Net2Text: An Edge Labelling Language Model for Personalized Review Generation

No Author Given

No Institute Given

Abstract. Writing an item review for online shopping or sharing the dining experience of a restaurant has become major Internet activities of young people. This kind of review system could not only help users express and exchange experience but also prompt business to improve service quality. Instead of taking time to type in the review, we would like to make the review process more automated. In this work, we study an edge labelling language model for personalized review generation, *e.g.*, the problem of generating text (*e.g.*, a review) on the edges of the network (*e.g.*, online shopping). It is related to both network structure and rich text semantic information. Previously, link prediction models have been applied to recommender system and event prediction. However, they could not migrate to text generation on the edges of networks since most of them are numerical prediction or tag labelling tasks. To bridge the gap between link prediction and natural language generation, in this paper, we propose a model called Net2Text, which can simultaneously learn the structural information in the network and build a language model over text on the edges. The performance of Net2Text is demonstrated in our experiments, showing that our model performs better than other baselines, and is able to produce reasonable reviews between users and items.

Keywords: Link Prediction · Language Model · Graph Mining · Data Mining

1 Introduction

Under the wave of mobile Internet development, we witness and engage more and more online review system. For example, before we purchase a product on Amazon we always check reviews from other buyers to determine if buy it or not. Meanwhile, we write our own using experience for the products purchased as well. In addition to online shopping, we also focus on movie reviews and restaurant reviews. Review system influences various aspects of our life.

Sometimes, writing a review is a trivial work for customers, and most customers share thoughts on fixed aspects (*e.g.*, quality, service). In this work, we would like to help customers generate accurate reviews automatically given historical reviews between users and items. Descriptive text can encode rich semantic information and structural relationship between user-item network.

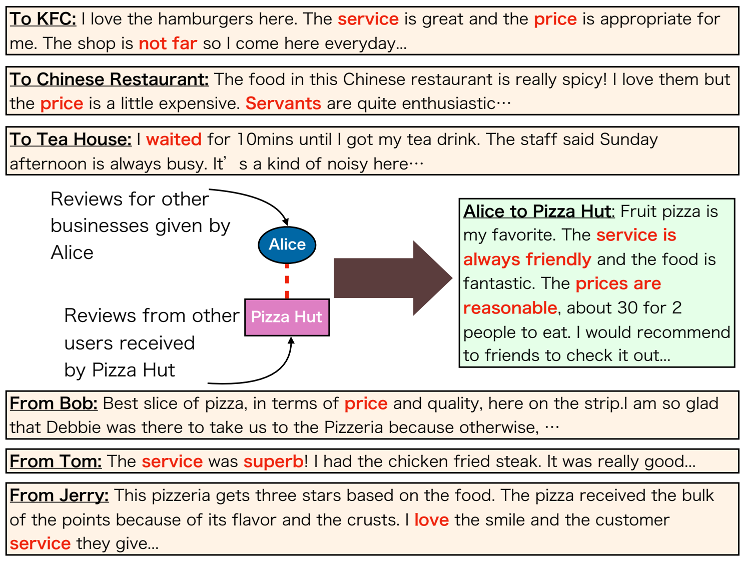
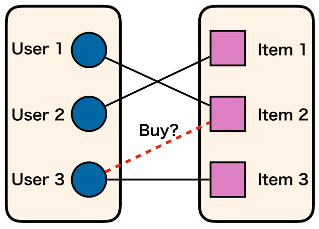

To KFC: I love the hamburgers here. The service is great and the price is appropriate for me. The shop is not far so I come here everyday...
To Chinese Restaurant: The food in this Chinese restaurant is really spicy! I love them but the price is a little expensive. Servants are quite enthusiastic…
To Tea House: I waited for 10mins until I got my tea drink. The staff said Sunday afternoon is always busy. It' s a kind of noisy here…
Reviews for other businesses given by Alice
Alice
Pizza Hut
Reviews from other users received by Pizza Hut
Alice to Pizza Hut: Fruit pizza is my favorite. The service is always friendly and the food is fantastic. The prices are reasonable, about 30 for 2 people to eat. I would recommend to friends to check it out...
From Bob: Best slice of pizza, in terms of price and quality, here on the strip.I am so glad that Debbie was there to take us to the Pizzeria because otherwise, …
From Tom: The service was superb! I had the chicken fried steak. It was really good...
From Jerry: This pizzeria gets three stars based on the food. The pizza received the bulk of the points because of its flavor and the crusts. I love the smile and the customer service they give...

Fig. 1: Our personalized review generation task. Top blocks are reviews that Alice gave to other restaurants, and bottom blocks are reviews that Pizza Hut received from other customers. Our goal is to help Alice generate review automatically for Pizza Hut (see the text in the green box). We provide detail explanation in Section 1.

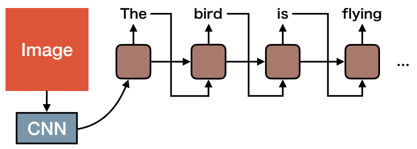
Figure 1 describes our task of personalized review generation. Top blocks are reviews that Alice gave to other restaurants, and bottom blocks are reviews that Pizza Hut received from other customers. After analyzing the reviews given by Alice, we can conclude that he concerns more about price, position and service (emphasized in red bold). Pizza Hut has received some reviews from other customers and some of them mentioned the attributes that Alice cares about. Since most of reviews to Pizza Hut are positive, our generated review for Alice is friendly and reasonable. Consequently, it will help users to alleviate the burden of typing by generating accurate reviews automatically.

