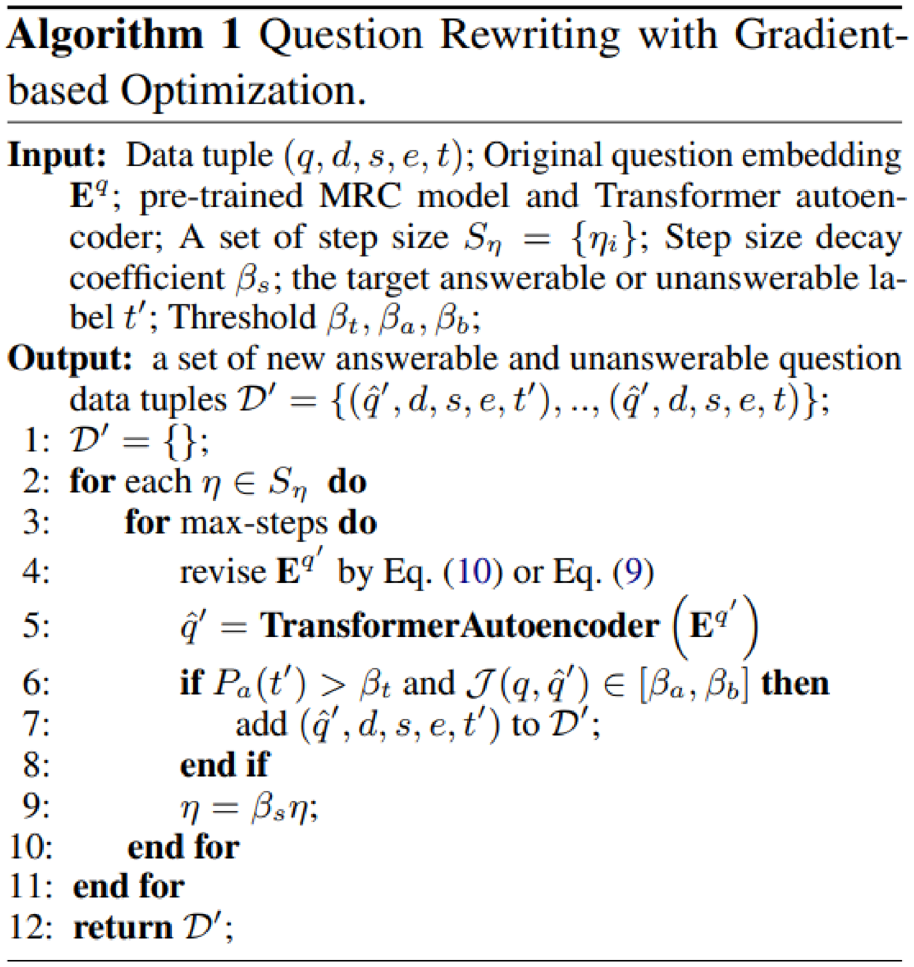
Rewriting the answerable question into another answerable question can be seen as a special constrained paraphrasing, which requires that the question after the paraphrasing is context-relevant answerable and its answer remains unchanged.

For goal-(II), we follow (Wang et al., 2019) to employ the dynamic-weight-initialization method to allocate a set of step-sizes $S_\eta = \{\eta_i\}$ as initial step-sizes. For each initial step-size, we perform a pre-defined max-step revision with the step size value decay (corresponds to Algorithm 1 line 2-11) to find the target question embedding. For goal-(III), we select the $\hat{q}'$ whose unigram word overlap rate with the original question q is within a threshold range $[\beta_a, \beta_b]$. The unigram word overlap is computed by:

$$\mathcal{J}(q, \hat{q}') = \frac{\text{count}(w_q \cap w_{\hat{q}})}{\text{count}(w_q \cup w_{\hat{q}})}, \tag{11}$$

here w_q is the word in q and $w_{\hat{q}}$ is the word in $\hat{q}'$. The whole question rewriting procedure is summarized in Algorithm 1.

Rewriting Question with Gradient-based Optimization



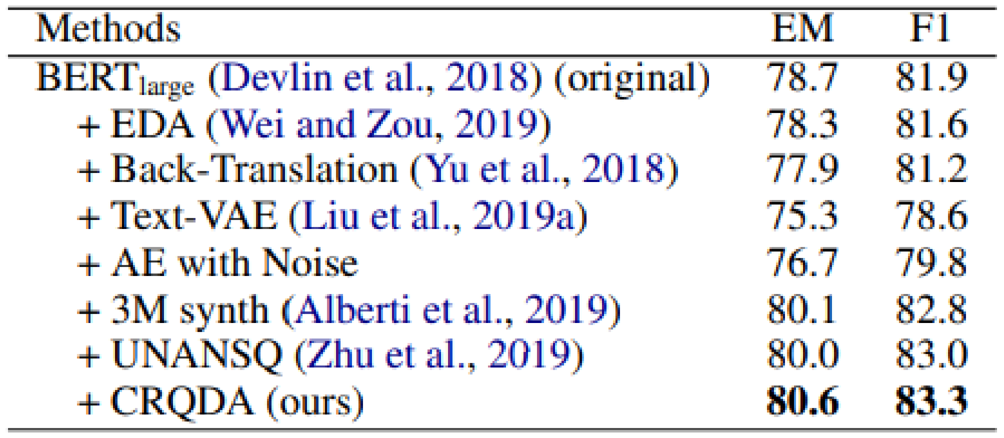
Algorithm 1 Question Rewriting with Gradient-based Optimization.

