Set Generation Networks for End-to-End Knowledge Base Population

Dianbo Sui[1,2], **Chenhao Wang**[1,2], **Yubo Chen**[1,2], **Kang Liu**[1,2], **Jun Zhao**[1,2], **Wei Bi**[3]
[1] National Laboratory of Pattern Recognition, Institute of Automation, CAS
[2] School of Artificial Intelligence, University of Chinese Academy of Sciences
[3] Tencent AI Lab
{dianbo.sui, chenhao.wang, yubo.chen, kliu, jzhao}@nlpr.ia.ac.cn

Abstract

The task of knowledge base population (KBP) aims to discover facts about entities from texts and expand a knowledge base with these facts. Previous studies shape end-to-end KBP as a machine translation task, which is required to convert unordered fact into a sequence according to a pre-specified order. However, the facts stated in a sentence are unordered in essence. In this paper, we formulate end-to-end KBP as a direct set generation problem, avoiding considering the order of multiple facts. To solve the set generation problem, we propose networks featured by transformers with non-autoregressive parallel decoding. Unlike previous approaches that use an autoregressive decoder to generate facts one by one, the proposed networks can directly output the final set of facts in one shot. Furthermore, to train the networks, we also design a set-based loss that forces unique predictions via bipartite matching. Compared with cross-entropy loss that highly penalizes small shifts in fact order, the proposed bipartite matching loss is invariant to any permutation of predictions. Benefiting from getting rid of the burden of predicting the order of multiple facts, our proposed networks achieve state-of-the-art (SoTA) performance on two benchmark datasets.

1 Introduction

Nowadays, knowledge bases (KBs) are valuable resources, which can provide back-end support for various knowledge-centric services of real-world applications, such as question answering systems (Cui et al., 2017), dialogue systems (Madotto et al., 2018) and recommendation systems (Guo et al., 2020). However, high-quality KBs still rely almost exclusively on human-curated structured or semi-structured data (Mesquita et al., 2019). Such a reliance on human curation is a major barrier to creating always-up-to-date KBs.

To overcome this barrier, knowledge base population (KBP) is proposed (Ji and Grishman, 2011; Getman et al., 2018), which is a task of automatically discovering facts about entities from texts and expanding the incomplete KB with these facts. As shown in Table 1, a KBP system is required to take a given sentence as input and transform it into a set of facts. A fact is in the form of <*h*, *r*, *t*>, where *h* is a head entity, *t* is a tail entity, and *r* is a predicate that falls in a predefined set of predicates. Following Trisedya et al. (2019), we also assume that *h* and *t* are existing entities in the given KB while the fact <*h*, *r*, *t*> does not exist in the KB, since KBs typically have much better coverage on entities than on relationships.

Conventionally, KBP is solved by several individual components in a pipeline manner (Shin et al., 2015; Angeli et al., 2015; Zhang et al., 2017; Chaganty et al., 2017; Mesquita et al., 2019), typically including (1) entity discovery or named entity recognition (Tjong Kim Sang and De Meulder, 2003), (2) entity linking (Milne and Witten, 2008) and (3) relation extraction (Zelenko et al., 2003). Entity discovery seeks to locate and classify named entities mentioned in text into predefined categories (e.g., people, organizations and locations). Entity linking is a task to disambiguate these recognized entity mentions by linking them to a reference KB. Relation extraction aims to predict semantic relations between pairs of entities. Though widely used in practice, this pipeline architecture is inherently prone to error propagation between its components (Trisedya et al., 2019).

To alleviate error propagation, some end-to-end KBP methods are proposed, such as Liu et al. (2018); Trisedya et al. (2019). These methods are all based on the sequence-to-sequence (seq2seq) framework (Sutskever et al., 2014; Cho et al., 2014). Under this framework, end-to-end KBP is treated as a translation of a sentence into a sequence of fact elements (entity or predicate). Considering the running example in Table 1, a seq2seq model would translate the sentence "`President Obama`

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 9650–9660
November 7–11, 2021.

Input Sentence:
President Obama welcomed President Xi Jinping of China to visit the United States.
Output Facts:
<Q76, P39, Q11696>; <Q15031, P39, Q655407>

Table 1: An example of KBP. In this example, "Obama", "President of United States", "Xi Jinping", and "President of People's Republic of China" are mapped to their unique Wikidata identifiers "Q76", "Q11696", "Q15031" and "Q655407" respectively, and the semantic relation "P39" ("position held" in Wikidata) is labeled between these entity pairs.

`welcomed President Xi Jinping of China to visit the United States"` into a sequence of Wikidata identifiers "`Q76 P39 Q11696 Q15031 P39 Q655407`". From the output sequence, two new facts would be derived.

Despite the success of existing end-to-end methods for KBP, they are still limited by two widely used modules in the seq2seq framework: autoregressive decoder and cross-entropy loss. The reasons are as follows: the facts contained in a sentence have no intrinsic order in essence. Considering the running example in Table 1, predicting <`Q76, P39, Q11696`> first and then <`Q15031, P39, Q655407`> has no difference from predicting <`Q15031, P39, Q655407`> first and then <`Q76, P39, Q11696`>. However, in order to adapt the autoregressive decoder, whose output is a sequence, unordered target facts must be sorted in a certain order during the training phase. Meanwhile, cross-entropy is a permutation-sensitive loss function, where a penalty is incurred for every fact that is predicted out of the position. Consequently, the current seq2seq models not only need to learn how to generate facts, but also are required to consider the extraction order of multiple facts.

