Neural Entity Summarization with Joint Encoding and Weak Supervision

Junyou Li[1] , **Gong Cheng**[1*] , **Qingxia Liu**[1] , **Wen Zhang**[2,3] , **Evgeny Kharlamov**[3,4] , **Kalpa Gunaratna**[5] and **Huajun Chen**[2]
[1]National Key Laboratory for Novel Software Technology, Nanjing University, China
[2]College of Computer Science and Technology, Zhejiang University, China
[3]Bosch Center for Artificial Intelligence, Robert Bosch GmbH, Germany
[4]Department of Informatics, University of Oslo, Norway
[5]Samsung Research America, Mountain View CA, USA
{141220056, qxliu2013}@smail.nju.edu.cn, gcheng@nju.edu.cn, {wenzhang2015, huajunsir}@zju.edu.cn, evgeny.kharlamov@de.bosch.com, k.gunaratna@samsung.com

Abstract

In a large-scale knowledge graph (KG), an entity is often described by a large number of triple-structured facts. Many applications require abridged versions of entity descriptions, called entity summaries. Existing solutions to entity summarization are mainly unsupervised. In this paper, we present a supervised approach NEST that is based on our novel neural model to jointly encode graph structure and text in KGs and generate high-quality diversified summaries. Since it is costly to obtain manually labeled summaries for training, our supervision is weak as we train with programmatically labeled data which may contain noise but is free of manual work. Evaluation results show that our approach significantly outperforms the state of the art on two public benchmarks.

1 Introduction

Semantic Web and knowledge graphs (KGs) provide entity descriptions in triples (i.e., node-arc-node), supporting many and various applications. Figure 1 illustrates a KG containing descriptions of entities such as Everest, which could be presented in a knowledge card in Google search. The size of an entity description can reach hundreds of triples, exceeding the capacity of a compact knowledge card and hence requiring a summarization method for generating a summary—an optimal subset of triples. This established research problem is referred to as *entity summarization* [Liu *et al.*, 2019].

Challenges. In this work we focus on computing general-purpose entity summaries. Existing methods are mostly unsupervised [Cheng *et al.*, 2011; Sydow *et al.*, 2013; Thalhammer *et al.*, 2016; Gunaratna *et al.*, 2015]. They use various heuristics for ranking and selecting triples. We attempt to develop a supervised approach where we face two challenges. (1) KG is a hybrid of graph structure and short text. Extracting and combining useful features from them to represent a set of triples is a non-trivial task. (2) It is expensive to manually label a large set of general-purpose entity summaries. The availability of training data for supervision is a bottleneck in this research.

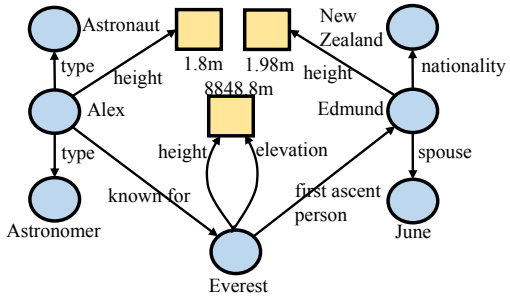

Figure 1: An example KG.

Contributions. To address these challenges, we develop NEST,[1] the abbreviation for Neural Entity SummarizaTion, where our contributions are summarized as follows. (1) Our neural encoder jointly extracts structural and textual features from KGs. Manual engineering is obviated. (2) We fine-tune with weak supervision from programmatically labeled data. Manual labeling is not needed.

Approach. The full model of NEST is shown in Figure 2. It relies on the assumption that a high-quality general-purpose summary should satisfy two natural conditions. (1) Each selected triple should describe an important feature of the entity. (2) The selected triples should diversely span different aspects of the entity. Accordingly, NEST incorporates two neural scorers: a salience-based triple scorer (STS) and a diversity-based summary scorer (DSS). Input KGs are fed into a neural encoder, which is firstly pre-trained in an unsupervised way to generate contextualized representations of elements, and then is linked to STS and DSS to be fine-tuned in a supervised way.

