Recurrent Neural Network Embedding for Knowledge-base Completion

Yuxing Zhang, Andrew ID: yuxingz

1 Introduction

Knowledge can often be represented using entities connected by relations. For example, the fact that tennis ball is round can be represented as "TennisBall HasShape Round", where a "TennisBall" is one entity, "HasShape" is a relation and "Round" is another entity. A knowledge base is a way to store such structured information, a knowledge base stores triples of the "an entity-relation-an entity" form, and a real world knowledge base often has millions or billions of such triples. There are several well-known knowledge bases including FreeBase [1], WordNet [2], YAGO [3], etc. They are important in fields like reasoning and question answering; for instance if one asks "what is the shape of a tennis ball", we can search the knowledge base for the triple as "TennisBall HasShape Round" and output "round" as the answer.

This approach to search for the exact triple in the knowledge base requires that triple to exist in the knowledge base. However it is often the case that a triple is missing due to the huge number of combinations between different entities and relations and the limited amount of information stored in the knowledge base. This is indeed one major problem with the real world knowledge bases, that their coverage is incomplete despite the fact that there are an enormous number of entries stored these knowledge bases. Such incompleteness is because for some entities in the knowledge base, we have part of their information but not all. Yet intuitively, we can still get some information from some similar triples that are in the knowledge base. For instance, "TennisBall HasShape Round" and "Ping-pongBall IsSimilarTo TennisBall" are both in the knowledge base, then intuitively, this implies that it is likely that ping-pong ball also has shape round. Another example would be that given "Water HasForm Liquid", "Water IsContained InABottle" and "OliveOil HasForm Liquid", we want to conclude that olive oil is also contained in a bottle, and then add this information to the knowledge base.

Another form of incompleteness is that some entities only appear in the knowledge base infrequently or not at all, and such sparsity in the knowledge base is actually very common. New products and some recent events are such examples. In these cases, we have little or no information of the entities, but we still want to do some kind of reasoning on them based on the information we have (usually the name of the entity). For example, if we have never heard of "iBottle" but we know that "Bottle" can be used to contain water, then maybe we can also use "iBottle" to do that. Here the motivation is that the name of an entity is often informative, and it would be helpful to somehow exploit this fact.

Knowledge base completion is the task of inferring the missing triples from existing triples in the knowledge base. In general, this task requires a good modeling of the multi-relational data. One example of such a model is the exact triples in the knowledge base, which suffers from the incompleteness issue as we have mentioned. The main difference between modeling multi-relational data and single-relational data is that one can easily compute a good representation for single-relational data in an ad-hoc fashion since there is only one relation being considered, while computing a representation for an entity such that the representation is good for all the relations is hard. The fact that people want to perform complicated reasoning over the knowledge base makes this task even more challenging. A similar task is collaborative filtering, where one tries to predict whether a user is interested in an item or not, which is to predict this particular relation. The difference between col-

laborative filtering and knowledge base completion is that knowledge base completion often deals with lots of relations instead of just one.

There are three major approaches to build a model for multi-relational data according to [4]: using a probabilistic graphical model as in [5], using a random walk to compute scores for different paths as in [6], or using embeddings to model the structures in the knowledge base. Recently, researchers have mainly focused on embedding based methods for knowledge base completion. Such methods map each entity to a vector in a low dimensional real space, and treat relations as transformations between entities. The main difference between these works is the way they represent the entities and relations. [7] uses existing word vectors to compute vectors of entities by computing the average among the word vectors, while [8] computes embeddings for both entities and relations directly. These methods differ in various aspects including the objective and the way to connect entities with relations, but they share a general framework: one entity can be approximately computed based on the relation and the other entity, so knowledge base completion can be done by choosing the entity that is closest to the entity vector, computed based on the similarity metric defined.

In this paper, we experiment with embedding models for knowledge base completion and propose a new embedding model to solve the entity sparsity problem. More specifically, we experiment with TransE [8] on both the FB15k data set and the prescription data set (which we will discuss in detail later), then describe our new model, which is based on a bi-directional GRU recurrent neural network and entity embedding. We compare these two models via an asymmetric knowledge base completion task on the prescription data set. The result shows that our proposed model outperforms the TransE model when there are some sparsity in the data.