In this paper, we formulate the end-to-end KBP task as a set generation problem, avoiding considering the order of multiple facts. In order to address the set generation problem, we propose end-to-end networks, dubbed "Set Generation Networks" (SGN), featured by transformers with non-autoregressive parallel decoding and bipartite matching loss. In detail, there are three parts in the proposed set generation networks: a sentence encoder, a set generator, and a set based loss function. First, we adopt the transformer-based encoder (Vaswani et al., 2017) to represent the sentence. After that, since the autoregressive decoder must generate items one by one in order, such a decoder is not suitable for generating unordered sets. In contrast, we leverage a transformer-based non-autoregressive decoder (Gu et al., 2018) as the set generator, which can predict all facts at once. Finally, in order to assign a predicted fact to a unique ground truth, we propose the bipartite matching loss function inspired by the assignment problem in operation research (Kuhn, 1955; Munkres, 1957; Edmonds and Karp, 1972). Compared with the cross-entropy loss that highly penalizes small shifts in fact order, the proposed loss function is invariant to any permutation of predictions; thus it is suitable for evaluating the difference between the ground truth set and the prediction set.

To summarize, our main contributions include:

- We formulate the end-to-end knowledge base population as a set generation problem.
- We combine non-autoregressive parallel decoding with the bipartite matching loss function to solve this problem.
- Our proposed method yields SoTA results on two benchmark datasets, and we perform various experiments to verify its effectiveness.

2 Methodology

The goal of end-to-end KBP is to identify all possible facts $Y = \{< h_1, r_1, t_1 >, ..., < h_n, r_n, t_n >\}$ stated in a given sentence X to enrich the given reference KB. To solve this task, we propose end-to-end set generation networks, which are shown in Figure 1. Four key components of the proposed neural networks will be elaborated in the following section. Concretely, we first present the joint learning of word and entity embeddings in Section 2.1, which are the basis of the proposed networks. Next, we introduce the sentence encoder in Section 2.2, which can represent each token in a given sentence based on its bidirectional context. Then, we illustrate the set generator in Section 2.3, which is based on a non-autoregressive decoder to generate a set of facts in a single pass. Finally, we describe a set-based loss in Section 2.4, called bipartite matching loss, which forces unique matching between predicted and ground truth facts.

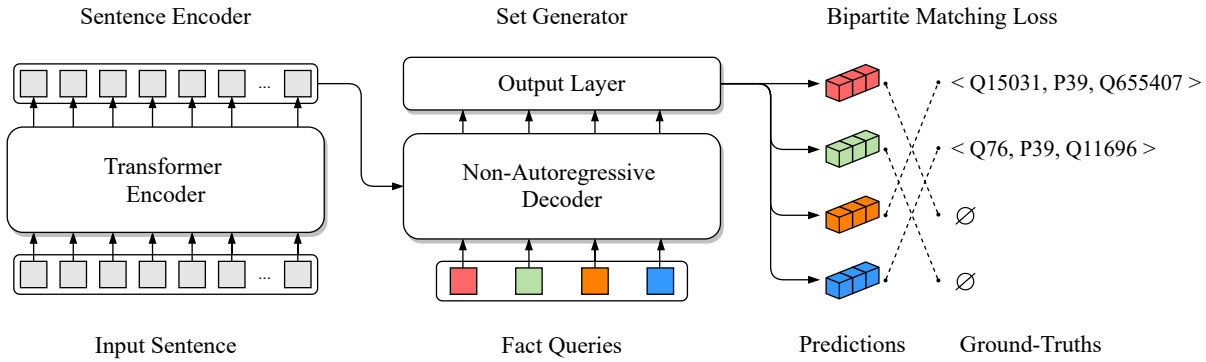

Figure 1: The main architecture of set generation networks. The set generation networks predict the final set of facts in parallel by combining a transformer-based encoder with a non-autoregressive decoder. In the training phrase, bipartite matching uniquely assigns predictions with ground truths to provide accurate training signals.

2.1 Joint Learning of Word and Entity Embeddings

In the first step, we jointly embed words, entities and predicates into the same vector space. To achieve this, we combine the anchor context model (Yamada et al., 2016) to compute the word embeddings with TransE (Bordes et al., 2013) to compute the entity and predicate embeddings.

Specifically, we first utilize the anchor context model to establish the interaction between the entity and word embeddings. In this model, a modified Wikipedia corpus is generated by replacing the hyperlinks with the related entity identifiers, and a skip-gram model (Mikolov et al., 2013) is trained on this corpus to compute the word and entity embeddings. Formally, given a sequence $[w_1, w_2, ..., w_T]$, the loss function of the anchor context model is:

$$J_W = -\sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log P(w_{t+j}|w_t) \quad (1)$$

where c is the size of the context window, w_t denotes the target word, and w_{t+j} is its context word. The conditional probability $P(w_{t+j}|w_t)$ is computed using the following softmax function:

$$P(w_{t+j}|w_t) = \frac{\exp(\mathbf{V}_{w_{t+j}}^{\mathsf{T}} \mathbf{U}_{w_t})}{\sum_{w \in W} \exp(\mathbf{V}_{w_t}^{\mathsf{T}} \mathbf{U}_w)} \quad (2)$$

where W is a set containing all words in the vocabulary, and $\mathbf{V}_w, \mathbf{U}_w \in \mathbb{R}^d$ represent the vectors of word w in matrices $\mathbf{V}$ and $\mathbf{U}$ (Mikolov et al., 2013).