*Contact Author

[1]https://github.com/nju-websoft/NEST

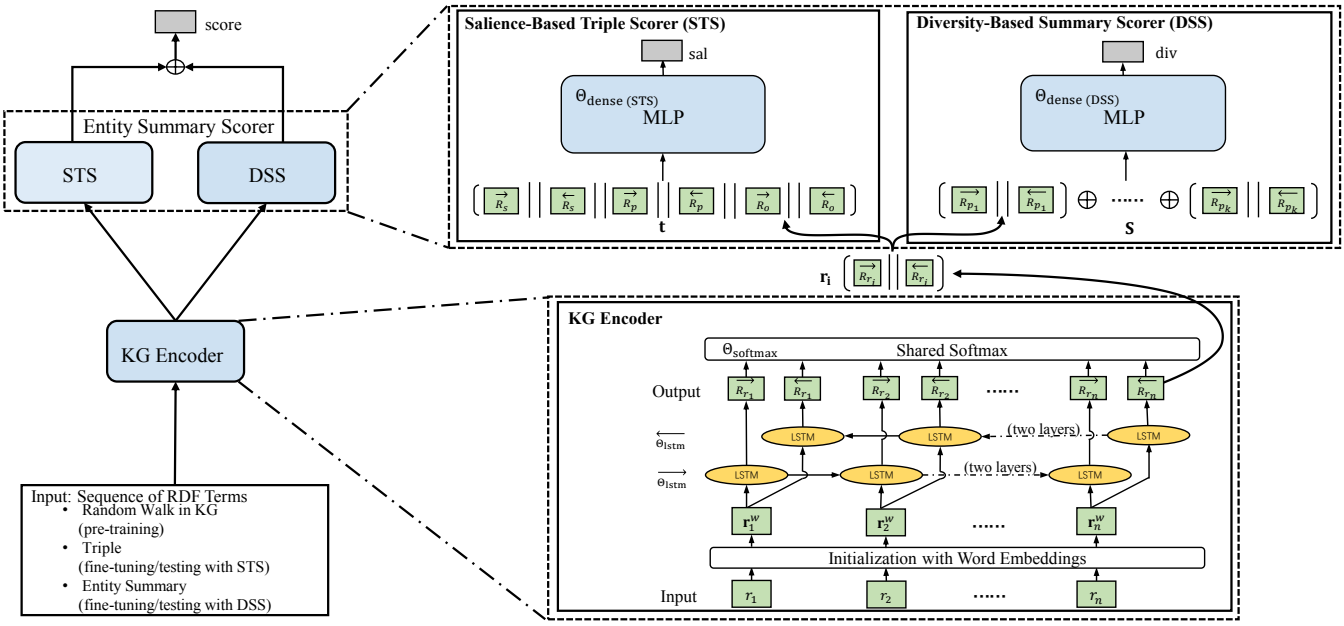

Figure 2: Proposed neural model for entity summarization.

Outline. The remainder of the paper is organized as follows. We formulate the problem in Section 2 and present our approach in Section 3. Experiments are described in Section 4, and the results are given in Section 5. Related work is discussed in Section 6 before we conclude in Section 7.

2 Problem Statement

The Resource Description Framework (RDF) is a popular format for interchanging KGs. Our formulation uses RDF, but our approach can be applied to KGs in other formats.

RDF. Let $\mathbb{I}, \mathbb{B}, \mathbb{L}$ be the disjoint sets of all Internationalized Resource Identifiers (IRIs), blank nodes, and literals in RDF, respectively. They are collectively known as RDF terms. An RDF term r has a textual form $\texttt{text}(r)$, e.g., a human-readable label of an IRI or blank node, or the lexical form of a literal. RDF data is a set of subject-predicate-object triples:

$$T \subseteq (\mathbb{I} \cup \mathbb{B}) \times \mathbb{I} \times (\mathbb{I} \cup \mathbb{B} \cup \mathbb{L}) \,. \tag{1}$$

KG can be straightforwardly formatted in RDF. Each triple $\langle s, p, o \rangle \in T$ represents an arc from subject s to object o labeled with predicate p. Figure 1 illustrates a KG where for RDF terms we show their textual forms.

Entity description. An entity is identified by an IRI or a blank node. The description of entity e is a set of triples:

$$\texttt{D}(e) = \{\langle e, p, o \rangle \in T\} \cup \{\langle s, p, e \rangle \in T\} \,. \tag{2}$$

Each triple $t \in \texttt{D}(e)$ represents a property-value pair of e:

$$\langle \texttt{p}(t), \texttt{v}(t) \rangle = \begin{cases} \langle p, o \rangle & \langle e, p, o \rangle \in \texttt{D}(e) \,, \\ \langle p^{-1}, s \rangle & \langle s, p, e \rangle \in \texttt{D}(e) \,, \end{cases} \tag{3}$$

where p^{-1} represents the inverse of p. In Figure 1, the description of `Everest` contains four triples (i.e., incident arcs).

Entity summarization. Given an integer size constraint k, a summary of entity e is a subset of triples $S \subseteq \texttt{D}(e)$ such that $|S| \leq k$. Entity summarization is to find an optimal summary for an entity. Optimality depends on the application. We study the generation of general-purpose summaries.

3 Approach

NEST is fed with a sequence of RDF terms as input, which represents a random walk, a triple, or an entity summary in a KG, depending on the phase of our approach. The sequence is processed by a KG encoder to generate a vector representation for each RDF term in the sequence (Section 3.1). Based on the representations, an entity summary scorer assesses the quality of a summary from two perspectives: STS for salience and DSS for diversity (Section 3.2). In the training phase, we pre-train the KG encoder in an unsupervised way and then fine-tune the entire model in a supervised way. In the test phase, we search for an optimal summary where we use the trained model to evaluate candidate summaries (Section 3.3).

3.1 KG Encoder

We convert KGs into a low dimensional space to facilitate downstream scorers. A KG is a graph labeled with textual forms of RDF terms. Our novel neural encoder jointly extracts structural and textual features from KGs.