2 Related Work

2.1 Word Embeddings

In 2006, Bengio et al. proposed a model for word representation in [9]. Prior to this work, people generally use the n-gram representation in statistical language models, which suffers from the curse of dimensionality because the number of such n-grams is approximately V^n where V is the vocabulary size. In Bengio et al.'s model, words are represented as vectors in a real vector space, and the distance between vectors representing different words measures their similarity. In this way, one sentence from the training data could provide much more information than the n-gram model since it also informs the model about similar sentences due to the continuity in the real vector space. Specifically they use a $V \times D$ matrix for the word vectors and use a neural network with a soft max layer to compute the probability of a fixed length sub-sentence. They train the model using gradient descent on both the word vectors and the neural network parameters with log-likelihood as the objective function and their model has a significantly better performance than the n-grams model in terms of perplexity. Such methods that represent words using real vectors are often called word embedding.

In 2013, Mikolov et al. introduced two novel architectures for computing real vector word representations using neural networks in [10], aiming at reducing the computational cost when training the model so it can handle much larger data set in a reasonable amount of time. The most expensive computation in the previous neural network language model is the non-linear hidden layer in the network. In their models, they remove this hidden layer from the neural network and share the projection layer between all inputs. The first architecture is called the continuous bag-of-word model, which predicts a word directly from its context within certain range. The second architecture does the opposite, predicting the context of a given word. They constructed a Semantic-Syntactic Word Relationship test set and ran their model on the test set. The result shows that their model outperforms the previous models both in accuracy and efficiency.

Later in [11], Mikolov et al. further improved the continuous Skip-gram model (which allows one to skip some words in between when constructing n grams) with two techniques, subsampling the frequent words and hierarchical softmax (or negative sampling). Using subsampling for the frequent words in the corpus, they significantly speed up the training process without compromising the quality of the model too much, since the frequent words are not carrying much information after they have appeared enough times. The second method to increase the training efficiency is to reduce the time to compute the full softmax function. One way to do this is to use a hierarchical softmax

function, which builds a binary tree on the probabilities over all the words. This will reduce the time for computing a probability from linear in the vocabulary size to log of the vocabulary size. Another way is to use negative sampling to replace the softmax function completely, that is to use some sample words to compute the loss function and optimize a surrogate objective function iteratively.

2.2 Knowledge Base Completion with Embeddings

To improve the result of performing complex inferences on a large scale knowledge base, Lao et al. proposed in [6] a path ranking algorithm [12] based learning procedure that can determine whether two entities are connected by a relation in the knowledge base. The path ranking algorithm computes a probability for a path that we will go from one end of the path to the other end following the same type of path (relation). Then to make a prediction on a knowledge base of whether one entity is connected by a relation (or relation chain) to another entity, the algorithm will run path ranking on a large set of proposed paths which are of bounded lengths, and are constructed with certain types of relations. These proposed paths can be treated as a set of experts, then once we have a score for each of them, the algorithm uses a logistic regression on these scores to decide whether the relation is valid or not. They run the new algorithm on the knowledge base extracted by NELL [13] and shows that the new algorithm outperforms the Horn-clause learning method and significantly improve the precision at rank 100.

In 2011, Bordes proposed a method to model multi-relational semantics in [14]. Inspired by word embeddings, Bordes et al. introduced TransE in [8], which is a method to model the relations between different entities in a knowledge base as translations in a low dimensional embedding space. Instead of using complex models with greater expressiveness, they model both the entities and the relations in the knowledge base as real vectors in the same embedding space, and represent the relations as translations from an entity to another. More specifically, for a triple (e_1, r, e_2) in the knowledge base, they treat it as $e_1 + r = e_2$. Then to train the model, they minimize the loss function defined as the sum of error on each triple (e_1, r, e_2) from the knowledge base which is $d(e_1 + r, e_2)$ minus the error of some randomly generated triples $d(e_1' + r', e_2')$. The intuition behind the loss function is that they want the correct right hand side entity to be close to the translation from the left hand side entity and relation while the random noise right hand side entity to be far away from the translation of a random left hand side entity. Their model achieves a higher accuracy on both the WordNet and the FreeBase data set than the other previous models.