Then, in order to map the entity and predicate embeddings into the same continuous vector space, a TransE model is trained on all facts in the given reference KB (Note that facts mentioned in the test set are not included in the KB). The loss function of the TransE model is defined as :

$$J_E = \sum_{t_r \in T_r} \sum_{t'_r \in T'_r} max(0, [\gamma + f(t_r) - f(t'_r)]) \quad (3)$$

where T_r is the set of valid facts in the give KG, T'_r is the set of corrupted facts, γ is the margin and $f = \|h + r - t\|_2$. The corrupted facts are created by replacing the head or tail entity of a valid fact with a random entity, and act as negative samples in training.

To jointly train the anchor context model and the TransE model, a hybrid loss function J is used, in which the above loss functions are linearly combined.

$$J = J_W + J_E \quad (4)$$

After training, we can obtain word embeddings $\mathbf{V_w}$, entity embeddings $\mathbf{V_e}$ and predicate embeddings $\mathbf{V_p}$, which coexist in the same continuous vector space.

2.2 Sentence Encoder

The sentence encoder is designed to generate the context-aware representation of each token in a sentence. Following previous work (Trisedya et al., 2019), we utilize the transformer-based encoder (Vaswani et al., 2017), which is a stack of layers composed of two sub-layers: multi-head self-attention followed by a feed-forward sub-layer. Specific steps to generate context-aware representations are as follows: First, a given sentence X is segmented with tokens. Then, these segmented tokens are projected to the continuous vector space by using the pretrained word embeddings $\mathbf{V_w}$ (mentioned in Section 2.1). After that, word embeddings of these tokens are fed into the transformer-based encoder. Finally, the transformer-based encoder

outputs the context-aware representation of each token in the sentence. The output of the transformer-based encoder is denoted as $\mathbf{H}_e \in \mathbb{R}^{l \times d}$, where l is the sentence length and d is the output dimension of the transformer-based encoder.

2.3 Set Generator

The goal of the set generator is to generate a set of predicted facts based on the output of the sentence encoder.

Input. The input of the set generator includes the output of the sentence encoder $\mathbf{H}_e \in \mathbb{R}^{l \times d}$ and m trainable embeddings, which are called fact queries. With m fact queries, the set generator is able to generate a fixed-size set of m predictions for each sentence. To meet all conditions, m is set to be the largest number of facts stated in a sentence.

Non-Autoregressive Decoder. The backbone of the set generator is a non-autoregressive decoder. The non-autoregressive decoder is composed of a stack of N identical layers. In each layer, there are multi-head self-attention mechanism to model the relationship between facts, and multi-head inter-attention mechanism to fuse the information of the given sentence. Notably, compared with the autoregressive decoder, the non-autoregressive decoder is not limited by an autoregressive factorization of the output, so there is no need to prevent earlier decoding steps from accessing information from later steps. Thus, there is no casual mask used in the multi-head self-attention mechanism. Instead, we use the unmasked self-attention. Through the non-autoregressive decoder, m fact queries are transformed into m output embeddings, which are denoted as $\mathbf{H}_d \in \mathbb{R}^{m \times d}$.

Output Layer. The output embeddings $\mathbf{H}_d$ are then independently decoded into predicates and entities by three feed forward networks, resulting m final predicted facts. Specifically, given an output embedding $\mathbf{h}_d \in \mathbb{R}^d$ in $\mathbf{H}_d$, the predicted distribution of the predicate is :

$$\mathbf{p}^r = \mathrm{softmax}(\mathbf{V_p}\mathbf{h}_\mathrm{d}) \tag{5}$$

where $\mathbf{V_p} \in \mathbb{R}^{p \times d}$ are the pretrained predicate embeddings, and p is the total number of predicate types. Note that a special predicate type $\varnothing$ is included to indicate no fact. Unlike the direct prediction of predicates, the prediction of entities requires a special handling, since there are typically millions of entities in KB, while the number of predicates is only a few hundred. In detail, based on the given output embedding $\mathbf{h}_d \in \mathbb{R}^d$, we first compute the predicted logit values of the entities:

$$\begin{aligned} \mathbf{logit}^h &= \mathbf{V_e}\mathrm{relu}(\mathbf{W_1}\mathbf{h}_\mathrm{d}) \\ \mathbf{logit}^t &= \mathbf{V_e}\mathrm{relu}(\mathbf{W_2}\mathbf{h}_\mathrm{d}) \end{aligned} \tag{6}$$

where $\mathbf{W_1}, \mathbf{W_2} \in \mathbb{R}^{d \times d}$ are learnable parameters, and $\mathbf{V_e}$ is entity embedding matrix mentioned in Section 2.1. Then, we conduct masked softmax[1] to compute the distribution of the entities:

$$\begin{aligned} \mathbf{p}^h &= \mathrm{masked\_softmax}(\mathbf{logit}^\mathrm{h}, \mathrm{C(X)}) \\ \mathbf{p}^t &= \mathrm{masked\_softmax}(\mathbf{logit}^\mathrm{t}, \mathrm{C(X)}) \end{aligned} \tag{7}$$

where $C(X)$ is the entity candidates of the given sentence X and is obtained through the process mention in the following paragraph.