Model. The input is a sequence of RDF terms $r_1 r_2 \cdots r_n$. We extract the textual features of each r_i in a standard way, by averaging the word embeddings of its textual form $\texttt{text}(r_i)$:

$$\mathbf{r}_i^w = \frac{1}{|\texttt{text}(r_i)|} \sum_{w_j \in \texttt{text}(r_i)} \mathbf{w}_j \,, \tag{4}$$

where $\mathbf{w}_j$ is the fixed embedding of word w_j, and $\mathbf{r}_i^w$ is the initial representation of r_i using textual features. For $\mathbf{w}_j$,

we use pre-trained fastText embeddings [Bojanowski *et al.*, 2017]. Further, to extract structural features of the sequence, we feed text-featured representations $\mathbf{r}_1^w \mathbf{r}_2^w \cdots \mathbf{r}_n^w$ into a forward model and a backward model, both using two layers of Long Short-Term Memory networks (LSTMs). They output

$$\begin{aligned} \overrightarrow{R_{r_1}}\overrightarrow{R_{r_2}}\cdots\overrightarrow{R_{r_n}} &= \overrightarrow{\text{LSTM}}(\mathbf{r}_1^w \mathbf{r}_2^w \cdots \mathbf{r}_n^w; \overrightarrow{\Theta_{\text{lstm}}}) \,, \\ \overleftarrow{R_{r_1}}\overleftarrow{R_{r_2}}\cdots\overleftarrow{R_{r_n}} &= \overleftarrow{\text{LSTM}}(\mathbf{r}_1^w \mathbf{r}_2^w \cdots \mathbf{r}_n^w; \overleftarrow{\Theta_{\text{lstm}}}) \,, \end{aligned} \tag{5}$$

where $\overrightarrow{\Theta_{\text{lstm}}}$ and $\overleftarrow{\Theta_{\text{lstm}}}$ are LSTM parameters. Each $\overrightarrow{R_{r_i}}$ (resp. $\overleftarrow{R_{r_i}}$) is the forward (resp. backward) representation of r_i contextualized by $r_1 r_2 \cdots r_{i-1}$ (resp. $r_n r_{n-1} \cdots r_{i+1}$). The output representation of r_i concatenates $\overrightarrow{R_{r_i}}$ and $\overleftarrow{R_{r_i}}$:

$$\mathbf{r_i} = [\, \overrightarrow{R_{r_i}} \parallel \overleftarrow{R_{r_i}} \,] \,. \tag{6}$$

Pre-training. We pre-train the KG encoder in an unsupervised way. We follow a common practice to sample random walks from a KG to capture its graph structure [Cai *et al.*, 2018]. For each node r, we generate a set of undirected random walks of fixed length d starting from r. Such a walk is a sequence of d RDF terms, alternating between nodes (i.e., subjects or objects of triples) and arcs (i.e., predicates). Walks are constrained to not pass through any literals; that is, a walk may start or end at a literal but cannot have literals as intermediate nodes, to prohibit meaningless connections via polysemous literals, e.g., numbers. In Figure 1, given $d = 5$, an example random walk is:

`Everest known-for Alex height 1.8m .`

For a sampled random walk $r_1 r_2 \cdots r_d$, the objective of pre-training is to maximize the existence probability of the walk. The forward and backward models compute this probability in opposite directions:

$$\begin{aligned} P(r_1 r_2 \cdots r_d) &= \prod_{i=1}^{d} P(r_i | r_1 r_2 \cdots r_{i-1}) \,, \\ P(r_1 r_2 \cdots r_d) &= \prod_{i=1}^{d} P(r_i | r_d r_{d-1} \cdots r_{i+1}) \,. \end{aligned} \tag{7}$$

Specifically, $\overrightarrow{R_{r_1}}\overrightarrow{R_{r_2}}\cdots\overrightarrow{R_{r_{i-1}}}$ in the forward model are used to predict r_i with a softmax layer, and $\overleftarrow{R_{r_d}}\overleftarrow{R_{r_{d-1}}}\cdots\overleftarrow{R_{r_{i+1}}}$ predict r_i in the backward model. We jointly maximize the log likelihood of the two directions:

$$\begin{aligned} \sum_{i=1}^{d} (&\log P(r_i | r_1 r_2 \cdots r_{i-1}; \overrightarrow{\Theta_{\text{lstm}}}, \Theta_{\text{softmax}}) \\ &+ \log P(r_i | r_d r_{d-1} \cdots r_{i+1}; \overleftarrow{\Theta_{\text{lstm}}}, \Theta_{\text{softmax}})) \,, \end{aligned} \tag{8}$$

where Θ_{softmax} are tied parameters for the softmax layer.

3.2 Entity Summary Scorer

To assess the quality of an entity summary, our novel neural scorer considers two perspectives: STS for the salience of a triple, and DSS for the diversity of a summary. STS and DSS work with the same KG encoder and share its parameters, but they are fine-tuned in succession with different training data.

STS—model. STS measures the salience of a triple $t = \langle s, p, o \rangle$. We feed t as a sequence of 3 RDF terms spo into the KG encoder, and concatenate the output representations of these RDF terms to represent t:

$$\mathbf{t} = [\, \overrightarrow{R_s} \parallel \overleftarrow{R_s} \parallel \overrightarrow{R_p} \parallel \overleftarrow{R_p} \parallel \overrightarrow{R_o} \parallel \overleftarrow{R_o} \,] \,. \tag{9}$$

Based on $\mathbf{t}$, we use a two-layer dense neural network (MLP) to compute a score representing the salience of t:

$$\texttt{sal}(t) = \text{MLP}(\mathbf{t}; \Theta_{\text{dense (STS)}}) \,, \tag{10}$$

where $\Theta_{\text{dense (STS)}}$ are parameters for the MLP.