Input: Data tuple (q, d, s, e, t); Original question embedding $\mathbf{E}^q$; pre-trained MRC model and Transformer autoencoder; A set of step size $S_\eta = \{\eta_i\}$; Step size decay coefficient β_s; the target answerable or unanswerable label t'; Threshold $\beta_t, \beta_a, \beta_b$;

Output: a set of new answerable and unanswerable question data tuples $\mathcal{D}' = \{(\hat{q}', d, s, e, t'), .., (\hat{q}', d, s, e, t)\}$;

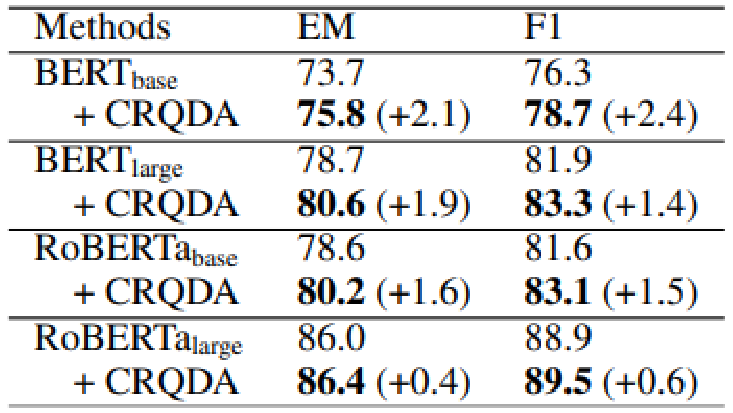
```
 1: $\mathcal{D}' = \{\}$;
 2: for each $\eta \in S_\eta$ do
 3:     for max-steps do
 4:         revise $\mathbf{E}^{q'}$ by Eq. (10) or Eq. (9)
 5:         $\hat{q}' = \mathbf{TransformerAutoencoder}\left(\mathbf{E}^{q'}\right)$
 6:         if $P_a(t') > \beta_t$ and $\mathcal{J}(q, \hat{q}') \in [\beta_a, \beta_b]$ then
 7:             add $(\hat{q}', d, s, e, t')$ to $\mathcal{D}'$;
 8:         end if
 9:         $\eta = \beta_s \eta$;
10:     end for
11: end for
12: return $\mathcal{D}'$;
```

Experiments



Methods	EM	F1
BERT$_{\text{large}}$ (Devlin et al., 2018) (original)	78.7	81.9
+ EDA (Wei and Zou, 2019)	78.3	81.6
+ Back-Translation (Yu et al., 2018)	77.9	81.2
+ Text-VAE (Liu et al., 2019a)	75.3	78.6
+ AE with Noise	76.7	79.8
+ 3M synth (Alberti et al., 2019)	80.1	82.8
+ UNANSQ (Zhu et al., 2019)	80.0	83.0
+ CRQDA (ours)	**80.6**	**83.3**

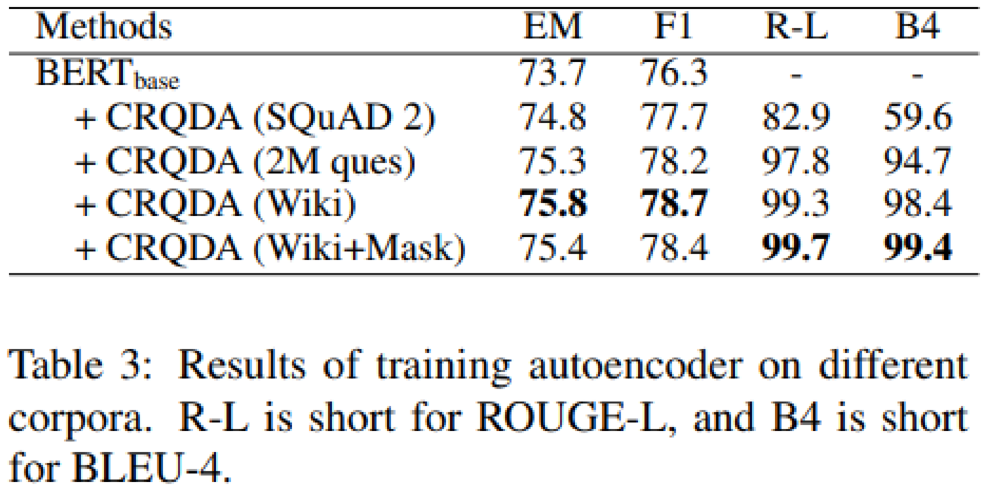
Table 1: Comparison results on SQuAD 2.0.