Candidate Selection. Inspired by the studies in entity linking (Ganea and Hofmann, 2017; Kolitsas et al., 2018), we conduct the candidate selection to avoid involving an extremely large number of entities. For each span s in the given sentence X, we select up to 10 entity candidates that might be referred by this span. These top entities are based on an empirical probabilistic entity-map $p(e|s)$ built from hyperlinks and disambiguation pages in Wikipedia. We denote this candidate set as $C(X)$ and use it at both training and test time. For more details about the candidate selection, we refer readers to Kolitsas et al. (2018).

2.4 Bipartite Matching Loss

The main difficulty of training is to score the predicted facts with respect to the ground truths in an end-to-end manner. We solve this difficulty by introducing a permutation-invariant loss function, called bipartite matching loss. The procedure of computing this loss can be divided into two steps: finding the optimal matching and computing the loss based on the optimal matching.

Notations. Let us denote $\mathbf{Y} = \{\mathbf{Y}_i\}_{i=1}^n$ as the set of ground truth facts, and $\hat{\mathbf{Y}} = \{\hat{\mathbf{Y}}_i\}_{i=1}^m$ as the set of m predicted facts, where m is greater than or equal to n. We can consider $\mathbf{Y}$ also as a set of size m padded with $\varnothing$ (no fact). Each element i of the ground truth set can be seen as a $\mathbf{Y}_i = (h_i, r_i, t_i)$,

[1] The masked softmax is defined as:

$$\mathrm{masked\_softmax(x, C)} = \begin{cases} \frac{\exp(\mathrm{x_i})}{\sum_{j \in C} \exp(\mathrm{x_j})} & i \in C \\ 0 & i \notin C \end{cases}$$

where h_i, r_i and t_i are the target head entity, predicate (which may be $\varnothing$) and tail entity, respectively. Each element i of the set of predicted facts is denoted as $\hat{\mathbf{Y}}_i = (\mathbf{p}_i^h, \mathbf{p}_i^r, \mathbf{p}_i^t)$, which is calculated based on Equation 5 and Equation 7.

Finding the Optimal Matching. The first step in bipartite matching loss is to find the optimal matching between the set of ground truth facts $\mathbf{Y}$ and the set of predicted facts $\hat{\mathbf{Y}}$, which can be reduced to a linear balanced assignment problem[2] (Burkard et al., 2012). In detail, we can regard the set of predicted facts $\hat{\mathbf{Y}}$ as a set of persons, and the set of ground truth $\mathbf{Y}$ as a set of jobs. For each ground truth fact, only one predicted fact is assigned to it, and vice versa. Meanwhile, the cost of assigning $\hat{\mathbf{Y}}_i$ (the persons i) with $\mathbf{Y}_j$ (the job j) is defined as:

$$\begin{aligned}\mathcal{C}_{match}(\hat{\mathbf{Y}}_i, \mathbf{Y}_j) = &- \mathbb{1}_{\{r_i \neq \varnothing\}}[\mathbf{p}_i^r(r_j) \\ &+ \mathbf{p}_i^h(h_j) + \mathbf{p}_i^t(t_j)]\end{aligned} \tag{8}$$

The goal of this problem is to find a permutation of elements $\pi^\star$ with the lowest cost, which is defined as:

$$\pi^\star = \arg\min_{\pi \in \Pi(m)} \sum_{i=1}^{m} \mathcal{C}_{match}(\hat{\mathbf{Y}}_{\pi(i)}, \mathbf{Y}_i) \tag{9}$$

where $\Pi(m)$ is the space of all m-length permutations, and $\mathcal{C}_{match}(\hat{\mathbf{Y}}_{\pi(i)}, \mathbf{Y}_i)$ is the cost between the predicted fact with index $\pi(i)$ and the ground truth $\mathbf{Y}_i$.

One of the most effective ways to solve the assignment problem is Hungarian algorithm[3] (Kuhn, 1955). Armed with this algorithm, the optimal assignment $\pi^\star$ with the minimum total cost can be easily computed in polynomial time ($O(m^3)$).

Computing the Loss. The second step is to compute the loss for all pairs matched in the previous step. We define the loss as:

$$\begin{aligned}\mathcal{L}(\hat{\mathbf{Y}}, \mathbf{Y}) = \sum_{i=1}^{m} \{&- \log \mathbf{p}^r_{\pi^\star(i)}(r_i) + \mathbb{1}_{\{r_i \neq \varnothing\}}[\\ &- \log \mathbf{p}^h_{\pi^\star(i)}(h_i) - \log \mathbf{p}^t_{\pi^\star(i)}(t_i)]\}\end{aligned} \tag{10}$$

where $\pi^\star$ is the optimal assignment computed by Hungarian algorithm in the first step.

3 Experiments

In this section, we carry out an extensive set of experiments with the aim of answering the following research questions (**RQ**s):

- **RQ1**: How well do our proposed set generation networks (SGN) perform, in comparison with the competitive baselines?
- **RQ2**: How efficient is the training and inference of the model?
- **RQ3**: How does each design of the proposed networks matter?
- **RQ4**: What is the performance of the proposed networks in sentences that mention different numbers of facts?