STS—fine-tuning. To train `sal` to compute salience, we fine-tune STS and the KG encoder with labeled entity descriptions where each triple t has a binary label Y_t indicating whether t is in an ideal summary ($Y_t = 1$) or not ($Y_t = 0$). The objective of fine-tuning is to minimize the mean squared error over a set of labeled triples T_{STS} for training:

$$\frac{1}{|T_{\text{STS}}|} \sum_{t \in T_{\text{STS}}} (\texttt{sal}(t) - Y_t)^2 \,. \tag{11}$$

DSS—model. DSS measures the diversity of a summary $S = \{\langle s_1, p_1, o_1 \rangle, \ldots, \langle s_k, p_k, o_k \rangle\}$ through the diversity of predicates $\{p_1, \ldots, p_k\}$. A diverse summary should not contain predicates describing similar aspects of an entity, e.g., `height` and `elevation` of `Everest` in Figure 1. We feed S as a sequence of $3k$ RDF terms $s_1 p_1 o_1 \cdots s_k p_k o_k$ into the KG encoder, and sum the output representations of the predicates to represent S:

$$\mathbf{S} = [\, \overrightarrow{R_{p_1}} \parallel \overleftarrow{R_{p_1}} \,] \oplus \cdots \oplus [\, \overrightarrow{R_{p_k}} \parallel \overleftarrow{R_{p_k}} \,] \,. \tag{12}$$

Here for the KG encoder, we reset the cell state of LSTMs for every 3 RDF terms (i.e., every triple) in the input sequence, to make $\mathbf{S}$ invariant to the order of triples in S as such kind of invariance is reasonable for a set. Based on $\mathbf{S}$, we use a two-layer dense neural network (MLP) to compute a score representing the diversity of S:

$$\texttt{div}(S) = \text{MLP}(\mathbf{S}; \Theta_{\text{dense (DSS)}}) \,, \tag{13}$$

where $\Theta_{\text{dense (DSS)}}$ are parameters for the MLP.

DSS—fine-tuning. To train `div` to compute diversity, we fine-tune DSS and the KG encoder with a heuristically computed diversity score, namely the average cosine distance between pairs of predicates in S using their representations output from the KG encoder:

$$Z_S = \frac{1}{\binom{|S|}{2}} \sum_{\substack{\langle s_i, p_i, o_i \rangle \in S \\ \langle s_j, p_j, o_j \rangle \in S}} (1 - \cos([\, \overrightarrow{R_{p_i}} \parallel \overleftarrow{R_{p_i}} \,], [\, \overrightarrow{R_{p_j}} \parallel \overleftarrow{R_{p_j}} \,])) \,. \tag{14}$$

The objective of fine-tuning is to minimize the mean squared error over a set of entity summaries $\mathbb{S}_{\text{DSS}}$ for self-training:

$$\frac{1}{|\mathbb{S}_{\text{DSS}}|} \sum_{S \in \mathbb{S}_{\text{DSS}}} (\texttt{div}(S) - Z_S)^2 \,. \tag{15}$$

Compared with directly using Z_S as $\texttt{div}(S)$, when we fine-tune DSS to approximate Z_S, we also tune the parameters of the KG encoder which is shared with STS, thereby having the potential to achieve better overall performance.

3.3 Entity Summary Generation

In the test phase, given entity description $\mathtt{D}(e)$ and each candidate summary $S \subseteq \mathtt{D}(e)$, we evaluate S using STS and DSS to generate its overall score:

$$\mathtt{score}(S) = \mathtt{div}(S) + \sum_{t \in S} \mathtt{sal}(t)\,. \tag{16}$$

For a size constraint k, there are $\binom{|\mathtt{D}(e)|}{k}$ candidate summaries. It may be computationally impracticable to evaluate all of them. We adopt the simulated annealing heuristic to search for an optimal summary. Specifically, we randomly sample an initial summary $S_0 \subseteq \mathtt{D}(e)$ with $|S_0| = k$ and compute $\mathtt{score}(S_0)$. Then in the i-th iteration, we replace a random triple in S_{i-1} with a new triple in $\mathtt{D}(e) \setminus S_{i-1}$ to form S_i and compute $\mathtt{score}(S_i)$. We will definitely continue iterations if $\mathtt{score}(S_i) > \mathtt{score}(S_{i-1})$. Otherwise, we will continue with an acceptance probability that is decreased after each iteration. Finally, S_i in the last iteration is output.

4 Experiments

We compared NEST with 11 baseline methods on two public benchmarks for general-purpose entity summarization.

4.1 Datasets and Evaluation Metrics

We evaluated with the two largest available benchmarks for general-purpose entity summarization: the Entity Summarization BenchMark (ESBM) [Liu *et al.*, 2020] and the FACES Evaluation Dataset (FED) [Gunaratna *et al.*, 2015]. ESBM v1.0[2] consists of ESBM-D and ESBM-L. ESBM-D provides 1,200 ground-truth summaries for 100 entities in DBpedia v2015-10. Each entity description is labeled with 6 summaries for $k = 5$, and 6 summaries for $k = 10$, created by different human experts. Similarly, ESBM-L provides 480 ground-truth summaries for 40 entities in LinkedMDB. In FED,[3] 50 entities in DBpedia v3.9 are labeled with 373 summaries for $k = 5$, and 373 summaries for $k = 10$.