Essentially, it is a kind of edge labelling task for network, which can be recognized as a link prediction problem. In previous researches, link prediction models are mainly applied to recommender system. For example, in social networks, we usually predict the evolution of networks like recommending new friends [1] by calculating similarities based on user activities. In online e-commerce websites, instead of just recommending new links, link prediction models also predict events like buying an item (Figure 2a). In this area, collaborative filtering (CF) is a successful recommendation paradigm that employs transaction information to enrich user and item features for recommendation [12]. However, these models could not migrate to text generation on the edges since most of them are numerical prediction or tag labelling tasks.

From the aspect of language modeling, although we have a lot of sophisticated language models nowadays, it is not a trivial task to employ it into network



(a) Recommender system



(b) Image caption [13]

Fig. 2: Previous models on link prediction task and multimodel language generation task.

structure. Common language generation models, including n-gram models [7] and deep learning models (*e.g.*, LSTM [25]), always generate text without considering additional features of network structures. Text produced by these models cannot reflect the specialization between vertices. Karpathy et al. proposed an image caption model (Figure 2b) and it is a kind of language model that considers image features [13]. This model cannot be applied in network structure but inspired us to build a multimodel language model over a network structure.

To our best of knowledge, there are no previous models on personalized review generation. Therefore, in this paper, we propose a model, Net2Text, based on network structure with text on the edges. Our model innovatively solves the task of generating personalized text on network edges automatically and makes up the gap in the study of edge labelling language model.

Our method proposed in this paper has following three-fold contributions:

1. It is the first work to propose automated personalized review generation task, which is an extensive work of image caption to network structure. It builds a bridge between network structure and text generation.
2. Our Net2Text is a novel model on edge labelling for personalized comment generation by learning the structural information in the network and language modeling in the text.
3. We demonstrate the performance of our model in experiments on real datasets. By comparing Net2Text to other baselines, we show that our model performs better under machine translation metrics. We also present some generated reviews to prove the readability and personality.

2 Problem Definition

In this section, we first introduce related concepts and notations, then define our problem.

In our task, we have two types of vertices in network. One is the user (Alice), the other is the item or business (Pizza Hut). Therefore, we denote a network, $G = (\mathcal{V}_1, \mathcal{V}_2, \mathcal{E})$, where $\mathcal{V}_1 = \{1, \cdots, n_1\}$ and $\mathcal{V}_2 = \{1, \cdots, n_2\}$ are two partitions of this network, and $\mathcal{E} = \{(l_i, r_i, \mathbf{s}_i)\}_{i=1}^{m}$ denotes the edges of the network. m is the total number of edges in G. For the i-th edge, $l_i \in \mathcal{V}_1$ and $r_i \in \mathcal{V}_2$ are two

Table 1: Important Notations.

Symbol	Definition
$G = (\mathcal{V}_1, \mathcal{V}_2, \mathcal{E})$	A network with text on edges.
$\mathcal{V}_1 = \{1, \cdots, n_1\}, \mathcal{V}_2 = \{1, \cdots, n_2\}$	Two sets of vertices, n_1 and n_2 are vertex number of each set.
$\mathcal{E} = \{(l_i, r_i, \mathbf{s}_i)\}_{i=1}^{m}$	The set of m links in network G.
l_i, r_i	Two connected vertices of i-th link in $\mathcal{E}$.
$\mathbf{s}_i = (s_i^1, \cdots, s_i^{T_i})$	Sequence of i-th edge text. $\forall t \in [1..T_i]$, s_i^t correspondes to a word of text $\mathbf{s}_i$ at position t. T_i is the number of words on the i-th sequence.
$\mathbf{x}_i = (s_i^0, \cdots, s_i^{T_i})$	Text sequence on i-th edge $\mathbf{s}_i$ with an additional pre tag *START*.
$\mathcal{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_m\}$	Input vectors for all instances.
$\mathcal{F} = \{\mathbf{f}_1, \cdots, \mathbf{f}_m\}$	Link features for all instances.
$\mathbf{f}_i = [\Phi(l_i), \Phi(r_i)] \in \mathbb{R}^{2 \times d}$	i-th link features combined by vector representation of vertex $l_i \in \mathcal{V}_1$ and vertex $r_i \in \mathcal{V}_2$. d is the dimension of each vertex vector.
$\mathcal{Y} = \{\mathbf{y}_1, \cdots, \mathbf{y}_m\}$	Output vectors for all instances.
$\mathbf{y}_i = (s_i^1, \cdots, s_i^{T_i+1})$	i-th edge text sequence $\mathbf{s}_i$ with an additional post tag *END*.
$\Phi \in \mathbb{R}^{(n_1+n_2) \times d}$	Representation vectors for all the vertices in network G.
$\mathcal{L}$ and $\mathcal{U}$	The training set and testing set.

connected vertices. $\mathbf{s}_i = (s_i^1, \cdots, s_i^{T_i})$ denotes the text sequence on i-th edge. $\forall t \in [1..T_i]$, s_i^t correspondes to a word in sequence $\mathbf{s}_i$ at position t. T_i is the number of words of the sequence on the i-th edge.