In the remainder of this section, we describe the datasets, experimental settings (in the Appendix), and all baselines.

3.1 Datasets and Evaluation Metrics

The Cold Start track in TAC (Getman et al., 2018) provides a testbed for KBP systems. However, the dataset is not publicly available and manual evaluation is used to examine a system's "justification" (Mesquita et al., 2019), which make it difficult to reproduce TAC's evaluation for new systems.

Instead, we validate the proposed method on two publicly available datasets: WIKI and GEO[4] (Trisedya et al., 2019). The statistics of these datasets are shown in Table 2. The training set, validation set and WIKI are constructed from Wikipedia articles. To evaluate methods on a different style of text than the training data, GEO is used as a testbed, which is a dataset about user reviews on 100 popular landmarks in Australia.

Instead of performing the irreproducible manual evaluation, standard precision, recall and micro-F1 are adopted to evaluate the model in these datasets. A fact is regarded as correct if the predicate and the two corresponding entities are all correct.

	#Sentences	#Facts	#Entities	#Predicates
Training	224881	300198	248244	157
Validation	988	1329	1683	37
WIKI	29785	39894	38690	109
GEO	1000	1000	124	11

Table 2: Statistics of the dataset.

3.2 Implementation Details

We tune the hyperparameters of our proposed method by grid searching using the validation

[2] https://en.wikipedia.org/wiki/Assignment_problem

[3] https://en.wikipedia.org/wiki/Hungarian_algorithm

[4] These datasets are available at https://bitbucket.org/bayudt/relation_extraction

Model		WIKI			GEO		
		Precision	Recall	F1	Precision	Recall	F1
Pipeline Models	AIDA + MinIE	36.72	48.56	41.82	35.74	39.01	37.30
	NeuralEL + MinIE	35.11	39.67	37.25	36.44	38.11	37.26
	AIDA + ClauseIE	36.17	47.28	40.99	35.31	39.51	37.29
	NerualEL + ClauseIE	34.45	37.86	36.07	35.63	37.91	36.73
	AIDA + CNN	40.35	35.03	37.50	37.15	31.65	34.18
	NeuralEL + CNN	36.89	35.21	36.03	37.81	30.05	33.49
End-to-End Models	Single Attention	60.56	52.31	56.13	58.69	48.51	53.12
	N-gram Attention	74.24	68.45	71.23	68.16	68.61	68.38
	Transformer	58.29	50.25	53.97	61.81	61.61	61.71
	Set Generation Networks	**82.75** ± 0.11	**77.55** ±0.27	**80.07** ±0.27	**86.89** ±0.51	**85.31**±0.47	**86.10**±0.34

Table 3: Precision (%) , Recall (%) and F1 score (%) of our proposed set generation networks and current SoTA methods on the WIKI and GEO dataset. The results of baselines are taken from Trisedya et al. (2019). To reduce randomness, we use 10 different random seeds to run the model with early stopping on the development set, and report the mean and standard deviation (%) of test precision, recall and F1 for each dataset.

set. For a fair comparison, the dimension of pre-trained word, entity and predicate embeddings is set to 64, which is the same as Trisedya et al. (2019). The initial learning rate is set to 0.0001, the number of stacked transformer blocks in the non-autoregressive decoder is set to 2 and the batch size is set to 8. We use the dropout strategy to mitigate overfitting, and the dropout rate is set to 0.1. All experiments are conducted with an NVIDIA GeForce RTX 2080 Ti.

3.3 Baselines

We compare the proposed model with the following systems that report SoTA results on these datasets.

Firstly, we compare our proposed model with pipeline models. In these pipeline models, we use two entity discovery and entity linking systems, **AIDA** (Hoffart et al., 2011; Yosef et al., 2011) and **NeuralEL** (Kolitsas et al., 2018). In AIDA, entity mentions are automatically detected by using the Stanford NER Tagger (Manning et al., 2014), and then are mapped to entities by using a probabilistic graphical model. In NeuralEL, all possible spans that have at least one possible entity candidate are generated, and are linked to entities by using a context-aware compatibility score. To label the relationship between two entities, we adopt supervised approaches like **CNN** (Lin et al., 2016) and OpenIE-based approaches, such as **MinIE** (Gashteovski et al., 2017) and **ClausIE** (Del Corro and Gemulla, 2013). In OpenIE-based approaches, we leverage the dictionary based paraphrase detection to map the extracted predicate of the output. We combine three paraphrase dictionaries including PATTY (Nakashole et al.), POLY (Grycner and Weikum, 2016), and PPDB (Ganitkevitch et al., 2013). Following previous work (Trisedya et al., 2019), we replace the extracted predicate with the correct predicate ID if one of the paraphrases of the correct predicate appears in the extracted predicate. Otherwise, we replace the extracted predicate with "NA" to indicate an unrecognized predicate.

Secondly, we compare our proposed model with end-to-end models, including **Single Attention** model (Bahdanau et al., 2015), **Transformer** model (Vaswani et al., 2017) and **N-gram Attention** model (Trisedya et al., 2019). Compared with the single attention model, the N-gram attention model computes the n-gram combination of attention weight to capture the verbal or noun phrase context. Note that all of these end-to-end models are based on the encoder-decoder framework and are required to sort the ground truth facts. Following previous work (Trisedya et al., 2019), we build the ground truth sequence according to the inherent order in these datasets.