Following ESBM, we compared a computed summary with a ground-truth summary and calculated F1 score. For each method to evaluate, we calculated its mean F1 over all the entities and ground-truth summaries in each dataset.

4.2 Programmatic Generation of Labeled Data

In the absence of a large set of manually labeled entity descriptions for training, we programmatically generated labeled data from the correspondence between DBpedia and Wikipedia. Such labels may contain noise and hence our supervision was weak but free of manual work. Methods trained with this data were then tested on ESBM and FED where entities were from not only DBpedia but also LinkedMDB, thereby also evaluating the generalizability of a method.

Specifically, for each entity e in DBpedia we found its corresponding article in Wikipedia, which typically began with an untitled section $\mathtt{A}(e)$ containing key factual information about e. We labeled triple $t \in \mathtt{D}(e)$ as positive, i.e., $Y_t = 1$ in Eq. (11), if t was *mentioned* in $\mathtt{A}(e)$, otherwise negative and $Y_t = 0$. We programmatically determined whether t was mentioned in $\mathtt{A}(e)$ as follows. We transformed the textual form of the property value, i.e., $\mathtt{text}(\mathtt{v}(t))$, into a normalized word sequence W_t in four steps: splitting the text using non-letters and non-digits as delimiters, converting each month word (e.g., February) into two digits (e.g., 02), lowercasing all the letters, and removing stop words. In the same way we transformed $\mathtt{A}(e)$ into $W_{\mathtt{A}(e)}$. A mention of t in $\mathtt{A}(e)$ was recognized if every word in W_t was a number appearing in $W_{\mathtt{A}(e)}$, or if more than half of the words in W_t appeared or had an approximate match in $W_{\mathtt{A}(e)}$. We recognized an approximate match between word $w_i \in W_t$ and word $w_j \in W_{\mathtt{A}(e)}$ if w_i consisted of more than γ characters and the edit distance between w_i and w_j was smaller than δ. We empirically set $\gamma = 7$ and $\delta = 3$ in the experiments.

		Gold standard	
		Positive	Negative
Generated label	Positive	959	92
	Negative	73	4,901

Table 1: Error matrix for programmatically labeled data.

To evaluate the quality of the generated labels, we randomly sampled 200 entities from DBpedia and manually labeled 6,025 triples in their descriptions as the gold standard. The error matrix is shown in Table 1. The true positive rate is 92.93%, and the true negative rate is 98.16%, demonstrating the good quality of our labeled data.

For the subsequent evaluation, from the programmatically labeled data we randomly sampled 11,672 entity descriptions containing a total of 264,998 triples. The sampling was controlled to be disjoint with the test entities in ESBM and FED. We used 80% of the labeled data for training methods and 20% for validation, e.g., tuning hyperparameters.

4.3 Configuration of NEST

For each KG we pre-trained and fine-tuned a separate model.

Pre-training KG encoder. Starting from each node in a KG we sampled 100 random walks with $d = 7$. Our LSTMs contained 4,096 units, 300 dimension projections, and a residual connection from the first layer to the second layer. We trained for 10 epochs with batch size 256.

Fine-tuning STS. From each labeled entity description $\mathtt{D}(e)$, we randomly sampled 5 triples as a batch and we sampled $\frac{|\mathtt{D}(e)|}{2}$ times. Our MLP had 4,096 units in each layer and applied ReLU activations. We trained for 10 epochs.

Fine-tuning DSS. From the labeled triples we randomly sampled 100,000 subsets of size 5 as summaries. Our MLP had 4,096 units in each layer and applied ReLU activations. We trained for 10 epochs with batch size 5.

Generating summary. In simulated annealing, we initialized the acceptance probability to 0.1, and decreased it by $\frac{0.1}{|\mathtt{D}(e)|}$ after each iteration.

[2] https://w3id.org/esbm/

[3] http://wiki.knoesis.org/index.php/FACES

	ESBM-D		ESBM-L		FED	
	$k = 5$	$k = 10$	$k = 5$	$k = 10$	$k = 5$	$k = 10$
RELIN	0.250	0.468	0.210	0.260	0.102	0.235
DIVERSUM	0.260	0.522	0.222	0.365	0.112	0.266
FACES	0.272	0.439	0.160	0.259	0.145	0.273
FACES-E	0.285	0.527	0.252	0.348	0.145	0.273
CD	0.299	0.531	0.215	0.326	0.147	0.271
LinkSUM	0.290	0.498	0.117	0.255	0.236	0.332
NEST	0.354 ▲▲▲▲▲▲	0.540 ▲△○○▲▲	0.332 ▲▲▲▲▲▲	0.465 ▲▲▲▲▲▲	0.272 ▲▲▲▲▲○	0.346 ▲▲▲▲▲○
ORACLE	0.601	0.721	0.631	0.680	0.530	0.582

Table 2: Comparison with existing methods for general-purpose entity summarization (F1). Significant improvements ($p < 0.01$ and $p < 0.05$) achieved by NEST over 6 baselines are indicated by ▲ and △, respectively. Insignificant differences are indicated by ○.

4.4 Baseline Methods

We compared with 11 baseline methods and an oracle method as a reference point used for comparisons.