2.3 Recurrent Neural Networks

Cho et al. proposed a gated recursive convolutional neural network model [15] for neural machine translation in 2014. The new model differs from the vanilla recursive neural network in the way it computes the transition on the hidden units between different steps. In a vanilla RNN, let $\mathbf{h}^{(t)}$ be the hidden unit at time t and let $\mathbf{x}_t$ be the t-th element in the input, then $\mathbf{h}^{(t)} = f(\mathbf{h}^{(t-1)}, x_t)$, where f is the activation function often chosen to be `tanh`. It is widely known that a vanilla RNN cannot keep long term memories due to the vanishing gradient issue (or exploding). In the new model, they add multiple gates to the computation of the new hidden states, which they call a gated recurrent unit (GRU). Now the j-th hidden unit is computed by $h_j^{(t)} = \omega_c \tilde{h}_j^{(t)} + \omega_l h_{j-1}^{(t-1)} + \omega_r h_j^{(t-1)}$, where $\tilde{h}_j^{(t)}$ is the usual value of hidden unit after activation. Here these three values are gated by $\omega_c, \omega_l, \omega_r$, and this structure enables the model to learn more complex functions over different input data without having the gradient issue, since the additive component in the hidden unit update can serve as a short cut for back propagating the gradient. They compare this model with the RNN Encoder-Decoder model [16] and evaluate both models on translation tasks between English and French. The result shows that the new model outperforms the RNN Encoder-Decoder model and has a much lower error rate on long sentences.

Chung et al. compared the GRU RNN presented in [15] and the LSTM RNN in [17] with the vanilla RNN model with tanh, and experimented with all three models on sequence modeling tasks including music modeling and speech signal modeling. The result shows that all three models have similar performance on the music modeling task while the first two models (gated RNN) outperform the vanilla RNN on the speech signal modeling task in terms of both the accuracy and the convergence rate. In the speech modeling task, the GRU RNN sometimes performs better than the LSTM RNN

and sometimes worse, which implies that the choice between LSTM and GRU depends on the actual task and data set being used.

Dos Santos et al. proposed a deep neural network model to learn character-level word embeddings in 2014 [18]. Their model exploits word morphology and outperforms existing models at part-of-speech tagging tasks in both English (on the Penn Treebank WSJ corpus) and Portuguese (on the Mac-Morpho corpus).

Inspired by the GRU and the character-level word embedding ideas, Dhingra et al. proposed a character based bi-directional GRU RNN model for sentence classification in 2016 [19]. The idea behind this work is that in most text data there will be many infrequent words, so if we model the text by treating a word as the basic unit, then most information we have is about the frequent words. Another concern is memory usage. Finally, we have to map some infrequent words to a single token which is obviously undesirable: as a result, we wouldn't know much about those infrequent words, and we can hardly build a good model that generalizes. Yet recent work [20] has shown that LSTM RNN on characters can produce informative word representation, and inspired by that, this work builds a bi-directional GRU RNN on characters to classify text posted on Twitter. The model has one forward GRU and one backward GRU, processing the input in opposite order. Then the output of the last forward hidden unit and the output of the first backward hidden unit are combined by a fully connected layer to compute the embedding for the tweet, followed by a softmax layer to calculate the probability of belonging to certain class. Here they predict the hashtags of a post. They compare the character level model with the word level model on the tweet classification task; the result shows that the character level model has a slightly better performance on the full data set while it has a much better performance on the rare word test set. This suggests that a character level model is more favorable when the data is sparse.

In our work, we want to extend knowledge base embedding with character level embeddings and make good predictions on sparse data.

3 Data & Problem Formulation

3.1 Data

3.1.1 FB15k Data Set

There are two data sets we used in our work, one is the FB15k data set[1]. It contains approximately 600,000 triples extracted from FreeBase [1]. There are around 15k unique entities in this data set, and this is also why the data set is called FB15k. It is often used for knowledge base completion tasks and this is the data set used with TransE [8]. Some statistics of the FB15k data set computed by [21] are shown in Table 2:

3.1.2 Prescription Data Set

The second data set we used is a prescription data set obtained from DailyMed, where the detailed information about a lot of common medicines is listed. Each webpage contains one medicine, so we downloaded the XML file for every medicine. Common attributes of a medicine from the XML file include the name, manufacturer, appearance (color, shape, size), package, flavor, dosage form, etc.

To retrieve useful information from the file, we use dom4j XPath[2] to parse the XML tree and extract the relevant tags. We generate a triple for each unique attribute of a medicine. The left hand side of each triple will be the medicine name, and the relation and the right hand side will be determined by the attribute of that medicine. For example, if there is a tag in the sample XML file saying that medicine Tamsulosin Hydrochloride is consumed orally, then we will create the triple (Tamsulosin Hydrochloride, Consumed in, Oral). Table 1 shows examples of triples we extract from the XML files.