Now we formally define our problem as follow:

Definition 1. *Given a network $G = (\mathcal{V}_1, \mathcal{V}_2, \mathcal{E})$ with text on its edges, for each edge $(l, r, \mathbf{s}) \in \mathcal{E}$, our goal is to learn an edge labelling language model $g(l, r) \approx \mathbf{s}$, that can map all the edges in the network into personalized reviews. For a new edge (l_{new}, r_{new}), we could generate a personalized review $\mathbf{s}_{new} = g(l_{new}, r_{new})$.*

Since we take network structure information into consideration in our task, for i-th edge $(l_i, r_i, \mathbf{s_i})$ we begin with learning vector representation $\Phi(l_i) \in \mathbb{R}^d$ of vertex $l_i \in \mathcal{V}_1$ and $\Phi(r_i) \in \mathbb{R}^d$ of vertex $r_i \in \mathcal{V}_2$, and then we denote a vector $\mathbf{f}_i = [\Phi(l_i), \Phi(r_i)] \in \mathbb{R}^{2 \times d}$ by concatenating representation of two vertices as features of i-th edge. d is the embedding dimensions of each vertex.

Now we create a new edge set $\mathcal{E}' = (\mathcal{X}, \mathcal{F}, \mathcal{Y})$, where $\mathcal{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_m\}$ represents input vectors, $\mathcal{F} = \{\mathbf{f}_1, \cdots, \mathbf{f}_m\}$ represents edge features and $\mathcal{Y} = \{\mathbf{y}_1, \cdots, \mathbf{y}_m\}$ represents output vectors.

For the sequence $\mathbf{s}_i$, we add a special tag *START* as s_i^0 to the beginning, which forms our input vector $\mathbf{x}_i = (s_i^0, \cdots, s_i^{T_i})$ for i-th edge. We add another special tag *END* as $s_i^{T_i+1}$ to the end, which forms our output vector $\mathbf{y}_i = (s_i^1, \cdots, s_i^{T_i+1})$.

Our instances in $\mathcal{E}'$ are then divided into a training set $\mathcal{L} \subset \{\mathcal{E}'_1, \cdots, \mathcal{E}'_m\}$ and test set $\mathcal{U} \subset \{\mathcal{E}'_1, \cdots, \mathcal{E}'_m\}$, where $\mathcal{L} \bigcup \mathcal{U} = \mathcal{E}'$ and $\mathcal{L} \bigcap \mathcal{U} = \emptyset$. For $\forall i \in \mathcal{L}$,

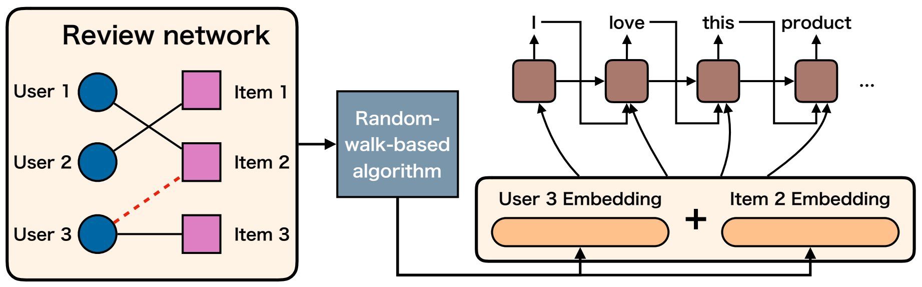


Fig. 3: Architecture of Net2Text. We first learn vertex representations from review network. For unconnected user 3 and item 2, we combine their embeddings into an edge feature, together with each word in the review sequence, as the input to the nGRU.

$\mathbf{x}_i$, $\mathbf{f}_i$ and $\mathbf{y}_i$ are fully observed, while $\forall i \in \mathcal{U}$, we only observe x_i^0, which is tag *START*, and feature $\mathbf{f}_i$ for i-th edge. Since we use previous output as new input in text generation process, we denote $\mathbf{y}_i^o$ as previously generated word sequence. Here, $\mathbf{y}_i^o = (y_i^0, \cdots, y_i^{t-1})$ changes as the step goes to t-th position.

Our task on the i-th instance is to predict $\mathbf{y}_i = (y_i^j)_{j=1}^{T_i}$ based on $\mathbf{y}_i^o$ and $\mathbf{f}_i$ until tag *END* appears or designated text length arrives.

We use $\mathcal{Y}_\mathcal{U} = \{\mathbf{y}_i | \forall i \in \mathcal{U}\}$ to denote all the target text on unconnected vertices for prediction. In addition, we define $\mathcal{Y}_\mathcal{U}^o = \{\mathbf{y}_i^o | \forall i \in \mathcal{U}\}$ to denote all the generated outputs during the test phase. $\mathcal{F}_\mathcal{U} = \{\mathbf{f}_i | \forall i \in \mathcal{U}\}$ is the collection of edge features in test set $\mathcal{U}$. The overall goal of Net2Text model is to estimate a probability distribution:

$$\Pr(\mathcal{Y}_\mathcal{U} | \mathcal{F}_\mathcal{U}, \mathcal{Y}_\mathcal{U}^o) \propto \prod_{i \in \mathcal{U}} \Pr(\mathbf{y}_i | \mathbf{y}_i^o, \mathbf{f}_i) \tag{1}$$

Table 1 explains details of our important notations.