Methods	EM	F1
BERT$_{\text{base}}$	73.7	76.3
+ CRQDA	**75.8** (+2.1)	**78.7** (+2.4)
BERT$_{\text{large}}$	78.7	81.9
+ CRQDA	**80.6** (+1.9)	**83.3** (+1.4)
RoBERTa$_{\text{base}}$	78.6	81.6
+ CRQDA	**80.2** (+1.6)	**83.1** (+1.5)
RoBERTa$_{\text{large}}$	86.0	88.9
+ CRQDA	**86.4** (+0.4)	**89.5** (+0.6)

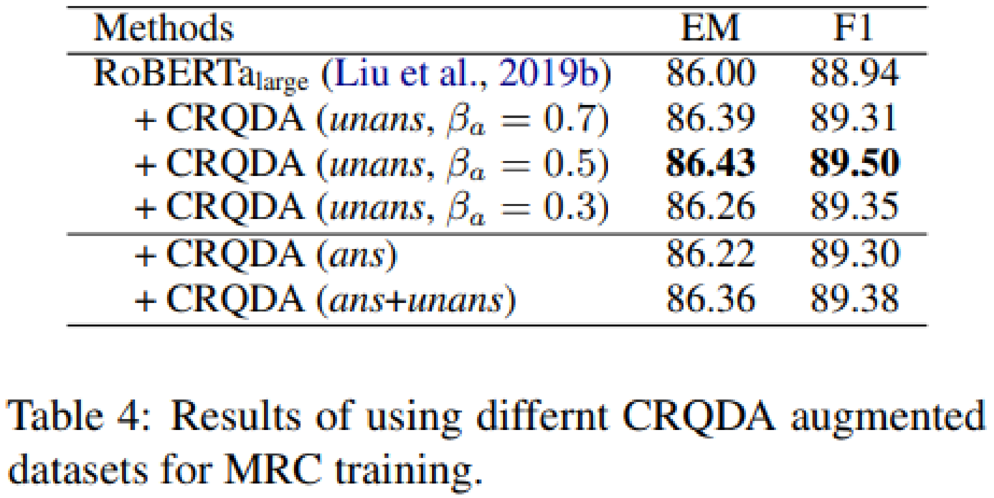
Table 2: Results of different MRC models with CRQDA on SQuAD 2.0.