3.4 Main Results

To start, we address the research question **RQ1**. Table 3 shows the results of our proposed model against baselines on two benchmark datasets.

Taken overall, our proposed model substantially outperforms baselines on these datasets. In WIKI, our proposed model achieves 8.51%, 9.10% and 8.84% improvements in Precision, Recall and F1 score respectively over the current SoTA method, N-gram Attention. In GEO, our proposed model achieves the SoTA results and there is 18.73%, 16.70% and 17.72% improvement in Precision, Recall and F1 score compared with the best existing

model. Such significant improvements demonstrate the effectiveness of our proposed method.

Meanwhile, we observe that pipeline models struggle to achieve satisfactory results in KBP. To further show the effect of error propagation, we first examine the performance of the entity discovery and entity linking module. Through experiments, AIDA can only get 43.02% (WIKI) and 54.75% (GEO) F1 score, and NeuralEL achieves 45.92% (WIKI) and 67.62% (GEO) in F1 score. Then, we remove the entity disambiguation preprocessing step by allowing the CNN model to access golden entities. In this setup, CNN achieves 81.92% and 75.82% in F1 score over the WIKI and GEO datasets, respectively. The poor experimental results indicate that mistakes made by entity discovery and linking modules are propagated to the final output of the system, negatively affecting the overall performance of pipeline models.

3.5 Analysis on Speed

Model	Training Time	Test Time
Ours	**689.82**	**49.97**
Transformer	793.84 ($\times$1.15)	157.86 ($\times$3.16)
Single Attention	941.58 ($\times$1.36)	100.17 ($\times$2.00)

Table 4: The performance of models in training and testing time. The time duration is measured in seconds.

Next, we examine the speed of models to answer the research question **RQ2**. As a basic NLP tool, a high speed of both training and inference is required. For a fair comparison, Transformer, Single Attention, and our proposed model are implemented under the same experimental conditions. We randomly select 10 training and testing epochs as samples. The average time of training and testing is shown in Table 4. From the table, we find that our proposed model is more efficient than Single Attention and Transformer in both training and inference. The reason behind that is Single Attention and Transformer are all based on autoregressive decoders, which generate each predicted element conditioned on the sequence previously generated. This process is not parallelizable. However, our proposed model leverages a non-autoregressive decoder, which does not have the constraint of an autoregressive factorization and can generate all elements in one shot. With such a parallelizable decoder, our proposed model is very fast in both training and inference.

Settings	Precision	Recall	F1	Δ
Bipartite Matching Loss, Number of Decoder Layer=2	83.00	77.31	80.06	-
Bipartite Matching Loss, Random Embeddings, Number of Decoder Layer = 2	80.34	71.03	75.40	$\downarrow$ 4.66
Bipartitie Matching Loss, Number of Decoder Layer=3	82.76	77.83	80.22	$\uparrow$ 0.16
Bipartite Matching Loss, Number of Decoder Layer=1	81.42	75.41	78.30	$\downarrow$ 1.76
Cross-Entropy Loss, Fixed Order, Number of Decoder Layer=2	78.66	69.15	73.60	$\downarrow$ 6.49
Cross-Entropy Loss, Random Order, Number of Decoder Layer=2	77.54	66.62	71.66	$\downarrow$ 8.40

Table 5: Results of ablation studies on WIKI dataset.

3.6 Ablation Studies

In this section, we turn to the research question **RQ3**. We conduct various ablation studies to investigate the effectiveness of the pretrained embeddings, the non-autoregressive decoder and the bipartite matching loss.

First, instead of using pretrained embeddings, we randomly initialize all embeddings. From Table 5, we can observe that there is a significant performance drop ($\downarrow$ 4.66%) by using randomly initialized embeddings. Next, we examine the effectiveness of the non-autoregressive decoder. From Table 5, we find that increasing the number of layers of the non-autoregressive decoder can achieve better results. When the number of decoder layers is set to 1, 2, and 3, the best results are 78.30%, 80.06% and 80.22%, respectively. We conjecture this is largely due to that with the deepening of the decoder layers, more multi-head self-attention modules allow for better modeling of relationships between fact queries, and more multi-head inter attention modules allow for more complete integration of sentence information into fact queries. Finally, we compare bipartite matching loss with widely used cross-entropy loss. In cross-entropy loss, we adopt two strategies to sort golden facts in training: **Fix Order** and **Random Order**. Fix Order means we randomly select one valid order before training and keep the order unchanged during training. Random Order means we randomly sort golden facts for each sentence in every training epoch. From the results, we find that: (1) Compared with the Fix Order strategy, simply shuffling (Random Order) will not improve the performance. (2) Compared with Fix Order and Random Order,

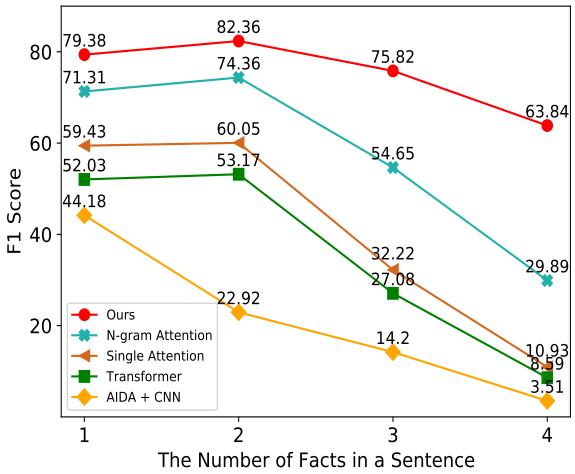

Figure 2: Detailed results on sentences that contain a different number of facts. We divide the sentences of the WIKI test set into 4 subclasses. Each class contains sentences that mention 1,2,3,or 4 facts.

introducing bipartite matching loss gains 6.49% and 8.40% improvements in F1 score, which verifies the effectiveness of bipartite matching loss.