3 Proposed Method

3.1 Overview

In this section we will explain details of our method.

In general, our model integrates the process of vertex embedding and text generation. We first learn representations of vertices with random-walk-based method, then train a language generation model over existing text on the edges by incorporating vertical features. Figure 3 describes the architecture of our model Net2Text.

Algorithm 1: Learning vector representations.

```
  Input: vertices set $\mathcal{V}$, edge set $\mathcal{E}$, windows size $w$
         embedding size $d$, number of walks per vertex $b$
  Output: matrix of vector representations $\Phi \in \mathbb{R}^{N \times d}$
1 Initialize $\Phi$
2 for $i = 1$ to $b$ do
3     Reorder the vertices set $\mathcal{V}$
4     foreach $v_i \in \mathcal{V}$ do
5         $\mathcal{W}_{v_i} = RandomWalk(\mathcal{V}, \mathcal{E}, v_i)$
6         $SkipGram(\Phi, \mathcal{W}_{v_i}, w)$
7     end
8 end
```

3.2 Learning representation of vertices

To learn latent representations of vertices in a network, we use local information obtained from truncated random walks by treating walks as the equivalent of sentences [21]. We merge vertices from $\mathcal{V}_1$ and $\mathcal{V}_2$ into $\mathcal{V} = \{1, \cdots, N\}$ where $N = n_1 + n_2$, then we define $\Phi \in \mathbb{R}^{N \times d}$ as vector representations of all the vertices.

We initialize the mapping function Φ by uniformly sampling a random walk. We choose a random vertex v_i as the root of the walk, then sample uniformly from the neighbors of previous vertices visited. Therefore, the objective function of this optimization problem is:

$$\min_{\Phi} -log \Pr(\{v_{i-w}, \cdots, v_{i-1}, v_{i+1}, \cdots, v_{i+w}\} | \Phi(v_i)) \tag{2}$$

where w is the context windows size for each vertex in the walk.

In each iteration, we start a new random walk at each vertex and update the objective function Φ. For each vertex v_i in the inner loop, we generate a random walk $\mathcal{W}_{v_i}$, which starts from vertex in one side to the vertex in the other side back and forth in the network. We then use the generated walk to update our representations Φ and our objective function Equation 3.2 with SkipGram [18] loop. For each vertex $v_j \in \mathcal{W}_{v_i}$ we map it to current representation vector $\Phi(v_j) \in \mathbb{R}^d$, and maximize the probability of its neighbors $\{u_k\}$ with context window size w in the walk.

3.3 Text generation on edges

Next step after learning vertex representations is to incorporate them into our edge labelling language model.

Formally, for each edge $(l_i, r_i, \mathbf{s}_i)$ in the network G, we first construct edge features $\mathbf{f}_i = [\Phi(l_i), \Phi(r_i))]$ by combining vector representations of vertex l_i and vertex r_i, then convert text $\mathbf{s}_i$ on the edge into a sequence of words $\mathbf{x}_i = (x_i^0, \cdots, x_i^{T_i})$.

Recurrent Neural Network (RNN) [16] is one of frequently used deep neural language model. It defines a probability distribution of the next word given the current word and previous context sequence generated. Simple RNN unit in vanilla version could not capture long dependency due to the gradient explosion/vanishing problem, therefore, we replace it with Gated Recurrent Unit [8]. Since current GRU model just tries to generate text that has correct spelling and grammar while ignoring our network structural information, we define a new unit called *nGRU*. nGRU makes an improvement to the original GRU, that it can condition text generation on vector representations Φ of all the vertices learned from previous process to build our edge labelling language model for our personalized review generation task.

The t-th nGRU unit has two gates: an update gate $\mathbf{z}_t$ and a reset gate $\mathbf{r}_t$, and two states: a candidate state $\tilde{\mathbf{h}}_t$ and a hidden state $\mathbf{h}_t$. For each step t, we compute input vector $\mathbf{u}_t$ which is a linear transformation of edge features f_i^t and current t-th word x_i^t, and outputs the next word $\mathbf{o}_t$. Update rules for our recurrent units are below:

$$\mathbf{u}_t = W_{uf} f_i^t + W_{ux} embed(x_i^t) + \mathbf{b}_u \tag{3}$$

$$\mathbf{z}_t = \sigma(W_{zu}\mathbf{u}_t + W_{zh}\mathbf{h}_{t-1}) \tag{4}$$

$$\mathbf{r}_t = \sigma(W_{ru}\mathbf{u}_t + W_{rh}\mathbf{h}_{t-1}) \tag{5}$$

$$\tilde{\mathbf{h}}_t = \tanh(W_{hu}\mathbf{u}_t + W_{hh}(\mathbf{r}_t \odot \mathbf{h}_{t-1})) \tag{6}$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t \tag{7}$$

$$\mathbf{o}_t = softmax(W_{oh}\mathbf{h}_t + \mathbf{b}_o) \tag{8}$$

Here, function *embed*() turns a word into a low dimension embedding vector. It can be assigned with fixed global word vector or trained by the model itself. $\mathbf{W}^{(\cdot)}$ are weight matrix and $\mathbf{b}^{(\cdot)}$ are bias vectors. σ is the logistic function and the operator $\odot$ means element-wise product between two vectors. We initialize $\mathbf{h}_0$ as a vector of all zeros.