Unsupervised methods. We compared with 6 unsupervised methods for general-purpose entity summarization: RELIN [Cheng *et al.*, 2011], DIVERSUM [Sydow *et al.*, 2013], FACES [Gunaratna *et al.*, 2015], FACES-E [Gunaratna *et al.*, 2016], CD [Xu *et al.*, 2016], and LinkSUM [Thalhammer *et al.*, 2016].

Supervised methods. We also implemented 5 supervised neural methods as baselines, including 4 methods based on different graph embeddings: TransE [Bordes *et al.*, 2013], Jointly (A-LSTM) [Xu *et al.*, 2017], RotatE [Sun *et al.*, 2019], RDF2Vec (K2V SG 300) [Ristoski *et al.*, 2019]; and 1 state-of-the-art method for document summarization: BERTSUMEXT (large) [Liu and Lapata, 2019]. In each graph embedding based method, we represented a triple by concatenating the embedding vectors of the property and the value in the triple. Then we performed pointwise learning-to-rank (L2R) and chose the k top-ranked triples as a summary. We experimented with 3 L2R models: XGBoost (xgb), random forest (rf), and naive Bayes (nb). To adapt the document summarization method BERTSUMEXT to solve entity summarization, we generated a pseudo sentence from each triple by concatenating the textual forms of its three RDF terms. All these methods and our NEST were trained and tuned on the same training and validation sets.

Oracle method. We implemented an oracle method to show the highest achievable F1 as a reference point used for comparisons. Recall that ESBM and FED provided multiple ground-truth summaries for each entity. ORACLE selected k triples that appeared in the most ground-truth summaries.

5 Results and Analysis

We firstly compared NEST with baselines and then conducted an ablation study. We tested three research hypotheses. **RH1**: NEST with weak supervision can outperform existing methods for general-purpose entity summarization. **RH2**: The novel neural model of NEST can outperform existing neural models in the entity summarization task. **RH3**: The joint encoding in NEST can be more effective than separate structure encoding and text encoding.

5.1 Comparison with Existing Methods for Entity Summarization and ORACLE

We compared NEST with 6 methods for general-purpose entity summarization. F1 scores are presented in Table 2. NEST achieved new state-of-the-art results on all the three datasets for both $k = 5$ and $k = 10$. It significantly ($p < 0.01$) outperformed all the baselines on ESBM-D ($k = 5$) and ESBM-L ($k = 5$ and $k = 10$). In other cases, differences were mostly significant. When $k = 5$, compared with the best-performing baseline on each dataset, NEST raised F1 from 0.299 to 0.354 by 18% on ESBM-D, from 0.252 to 0.332 by 32% on ESBM-L, and from 0.236 to 0.272 by 15% on FED. *The results supported research hypothesis RH1.*

Although these baseline methods were unsupervised while NEST was supervised, our supervision was weak as our labeled data was programmatically generated and hence free of manual work. Further, the generation used DBpedia, while NEST also showed the best performance on LinkedMDB (i.e., ESBM-L), demonstrating its generalizability.

The F1 scores of ORACLE were in the range of 0.530–0.721. It was impossible for ORACLE or any other method to reach F1 = 1, because for each entity there were multiple different ground-truth summaries which could not simultaneously match a computed summary. NEST stepped closer to ORACLE than the baselines, but the gap was still considerable and suggested room for improvement.

5.2 Comparison with Supervised Neural Methods

We compared NEST with 5 supervised neural methods. F1 scores are presented in Table 3. Among these baselines, RotatE with XGBoost (xgb) generally achieved the best result on ESBM-D, while BERTSUMEXT topped on ESBM-L and FED. However, none of these methods were comparable with NEST. When $k = 5$, compared with the best-performing baseline on each dataset, NEST raised F1 from 0.297 to 0.354 by 19% on ESBM-D, from 0.248 to 0.332 by 34% on ESBM-L, and from 0.185 to 0.272 by 47% on FED. *The results supported research hypothesis RH2.*

All these methods were supervised and were trained and tuned on the same data. Therefore, we mainly attributed the superiority of NEST to the novelty of our KG encoder which jointly extracted and tightly fused structural and textual features from KGs by keeping track of the structural dependencies between textualized RDF terms. By contrast, TransE,

	ESBM-D						ESBM-L						FED					
	$k=5$			$k=10$			$k=5$			$k=10$			$k=5$			$k=10$		
	xgb	rf	nb	xgb	rf	nb	xgb	rf	nb	xgb	rf	nb	xgb	rf	nb	xgb	rf	nb
TransE	0.263	0.230	0.250	0.440	0.466	0.487	0.085	0.117	0.152	0.215	0.241	0.262	0.140	0.131	0.184	0.264	0.267	0.292
Jointly	0.232	0.223	0.252	0.455	0.436	0.469	0.203	0.193	0.188	0.309	0.283	0.238	0.164	0.144	0.149	0.274	0.268	0.268
RotatE	0.297	0.277	0.191	0.483	0.458	0.461	0.221	0.172	0.231	0.323	0.310	0.275	0.146	0.123	0.130	0.256	0.252	0.263
RDF2Vec	0.225	0.249	0.174	0.444	0.454	0.443	0.161	0.136	0.140	0.303	0.322	0.248	0.139	0.129	0.177	0.272	0.254	0.276
BERTSUMEXT		0.245			0.400			0.248			0.351			0.185			0.287	
NEST		0.354			0.540			0.332			0.465			0.272			0.346	

Table 3: Comparison with supervised neural methods (F1).