3.7 Detailed Results on Sentences with Different Number of Facts

Finally, we answer the research question **RQ4**. We compare the models' ability on sentences that mention a different number of facts. We divide the sentences in the WIKI test set into 4 subclasses. Each class contains sentences that mention 1,2,3,or 4 facts. The results are shown in Figure 2. From the results, we can observe that: (1) Compared with the other models, our proposed model achieves the highest performance in all cases. Such results demonstrate the ability of our proposed model in handling multiple facts. (2) When extracting facts from sentences that mention 1 fact or 2 facts, most models can achieve the best performance. However, when the number of facts increases, the performance of models decreases significantly.

4 Related Work

Knowledge Base Population. Traditionally, KBP has been tackled with pipeline models (Shin et al., 2015; Angeli et al., 2015; Zhang et al., 2017; Chaganty et al., 2017; Mesquita et al., 2019).The main shortcoming of pipeline systems is error propagation. End-to-end systems (Liu et al., 2018; Trisedya et al., 2019) are a promising solution for addressing error propagation. These methods are all based on the seq2seq framework. However, a roadblock for the advancement of this line of research is that an inexistent order of facts must be introduced to train the seq2seq model. In this paper, we introduce set generation networks to overcome this roadblock.

Non-Autoregressive Model for Generation. Gu et al. (2018) began to explore non-autoregressive model, the aim of which is to generate sequences in a parallel manner. Since then, there is rich literature devoted to this topic, such as Lee et al. (2018); Ma et al. (2019); Ren et al. (2020); Ran et al. (2020); Kong et al. (2020). Nowadays, non-autoregressive models are widely explored in natural language and speech processing tasks such as neural machine translation (Lee et al., 2018; Ma et al., 2019) and automatic speech recognition (Chen et al., 2019; Tian et al., 2020; Bai et al., 2020). To the best of our knowledge, this is the first work to apply the non-autoregressive model to knowledge base population. In this work, we resort to the non-autoregressive model to generate the set of relational facts in one shot.

Set Prediction. The problem with predicting sets is that the output order of the elements is arbitrary, so computing an element-wise loss does not make sense; there is no guarantee that the elements in the target set happen to be in the same order as they were generated. Assignment-based losses are a popular choice on point clouds (Fan et al., 2017; Yang et al., 2018) and object detection (Carion et al., 2020; Zhu et al., 2020; Yao et al., 2021). An alternative approach is to perform the set generation sequentially (Stewart et al., 2016; You et al., 2018). Furthermore, Zhang et al. (2019) develop a FSPool-based set prediction method. In this paper, we formulate the end-to-end knowledge base population task as a set generation problem

5 Conclusion

In this paper, we introduce set generation networks for end-to-end KBP. Compared with previous seq2seq models, we formulate the KBP task as a set generation problem. In such a way, the model will be relieved of predicting the order between multiple facts. To solve the set generation problem, We combine non-autoregressive parallel decoding with the bipartite matching loss function. To validate the effectiveness of the proposed networks, we conduct extensive experiments. Experimental results show our proposed networks outperform current SoTA baselines over different scenarios.

Acknowledgments

We thank the anonymous reviewers for their constructive comments. This work is supported by the National Key Research and Development Program of China (No.2020AAA0106400), the National Natural Science Foundation of China (No.61922085 and No.61976211), the CCF-Tencent Open Research Fund, the Beijing Academy of Artificial Intelligence (No. BAAI2019QN0301), the Key Research Program of the Chinese Academy of Sciences (No. ZDBS-SSW-JSC006) and the Youth Innovation Promotion Association CAS.

References

Gabor Angeli, Victor Zhong, Danqi Chen, Arun Chaganty, Jason Bolton, Melvin Johnson Premkumar, Panupong Pasupat, Sonal Gupta, and Christopher D Manning. 2015. Bootstrapped self training for knowledge base population. In *TAC*.

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

Ye Bai, Jiangyan Yi, Jianhua Tao, Zhengkun Tian, Zhengqi Wen, and Shuai Zhang. 2020. Listen attentively, and spell once: Whole sentence generation via a non-autoregressive architecture for low-latency speech recognition. In *arXiv*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NeurIPS*.

Rainer Burkard, Mauro Dell'Amico, and Silvano Martello. 2012. *Assignment problems: revised reprint*. SIAM.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-end object detection with transformers. In *European Conference on Computer Vision*.

Arun Chaganty, Ashwin Paranjape, Percy Liang, and Christopher D. Manning. 2017. Importance sampling for unbiased on-demand evaluation of knowledge base population. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Nanxin Chen, Shinji Watanabe, Jesús Villalba, and Najim Dehak. 2019. Non-autoregressive transformer automatic speech recognition. In *arXiv*.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *EMNLP*.