Until now, we build the basic architecture of our model as shown in Figure 3, in next part we will introduce the procedure of our algorithm.

3.4 Net2Text Algorithm

We conclude a procedure of our method as shown in Algorithm 2. The algorithm has following steps:

1. **Data Preparing.** First, we use random-walk-based algorithm to extract the representations of all the vertices Φ, and generate a new edge set $\mathcal{E}' = (\mathcal{X}, \mathcal{F}, \mathcal{Y})$ via the problem definition mentioned in Section 2.
2. **Model Training.** We extend text vectors $\mathcal{X}$ to a new combined vectors $\mathcal{X}'$ by concatenating edge features to each word over the whole sequence. With this operation, we can take local network features into consideration while modeling languages. Finally we train a language model g on dataset $\mathcal{X}'$.

Algorithm 2: Net2Text Algorithm

```
Input: Network $G = (\mathcal{V}, \mathcal{E})$, training set $\mathcal{L}$, test set $\mathcal{U}$
       windows size $w$, embedding size $d$, walks per vertex $b$, max sentence
       length $p_{max}$
Output: $\mathcal{Y}_{\mathcal{U}} = \{\mathbf{y}_i | \forall i \in \mathcal{U}\}$, each $\mathbf{y}_i$ is a generated link text in test set $\mathcal{U}$.
Data Preparing:
    - Generate vector representations:
      $\Phi = \textit{Learning vector representations}(\mathcal{V}, \mathcal{E}, w, d, b)$
    - Create new link dataset: $\mathcal{E}' = (\mathcal{X}, \mathcal{F}, \mathcal{Y})$
Training Phase:
    - Construct extended dataset $\mathcal{X}' = (\mathbf{x}'_1, \cdots, \mathbf{x}'_m)$:
    for $i = 1$ to $m$ do
        Extend $i$-th link $\mathbf{x}'_i = ([x_i^0, f_i], \cdots, [x_i^T, f_i])$ by concatenating link
          features to each timestep over the whole text.
    end
    - Let $g$ = Net2Text($\mathcal{X}'$) be our model trained on $\mathcal{X}'$
Test:
    foreach $i \in \mathcal{U}$ do
        - Use START as our first token: $y_i^0$=(START)
        - Initialize our test parameters: $len = 1$, $\mathbf{x}_i = ([y_i^0, f_i])$
        repeat
            - Predict next character with our trained model: token = $g(\mathbf{x}_i)$
            - Append newly generated character to existed text:
              $\mathbf{y}_i$.append(token)
            - Update input vector with the new token: $\mathbf{x}_i$.append($[token, f_i]$)
            - Move to next character: $len = len + 1$
        until $token = END$ or $len > p_{max}$
    end
```

3. **Text Prediction.** In the testing phase, we first set tag *START* as the initial word to our model. For each generation step, we update input vectors as in the training phase, and repeat predicting new word until the tag *END* appears or maximum length of text reaches.

4 Experiments

In order to validate the performance of our model, we applied Net2Text to several real world review datasets in our experiments.

4.1 DataSets

• Amazon[1] is the world leading e-ecommerce platform, and customers often write down their reviews for their purchases. Amazon datasets [11] we used

[1] https://www.amazon.com/

Table 2: Experiment Datasets.

Dataset	**Average words/review**	**Number of reviews**
Amazon - Clothing & Jewelry	57	263890
Amazon - Health Care	72	301284
Amazon - Sports	67	263212
Amazon - Video Games	81	151018
Yelp	98	517729

in this experiment contains product reviews, including 142.8 million reviews spanning May 1996 - July 2014. We choose 4 categories of items, and each of which has been reduced to extract the 5-core, such that each of the remaining users and items has 5 reviews each.
• Yelp[2] is one of the biggest online restaurant reservation service provider. It has a huge amount of data, including customer profile, resturant information, customer review and so on. They open-sourced their dataset on the official website[3]. We reduced the dataset to the 30-core, such that each of the remaining customers and businesses has 30 reviews each.

All the datasets are splitted into training sets and test sets. Statistical details on our datasets are presented in Table 2.

4.2 Experiment Setup

We have some basic configurations in our experiments. The vector representation dimension for user and item vertices is set to 256. For language modelling part, we use nGRU units with 512 dimension hidden nodes. We choose cross-entropy [24] as our loss function. In Section 4.4 we compare performance of different dimensions for nGRU units and vector representation. For comparison algorithms we have the same configurations as ours. All the experiments were conducted under a Linux workstation with a Nvidia GTX 1080 GPU (8G graphic memory).

4.3 Algorithms for Comparison

To our best of knowledge, there are no comparable models since we are the first to propose text generation on network edges. In addition to various versions of Net2Text, we evaluated the performance against the following baseline algorithms (Summarized in Table 3):
• wordRNN [26]: This method does not consider the local network structure between user and item but only concentrates on review content. It only builds word level generation language model for all the reviews and generates review based on the first few words. We use the same language model structure as Net2Text but remove edge features.