	ESBM-D		ESBM-L		FED	
	$k=5$	$k=10$	$k=5$	$k=10$	$k=5$	$k=10$
NEST	0.354	0.540	0.332	0.465	0.272	0.346
w/o structure	0.339	0.524	0.318	0.445	0.250	0.320
w/o text	0.301	0.504	0.284	0.440	0.226	0.317

Table 4: Ablation study (F1).

RotatE, and RDF2Vec only encoded graph structure, BERTSUMEXT only encoded text, and Jointly loosely combined structure and text in a linear way.

Furthermore, among all the 11 baselines, supervised methods did not perform better than unsupervised methods. Therefore, developing an effective supervised method for general-purpose entity summarization was not a trivial task. It helped to demonstrate the significance of our work.

5.3 Ablation Study

We implemented two variants of NEST to analyze the effectiveness of its novel joint encoding. The first variant (w/o structure) was not pre-trained with random walks in KGs but directly fine-tuned, thus focusing on textual features. In the second variant (w/o text), the initial representation of an RDF term was given by its identifier rather than using word embeddings, thus focusing on structural features.

F1 scores are presented in Table 4. Compared with NEST, the performance of both variants decreased notably. Text had a stronger influence than graph structure. *The results supported research hypothesis RH3.* In fact, the two variants still outperformed all the baselines on ESBM-D ($k = 5$) and ESBM-L ($k = 5$ and $k = 10$), demonstrating the effectiveness of our scorer and weak supervision.

6 Related Work

Extensive research has been focused on general-purpose entity summarization. RELIN [Cheng *et al.*, 2011] computes the informativeness of a triple. DIVERSUM [Sydow *et al.*, 2013] improves the diversity of a summary by choosing triples about different properties. FACES [Gunaratna *et al.*, 2015] and FACES-E [Gunaratna *et al.*, 2016] cluster similar triples to improve diversity and rank triples by their informativeness and value frequency. CD [Xu *et al.*, 2016] formulates entity summarization as a quadratic knapsack problem to select the most informative while dissimilar triples. LinkSUM [Thalhammer *et al.*, 2016] considers PageRank and backlinks. All of these methods are unsupervised, using various heuristics for ranking and selecting triples. By contrast, our approach is supervised, and it features a novel neural model to neatly encode structure and text in KGs.

There are also methods for summarizing entity descriptions for a particular task, e.g., entity linking [Cheng *et al.*, 2015b], entity matching [Cheng *et al.*, 2015a], entity search [Hasibi *et al.*, 2017], and document enrichment [Gunaratna *et al.*, 2017]. They focus on task-specific techniques, thus not comparable with our work. Although some of these methods also perform supervised learning, they rely on manually defined task-specific features [Zhang *et al.*, 2012; Hasibi *et al.*, 2017], while we employ a novel neural model to jointly extract structural and textual features from KGs. Besides, for their search task, labeled data is relatively easy to obtain from relevance judgment. However, for general-purpose entity summarization, we have to programmatically generate labeled data. Rather than extracting a subset of triples as a summary, some methods generate a textual summary using a sequence-to-sequence framework [Lebret *et al.*, 2016; Hachey *et al.*, 2017; Vougiouklis *et al.*, 2018]. Their problem and techniques are fundamentally different from ours.

Our KG encoder can be viewed as a graph embedding method [Cai *et al.*, 2018; Cui *et al.*, 2019; Gesese *et al.*, 2019]. Similar to RDF2Vec [Ristoski *et al.*, 2019] and KGloVe [Cochez *et al.*, 2017], we sample random walks to capture the graph structure of a KG. Whereas they encode walks as sequences of RDF term identifiers, we further exploit the textual form of each RDF term to jointly extract structural and textual features. This way of fusion is tighter than straightforward linear combination adopted by existing methods [Xie *et al.*, 2016; Xu *et al.*, 2017].

7 Conclusion

We presented NEST, a novel neural approach to general-purpose entity summarization. Its KG encoder is pre-trained to jointly extract structural and textual features from KGs. Its summary scorer is fine-tuned to generate high-quality diversified summaries based on programmatically labeled data. Extensive experiments have demonstrated the effectiveness and generalizability of NEST, which significantly outperforms the state of the art on two public benchmarks.

Our KG encoder has the potential to support other downstream tasks such as entity clustering and link prediction. Besides, it is possible to fine-tune our model to generate entity summaries for specific tasks (e.g., entity search) based on task-specific training data (e.g., relevance judgments).

Acknowledgements

This work was supported partially by the National Key R&D Program of China (2018YFB1005100) and the Six Talent Peaks Program of Jiangsu Province (RJFW-011).

References

[Bojanowski *et al.*, 2017] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Trans. Assoc. Comput. Linguist.*, 5:135–146, 2017.

[Bordes *et al.*, 2013] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, pages 2787–2795, 2013.

[Cai *et al.*, 2018] HongYun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Trans. Knowl. Data Eng.*, 30(9):1616–1637, 2018.

[Cheng *et al.*, 2011] Gong Cheng, Thanh Tran, and Yuzhong Qu. RELIN: relatedness and informativeness-based centrality for entity summarization. In *ISWC, Part I*, pages 114–129, 2011.