Wanyun Cui, Yanghua Xiao, Haixun Wang, Yangqiu Song, Seung-won Hwang, and Wei Wang. 2017. Kbqa: learning question answering over qa corpora and knowledge bases. In *VLDB*.

Luciano Del Corro and Rainer Gemulla. 2013. Clausie: clause-based open information extraction. In *WWW*.

Jack Edmonds and Richard M Karp. 1972. Theoretical improvements in algorithmic efficiency for network flow problems. In *JACM*.

Haoqiang Fan, Hao Su, and Leonidas J Guibas. 2017. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep joint entity disambiguation with local neural attention. In *EMNLP*.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. Ppdb: The paraphrase database. In *NAACL*.

Kiril Gashteovski, Rainer Gemulla, and Luciano del Corro. 2017. Minie: minimizing facts in open information extraction. In *EMNLP*.

Jeremy Getman, Joe Ellis, Stephanie Strassel, Zhiyi Song, and Jennifer Tracey. 2018. Laying the groundwork for knowledge base population: Nine years of linguistic resources for TAC KBP. In *LREC*.

Adam Grycner and Gerhard Weikum. 2016. Poly: Mining relational paraphrases from multilingual sentences. In *EMNLP*.

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

Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A survey on knowledge graph-based recommender systems. In *IEEE Transactions on Knowledge Data Engineering*.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *EMNLP*.

Heng Ji and Ralph Grishman. 2011. Knowledge base population: Successful approaches and challenges. In *ACL*.

Nikolaos Kolitsas, Octavian-Eugen Ganea, and Thomas Hofmann. 2018. End-to-end neural entity linking. In *CoNLL*.

Xiang Kong, Zhisong Zhang, and Eduard Hovy. 2020. Incorporating a local translation mechanism into non-autoregressive translation. In *EMNLP*.

Harold W Kuhn. 1955. The hungarian method for the assignment problem. In *Naval research logistics quarterly*.

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

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *ACL*.

Yue Liu, Tongtao Zhang, Zhicheng Liang, Heng Ji, and Deborah L McGuinness. 2018. Seq2rdf: An end-to-end application for deriving triples from natural language text. In *CEUR Workshop Proceedings*.

Xuezhe Ma, Chunting Zhou, Xian Li, Graham Neubig, and Eduard Hovy. 2019. FlowSeq: Non-autoregressive conditional sequence generation with generative flow. In *EMNLP*.

Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2018. Mem2seq: Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems. In *ACL*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *ACL System Demonstrations*.

Filipe Mesquita, Matteo Cannaviccio, Jordan Schmidek, Paramita Mirza, and Denilson Barbosa. 2019. KnowledgeNet: A benchmark dataset for knowledge base population. In *EMNLP-IJCNLP*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NeurIPS*.

David Milne and Ian H Witten. 2008. Learning to link with wikipedia. In *CIKM*.

James Munkres. 1957. Algorithms for the assignment and transportation problems. In *Journal of the society for industrial and applied mathematics*.

Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. Patty: a taxonomy of relational patterns with semantic types. In *EMNLP*.

Qiu Ran, Yankai Lin, Peng Li, and Jie Zhou. 2020. Learning to recover from multi-modality errors for non-autoregressive neural machine translation. In *ACL*.

Yi Ren, Jinglin Liu, Xu Tan, Zhou Zhao, Sheng Zhao, and Tie-Yan Liu. 2020. A study of non-autoregressive model for sequence generation. In *ACL*.

Jaeho Shin, Sen Wu, Feiran Wang, Christopher De Sa, Ce Zhang, and Christopher Ré. 2015. Incremental knowledge base construction using deepdive. In *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases*.

Russell Stewart, Mykhaylo Andriluka, and Andrew Y Ng. 2016. End-to-end people detection in crowded scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NeurIPS*.

Zhengkun Tian, Jiangyan Yi, Jianhua Tao, Ye Bai, Shuai Zhang, and Zhengqi Wen. 2020. Spike-triggered non-autoregressive transformer for end-to-end speech recognition. In *arXiv*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *HLT-NAACL*.

Bayu Distiawan Trisedya, Gerhard Weikum, Jianzhong Qi, and Rui Zhang. 2019. Neural relation extraction for knowledge base enrichment. In *ACL*.

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 *NeurIPS*.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. In *CoNLL*.

Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. 2018. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Zhuyu Yao, Jiangbo Ai, Boxun Li, and Chi Zhang. 2021. Efficient detr: Improving end-to-end object detector with dense prior. *arXiv preprint arXiv:2104.01318*.

Mohamed Amir Yosef, Johannes Hoffart, Ilaria Bordino, Marc Spaniol, and Gerhard Weikum. 2011. Aida: An online tool for accurate disambiguation of named entities in text and tables. In *VLDB*.

Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. 2018. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International Conference on Machine Learning*.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. In *booktitle of machine learning research*.

Ce Zhang, Christopher Ré, Michael Cafarella, Christopher De Sa, Alex Ratner, Jaeho Shin, Feiran Wang, and Sen Wu. 2017. Deepdive: Declarative knowledge base construction. In *Communications of the ACM*.

Yan Zhang, Jonathon Hare, and Adam Prügel-Bennett. 2019. Deep set prediction networks. *arXiv preprint arXiv:1906.06565*.

Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. 2020. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*.