[2] https://www.yelp.com/
[3] https://www.yelp.com/dataset/challenge

Table 3: Algorithms for comparison.

Method	Content	Network	Publication
wordRNN	✓	⊘	[26]
random-walk-based	⊘	✓	[21, 10]
Net2Text	✓	✓	This paper

Table 4: Experiments results under different datasets and algorithms.

Dataset	**BLEU**		
	wordRNN	random-walk-based	Net2Text
Amazon - Clothing & Jewelry	0.192	0.173	**0.345**
Amazon - Health Care	0.185	0.165	**0.332**
Amazon - Sports	0.181	0.168	**0.357**
Amazon - Video Games	0.184	0.153	**0.323**
Yelp	0.19	0.17	**0.382**

• Random-walk-based algorithm [21, 10]: We compare to a baseline with the idea of edge similarity. It chooses a review from the training set that expresses the most similar edge features. In other words, it only considers network structure information and neglects text content.
• Net2Text: This is our proposed method mentioned in Section 3, which generates personalized review for users conditioned on both historical review content and network relationships between users and items.

4.4 Evaluation

BLEU (Bilingual Evaluation Understudy) score [19] is a commonly used metric in the area of language generation. We apply it to measure the quality of our generated reviews.

We compared our model Net2Text with the other two baseline algorithms. As the results showed in Table 4, our method Net2Text considers not only review semantic information, but also local network structure features, and has significant improvement than baseline algorithms. As a contrast, the baseline random-walk-based algorithm has the worst performance under BLEU score. One explanation is that random-walk-based algorithm only selects a similar existing review without concerning its content, it may be totally different from what the review really is, which leads to a low BLEU score. Instead, wordRNN tries to generate reviews based on first few words from real reviews, and it helps establish a similar context (quality, service) and predicts in the right direction. Therefore, wordRNN performs better than random-walk-based algorithm under BLEU score. We will show some cases in Section 4.6.

4.5 Model Selection

In this section, we will conduct some analysis on model parameters. We mainly consider three aspects of parameter influence. First two are embedding method

Table 5: Experiments results under different embedding methods.

Dataset	**BLEU**	
	Deepwalk	node2vec
Amazon - Clothing & Jewelry	0.202	**0.345**
Amazon - Health Care	0.211	**0.332**
Amazon - Sports	0.302	**0.357**
Amazon - Video Games	0.296	**0.323**
Yelp	0.247	**0.382**

Table 6: Experiments results under different vertex embedding dimension.

Dataset	**BLEU**			
	Dim. 64	Dim. 128	Dim. 256	Dim. 512
Amazon - Clothing & Jewelry	0.205	0.168	0.229	**0.345**
Amazon - Health Care	0.194	0.221	0.214	**0.332**
Amazon - Sports	0.252	0.212	0.241	**0.357**
Amazon - Video Games	0.187	0.199	0.291	**0.323**
Yelp	0.239	0.253	0.343	**0.382**

and embedding dimension for the network. Another one is hidden dimension of nGRU units. All of them have effects on performance according to our experiments.

Influence of embedding method Our model use random-walk-based algorithms as our vertex embedding method, therefore, we compare experiments performance between Deepwalk [21] and node2vec [10]. Table 5 tells us node2vec leads to a better performance. Since node2vec defines a flexible notion of a node's network neighborhood and design a biased random walk procedure, which efficiently explores diverse neighborhoods, and this kind of flexibility in exploring neighborhoods is the key to learning richer representations [10]. We choose the flexible parameters that applicable in our experiments.

Influence of vertex embedding dimensions In accordance with [21], network embedding dimensional representations are distributed, meaning each user or item is expressed by a subset of the dimensions and each dimension contributes to a subset of social concepts expressed by the space. We used the default experiment setup, and changed the variable embedding size d in vector representations learning phase. Table 6 shows that the dimension 512 performs the best, which means larger dimension could express complex social concepts and that could help make personalized review more accurate.

Influence of nGRU parameters In our experiment we chose nGRU as the base language modeling method. To improve the performance of Net2Text, we adjust hidden dimension of nGRU units. Table 7 shows that 512 hidden dimension has significant improvement than other small dimension.

Table 7: Experiments results under different nGRU dimension.

Dataset	**BLEU**		
	Dim. 128	Dim. 256	Dim. 512
Amazon - Clothing & Jewelry	0.225	0.243	**0.345**
Amazon - Health Care	0.282	0.304	**0.332**
Amazon - Sports	0.221	0.289	**0.357**
Amazon - Video Games	0.224	0.25	**0.323**
Yelp	0.282	0.319	**0.382**

4.6 Case Study

Our goal is that the predicted reviews can reflect suggestions and feelings existed in the real reviews as much as possible, since it could alleviate the burden of user's typing modification.

To gain a better understanding of our method, we explain more details on some example reviews we generated. Table 8 lists cases of our results selected from each of five datasets compared to original reviews.

Case 1 is a perfect generated review example. The customer mentions good feedback about the looking and the feeling of purchased product slippers (*a really comfortable slippers* and *Looks great*). In our generated review, we predict precisely the same keywords (*very comfortable* and *They look good*) as the original one.