[Cheng *et al.*, 2015a] Gong Cheng, Danyun Xu, and Yuzhong Qu. C3D+P: A summarization method for interactive entity resolution. *J. Web Semant.*, 35:203–213, 2015.

[Cheng *et al.*, 2015b] Gong Cheng, Danyun Xu, and Yuzhong Qu. Summarizing entity descriptions for effective and efficient human-centered entity linking. In *WWW*, pages 184–194, 2015.

[Cochez *et al.*, 2017] Michael Cochez, Petar Ristoski, Simone Paolo Ponzetto, and Heiko Paulheim. Global RDF vector space embeddings. In *ISWC, Part I*, pages 190–207, 2017.

[Cui *et al.*, 2019] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. A survey on network embedding. *IEEE Trans. Knowl. Data Eng.*, 31(5):833–852, 2019.

[Gesese *et al.*, 2019] Genet Asefa Gesese, Russa Biswas, Mehwish Alam, and Harald Sack. A survey on knowledge graph embeddings with literals: Which model links better literal-ly? *CoRR*, abs/1910.12507:1–24, 2019.

[Gunaratna *et al.*, 2015] Kalpa Gunaratna, Krishnaprasad Thirunarayan, and Amit P. Sheth. FACES: diversity-aware entity summarization using incremental hierarchical conceptual clustering. In *AAAI*, pages 116–122, 2015.

[Gunaratna *et al.*, 2016] Kalpa Gunaratna, Krishnaprasad Thirunarayan, Amit P. Sheth, and Gong Cheng. Gleaning types for literals in RDF triples with application to entity summarization. In *ESWC*, pages 85–100, 2016.

[Gunaratna *et al.*, 2017] Kalpa Gunaratna, Amir Hossein Yazdavar, Krishnaprasad Thirunarayan, Amit P. Sheth, and Gong Cheng. Relatedness-based multi-entity summarization. In *IJCAI*, pages 1060–1066, 2017.

[Hachey *et al.*, 2017] Ben Hachey, Will Radford, and Andrew Chisholm. Learning to generate one-sentence biographies from Wikidata. In *EACL*, pages 633–642, 2017.

[Hasibi *et al.*, 2017] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. Dynamic factual summaries for entity cards. In *SIGIR*, pages 773–782, 2017.

[Lebret *et al.*, 2016] Rémi Lebret, David Grangier, and Michael Auli. Neural text generation from structured data with application to the biography domain. In *EMNLP*, pages 1203–1213, 2016.

[Liu and Lapata, 2019] Yang Liu and Mirella Lapata. Text summarization with pretrained encoders. In *EMNLP-IJCNLP*, pages 3728–3738, 2019.

[Liu *et al.*, 2019] Qingxia Liu, Gong Cheng, Kalpa Gunaratna, and Yuzhong Qu. Entity summarization: State of the art and future challenges. *CoRR*, abs/1910.08252:1–40, 2019.

[Liu *et al.*, 2020] Qingxia Liu, Gong Cheng, Kalpa Gunaratna, and Yuzhong Qu. ESBM: an entity summarization benchmark. In *ESWC*, pages 1–16, 2020.

[Ristoski *et al.*, 2019] Petar Ristoski, Jessica Rosati, Tommaso Di Noia, Renato De Leone, and Heiko Paulheim. RDF2Vec: RDF graph embeddings and their applications. *Semant. Web*, 10(4):721–752, 2019.

[Sun *et al.*, 2019] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. RotatE: knowledge graph embedding by relational rotation in complex space. In *ICLR*, pages 1–18, 2019.

[Sydow *et al.*, 2013] Marcin Sydow, Mariusz Pikula, and Ralf Schenkel. The notion of diversity in graphical entity summarisation on semantic knowledge graphs. *J. Intell. Inf. Syst.*, 41(2):109–149, 2013.

[Thalhammer *et al.*, 2016] Andreas Thalhammer, Nelia Lasierra, and Achim Rettinger. LinkSUM: using link analysis to summarize entity data. In *ICWE*, pages 244–261, 2016.

[Vougiouklis *et al.*, 2018] Pavlos Vougiouklis, Hady ElSahar, Lucie-Aimée Kaffee, Christophe Gravier, Frédérique Laforest, Jonathon Hare, and Elena Simperl. Neural Wikipedian: Generating textual summaries from knowledge base triples. *J. Web Semant.*, 52-53:1–15, 2018.

[Xie *et al.*, 2016] Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. Representation learning of knowledge graphs with entity descriptions. In *AAAI*, pages 2659–2665, 2016.

[Xu *et al.*, 2016] Danyun Xu, Liang Zheng, and Yuzhong Qu. CD at ENSEC 2016: Generating characteristic and diverse entity summaries. In *SumPre*, pages 1–3, 2016.

[Xu *et al.*, 2017] Jiacheng Xu, Xipeng Qiu, Kan Chen, and Xuanjing Huang. Knowledge graph representation with jointly structural and textual encoding. In *IJCAI*, pages 1318–1324, 2017.

[Zhang *et al.*, 2012] Lanbo Zhang, Yi Zhang, and Yunfei Chen. Summarizing highly structured documents for effective search interaction. In *SIGIR*, pages 145–154, 2012.