Experiments



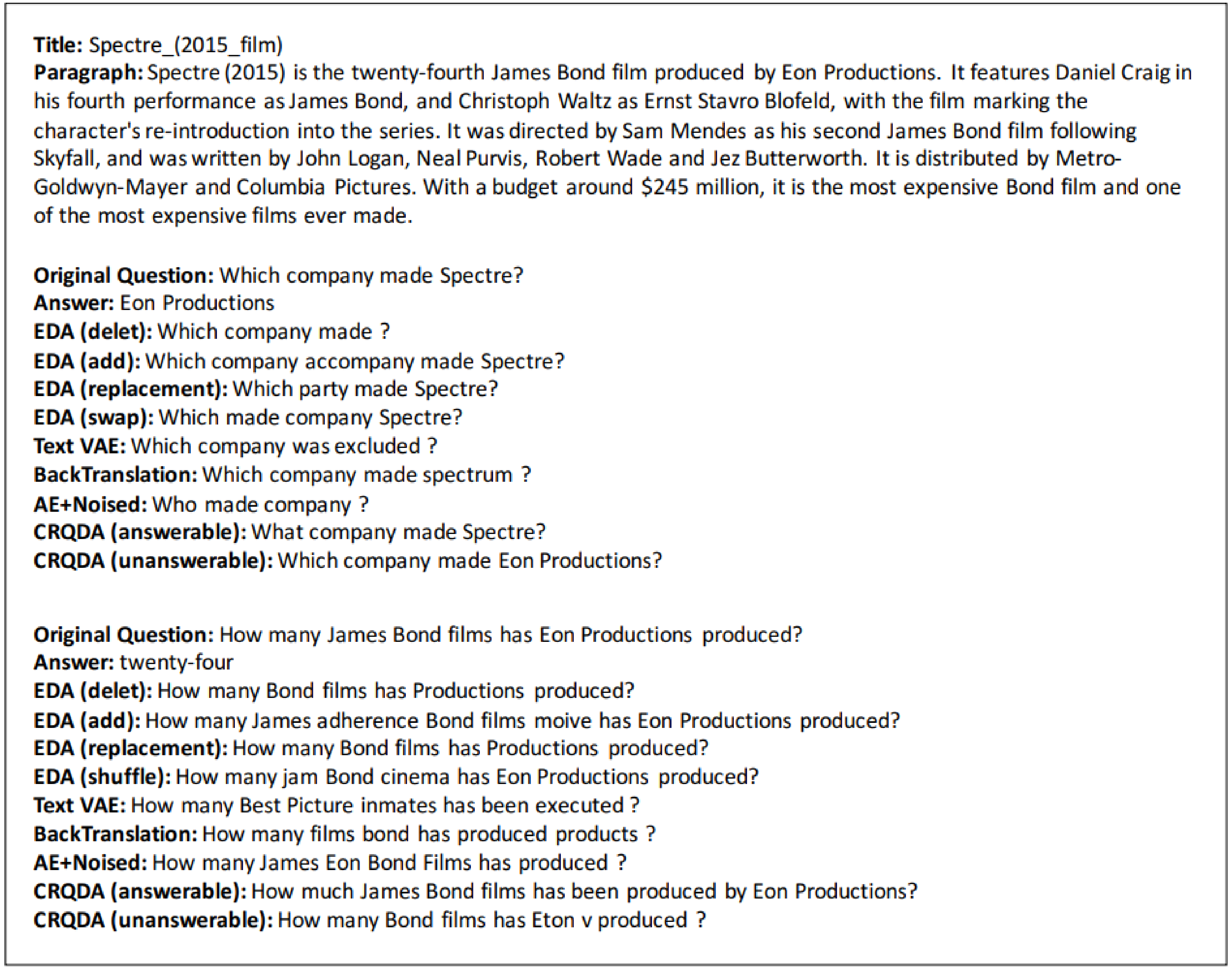
Methods	EM	F1	R-L	B4
BERT$_{base}$	73.7	76.3	-	-
+ CRQDA (SQuAD 2)	74.8	77.7	82.9	59.6
+ CRQDA (2M ques)	75.3	78.2	97.8	94.7
+ CRQDA (Wiki)	**75.8**	**78.7**	99.3	98.4
+ CRQDA (Wiki+Mask)	75.4	78.4	**99.7**	**99.4**

Table 3: Results of training autoencoder on different corpora. R-L is short for ROUGE-L, and B4 is short for BLEU-4.



Methods	EM	F1
RoBERTa$_{large}$ (Liu et al., 2019b)	86.00	88.94
+ CRQDA (*unans*, $\beta_a = 0.7$)	86.39	89.31
+ CRQDA (*unans*, $\beta_a = 0.5$)	**86.43**	**89.50**
+ CRQDA (*unans*, $\beta_a = 0.3$)	86.26	89.35
+ CRQDA (*ans*)	86.22	89.30
+ CRQDA (*ans+unans*)	86.36	89.38

Table 4: Results of using diffrent CRQDA augmented datasets for MRC training.



Title: Spectre_(2015_film)
Paragraph: Spectre (2015) is the twenty-fourth James Bond film produced by Eon Productions. It features Daniel Craig in his fourth performance as James Bond, and Christoph Waltz as Ernst Stavro Blofeld, with the film marking the character's re-introduction into the series. It was directed by Sam Mendes as his second James Bond film following Skyfall, and was written by John Logan, Neal Purvis, Robert Wade and Jez Butterworth. It is distributed by Metro-Goldwyn-Mayer and Columbia Pictures. With a budget around $245 million, it is the most expensive Bond film and one of the most expensive films ever made.

Original Question: Which company made Spectre?
Answer: Eon Productions
EDA (delet): Which company made ?
EDA (add): Which company accompany made Spectre?
EDA (replacement): Which party made Spectre?
EDA (swap): Which made company Spectre?
Text VAE: Which company was excluded ?
BackTranslation: Which company made spectrum ?
AE+Noised: Who made company ?
CRQDA (answerable): What company made Spectre?
CRQDA (unanswerable): Which company made Eon Productions?

Original Question: How many James Bond films has Eon Productions produced?
Answer: twenty-four
EDA (delet): How many Bond films has Productions produced?
EDA (add): How many James adherence Bond films moive has Eon Productions produced?
EDA (replacement): How many Bond films has Productions produced?
EDA (shuffle): How many jam Bond cinema has Eon Productions produced?
Text VAE: How many Best Picture inmates has been executed ?
BackTranslation: How many films bond has produced products ?
AE+Noised: How many James Eon Bond Films has produced ?
CRQDA (answerable): How much James Bond films has been produced by Eon Productions?
CRQDA (unanswerable): How many Bond films has Eton v produced ?