In case 2, the customer originally gave a long review to the item. Although long sentence generation is more challenging, we still have the ability to capture the important information for this purchase relationship. For example, we produce the name of this product (*vitamins*), and mention the product quality (*works for me* and *good quality product*) and the price (*far cheaper* and *Great price*) as well.

Case 3 is the case from sports shopping in Amazon. We generate the word *easy* as the same from source review. The customer also wrote the joy feeling to have this item, and we generate it in a different expression (*happy to have it* v.s. *good purchase*).

Review in cases 4 looks like a general comment to video games. We emphasize graphical performance that summarize description in source review.

After comparing the generated review with the original review in case 5, we find that this customer went to a Japanese food resturant and he/she praised the food (*great sushi*, *enjoy the panko a lot* and *love family style ramen*). This customer also complained about the waiting time (*Always a wait* and *always looking for a spot*).

According to above cases, our model could not only capture real facts about items/restaurants but also express original meanings for users. This work will greatly help users alleviate the trouble of typing text by filling the review content automatically.

Table 8: Comparison of generated reviews with ground-truth reviews.

Case 1: Amazon - Clothing & Jewelry	
Source	Crocs are a really comfortable slippers. Looks great and you can make your activities with comfort and good looking.
Generated	I really love them! They are perfect and very comfortable. They look good and I will recommend this item.
Case 2: Health Care	
Source	I take vitamins three times a day. This product fits the bill perfectly when I travel. I like it so much! I bought two of them. I first went to a chain pharmacy and discount store but their offerings were too expensive and were too small. This product was far cheaper and works for me .
Generated	I love these vitamins. Easy small pills to take and a good quality product. Great price on Amazon! Will order again.
Case 3: Amazon - Sports	
Source	It is very easy to use good equipment. It came with in a short time. Realy happy to have it.
Generated	It's great and easy and it is not too big to install. It will work. I have a good purchase.
Case 4: Amazon - Video Games	
Source	This game is awesome! Beautiful landscapes and adventures! It puts you right in the fantasy world. Very addictive!
Generated	Great graphics! I would love playing the series and it was very well! Great price and I recommend this.
Case 5: Yelp	
Source	Always great sushi no matter if you are doing AYCE or individual orders. Always a wait for this place.
Generated	I enjoy the panko a lot. I always love family style ramen but always looking for a spot.

5 Related Work

As we mentioned in Section 1, our task is a variant to common link prediction problems.

Most link prediction models study the task of recommender system. For example, in social networks (*e.g.* Facebook, Twitter, Instagram, *etc.*) we can recommend new friends based on current relationships [1]. In some domain specific social networks like academic social network, link prediction can help find domain experts or co-authors [20, 28]. Large online shopping platforms could form a behaviour network based on user behaviours as well, and link prediction models can recommend personalized items for individual customers [14]. In the advertisement recommendation area, Wang et al. propose a framework SHINE to predict possibly existing sentiment links in the presence of heterogeneous information, which could generate sentiment tags between users and advertisers [27]. Link prediction also predicts the missing links on an incomplete observed networks [15]. For dynamic networks such as P2P lending networks, it can infer the future newly added links and evolution process [31] as well. In other domains like bioinformatics area, there exists networks such as gene expression networks. We can use link prediction to predict new protein-protein interactions [2], which has great significance to human life health and disease treatment. All the problems mentioned above are numerical prediction or tag labelling tasks on links, and they cannot be applied to our problem.

Statistical language modeling (SLM) is an important research topic all the time in natural language processing field. It is commonly used in speech recognition [9], machine translation [4], handwriting recognition [30] and other applications. N-gram model [7] was the earliest technique for language modeling. It models the probability of a word appears next given a sequence of context words. Later, models based on decision tree [22] and maximum entropy techniques [6] appeared. They build models with consideration of other features such as part of speech tags. Bengio et al. [5] first applied neural networks to language model domain in 2003. He proposed a Feed-forward Neural Network (FNN) which can predict the next word given the fixed size of the previous sequence of words and learn the word representation in the vocabulary at the same time. This model greatly improves the performance of speech recognition [23]. However, previous models still have the shortcoming that long range dependencies cannot be captured due to the fixed context constrain. Mikolov et al. proposed a Recurrent Neural Network (RNN) which allows context information passing all through [16, 17]. To address the issue of gradient vanishing, a variant of RNN called Long-Short Term Memory (LSTM) [25] is presented later. In recent years, generating natural language text for image (Image Caption) has been researched in depth. Karpathy et al. proposed an image caption model and it builds a multimodel RNN that considers image features [13]. Anderson et al. propose a combined bottom-up and top-down attention mechanism that can enable deeper image understanding for image caption and visual question anwsering [3]. However, image caption models cannot be migrated to network because they have different structures. Yao et al. also study the problem of generating fake reviews for specific restaurant automatically [29]. Their reviews cannot reflect the personalization because they do not concern about the network structure even the customers.

6 Conclusion

In this paper, we propose a novel application of language modelling on network structure, which is generating text on each edge of a network. It is a challenging work because text on edges encodes not only the relationship between two network entities but also rich semantic information. We build an edge labelling language model Net2Text that considers both network structure information and text on edges, to generate personalized reviews for users. The performance of Net2Text is demonstrated in experiments on real world datasets, showing that our model performs better than other baselines, and is able to generate reasonable reviews for customers.

References

1. Aiello, L.M., Barrat, A., Schifanella, R., Cattuto, C., Markines, B., Menczer, F.: Friendship prediction and homophily in social media. ACM Transactions on the Web **6**(2), 9 (2012)
2. Almansoori, W., Gao, S., Jarada, T.N., Elsheikh, A.M., Murshed, A.N., Jida, J., Alhajj, R., Rokne, J.: Link prediction and classification in social networks and its application in healthcare and systems biology. Network Modeling Analysis in Health Informatics and Bioinformatics **1**(1-2), 27–36 (2012)
3. Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., Zhang, L.: Bottom-up and top-down attention for image captioning and visual question answering. In: CVPR'18. p. 6 (2018)
4. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: ICLR'15 (2015)
5. Bengio, Y., Ducharme, R., Vincent, P., Jauvin, C.: A neural probabilistic language model. Journal of Machine Learning Research **3**(Feb), 1137–1155 (2003)
6. Berger, A.L., Pietra, V.J.D., Pietra, S.A.D.: A maximum entropy approach to natural language processing. Computational linguistics **22**(1), 39–71 (1996)
7. Brown, P.F., Desouza, P.V., Mercer, R.L., Pietra, V.J.D., Lai, J.C.: Class-based n-gram models of natural language. Computational linguistics **18**(4), 467–479 (1992)
8. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555 (2014)
9. Graves, A., Mohamed, A.r., Hinton, G.: Speech recognition with deep recurrent neural networks. In: ICASSP'13. pp. 6645–6649. IEEE (2013)
10. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: KDD'16. pp. 855–864. ACM (2016)
11. He, R., McAuley, J.: Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In: WWW'16. pp. 507–517. International World Wide Web Conferences Steering Committee (2016)
12. Herlocker, J.L., Konstan, J.A., Terveen, L.G., Riedl, J.T.: Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems **22**(1), 5–53 (2004)
13. Karpathy, A., Fei-Fei, L.: Deep visual-semantic alignments for generating image descriptions. In: CVPR'15. pp. 3128–3137 (2015)
14. Li, X., Chen, H.: Recommendation as link prediction in bipartite graphs: A graph kernel-based machine learning approach. Decision Support Systems **54**(2), 880–890 (2013)
15. Marchette, D.J., Priebe, C.E.: Predicting unobserved links in incompletely observed networks. Computational Statistics & Data Analysis **52**(3), 1373–1386 (2008)
16. Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., Khudanpur, S.: Recurrent neural network based language model. In: InterSpeech'10. vol. 2, p. 3 (2010)
17. Mikolov, T., Kombrink, S., Burget, L., Černockỳ, J., Khudanpur, S.: Extensions of recurrent neural network language model. In: ICASSP'11. pp. 5528–5531. IEEE (2011)
18. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS'13. pp. 3111–3119 (2013)
19. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: Bleu: a method for automatic evaluation of machine translation. In: ACL'02. pp. 311–318. Association for Computational Linguistics (2002)

20. Pavlov, M., Ichise, R.: Finding experts by link prediction in co-authorship networks. In: FEWS'07. pp. 42–55 (2007)
21. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: KDD'14. pp. 701–710. ACM (2014)
22. Potamianos, G., Jelinek, F.: A study of n-gram and decision tree letter language modeling methods. Speech Communication **24**(3), 171–192 (1998)
23. Schwenk, H., Gauvain, J.L.: Training neural network language models on very large corpora. In: HLT/EMNLP'05. pp. 201–208. Association for Computational Linguistics (2005)
24. Shore, J., Johnson, R.: Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy. TIT **26**(1), 26–37 (1980)
25. Sundermeyer, M., Schlüter, R., Ney, H.: Lstm neural networks for language modeling. In: InterSpeech'12 (2012)
26. Sutskever, I., Martens, J., Hinton, G.E.: Generating text with recurrent neural networks. In: ICML'11. pp. 1017–1024 (2011)
27. Wang, H., Zhang, F., Hou, M., Xie, X., Guo, M., Liu, Q.: Shine: signed heterogeneous information network embedding for sentiment link prediction. In: WSDM'18. pp. 592–600. ACM (2018)
28. Wohlfarth, T., Ichise, R.: Semantic and event-based approach for link prediction. In: PAKM'08. pp. 50–61. Springer (2008)
29. Yao, Y., Viswanath, B., Cryan, J., Zheng, H., Zhao, B.Y.: Automated crowdturfing attacks and defenses in online review systems. In: CCS'17. pp. 1143–1158. ACM (2017)
30. Zamora-Martinez, F., Frinken, V., España-Boquera, S., Castro-Bleda, M.J., Fischer, A., Bunke, H.: Neural network language models for off-line handwriting recognition. Pattern Recognition **47**(4), 1642–1652 (2014)
31. Zhang, Y., Xiong, Y., Kong, X., Zhu, Y.: Netcycle: Collective evolution inference in heterogeneous information networks. In: KDD'16. pp. 1365–1374 (2016)