There are about 13,000 XML files in our data set, and after converting them into triple form, we have 360,000 such triples; but there are many duplicated triples and unrelated attributes. For example,

[1] https://everest.hds.utc.fr/doku.php?id=en:transe

[2] http://dom4j.sourceforge.net/dom4j-1.6.1

Left Entity	Relation	Right Entity
Prescription data set extracted from structured XML file		
Vesicare	Contains Ingredient	Solifenacin Succinate
Testopel	Contains Ingredient	Testosterone
Kristalose	Contains Ingredient	Lactulose
methysergide Maleate	Consumed In	Oral
Bexarotene	Consumed In	Topical
Pralidoxime Chloride	Consumed In	Intramuscular
Depen	Dosage Form	Tablet
Kristalose	Dosage Form	Powder
Topex	Dosage Form	Dentifrice
Benzo-jel	Package Form	Bottle
Testopel	Package Form	Ampule
Topex	Package Form	Jar
Unstructured data set extracted from DailyMed text		
Misoprostol	Side Effects	Tremor
Hydroquinone	Side Effects	Exogenous Ochronosis
Prednisone	Conditions To Prevent	Corticoid Therapy
Ziprasidone Hcl	Conditions To Prevent	Mean Weight Gain
Amoxicillin	Used To Treat	Bacterial Infections
Insulin Aspart	Used To Treat	Upper Arm

Table 1: Examples for triples extracted form prescription data set and unstructured data set

one of the attributes is a unique identifier for that specific medicine, which is a random string, and clearly it makes no sense to predict a unique identifier in an knowledge base completion task, nor does modeling this attribute tell us anything about the medicine. Other unrelated attributes include manufacturer, color, imprint, approval status, etc. After removing redundant data and keeping only relevant relations (Ingredient, Package Form, Way of Consuming, Dosage Form), we end up with around 54,000 unique triples, and we call this data set Prescription Unique.

Besides this, we also build another data set on the prescription data set called Prescription Frequent. Here we take the Prescription Unique data set and remove rare right hand side entities (entities that appear less than 5 times) along with triples containing these entities.

Table 2 shows some statistics of the FB15k data set and two prescription data sets, where PRUniq is for Prescription Unique and PRFreq is for Prescription Frequent:

Name	# Total Ent	# LHS Ent	# Relations	# RHS Ent	# Triples
FB15k	14,951	14,834	1,345	14,903	592,213
PRUniq	6,180	3,083	4	3,097	54,239
PRFreq	3,873	3,083	4	790	43,929

Table 2: Statistics of FB15k and Prescription Data Set

The prescription data set is quite different from the FB15k data set:

- Symmetry: entities in FB15k data set are symmetric, in the sense that most of the entities appear both in the left hand side and right hand side in different triples. Yet entities in prescription data set are asymmetric: all the left hand side entities are medicine names, all the right hand side entities are attributes, and they don't overlap, so this is actually a bipartite graph. This difference also influences the way we do knowledge base completion: we can predict both left hand side and right hand side for the FB15k data set, but we will only make predictions on the right hand side of the prescription data set, which is to predict the properties of a medicine.
- Density: the knowledge graph of FB15k data set is well connected. The average degree of an entity in this data set is around 40. While the prescription data set is relatively sparse, if we consider each medicine, it only appears on the left hand side with an average degree

of 14 and there are many medicines with out degree no more than 5, meaning that these medicines don't appear very often in the data set. As a result, if we randomly partition the data set into training set, validation set and test set, there might be some medicines that never appear in the training set. Along with the bipartite property of the knowledge graph from the prescription data set, we can see that there is much less information we can learn for each entity in this data set than in the FB15k data set.

Besides the prescription data set, we also have another data set extracted from the text paragraphs in DailyMed using the method proposed in [22], where Bing et al. improved the quality of distant labeling using label propagation on graphs constructed entity mentions. Here each triple in the data set is given a score between 0 and 1 indicating how confident we are about the quality of the triple, if the score is high, say 0.95, then we are very confident that the two entities are connected by the given relation. We keep only the high score triples (score ≥ 0.9) that have at least one entity appearing in the prescription data set, meaning that we will discard a triple if it has a low score or none of its entities appear in the prescription data set; this gives a data set with 940,735 triples. Since this data set is extracted automatically from unstructured text corpus, we are only considering using it to augment the prescription data set, and we call it the unstructured data set. See Table 1 for some examples of triples from this data set.

Figure 1 is a visualization of the prescription frequent data set, where each node in the graph represents an entity and each edge between two entities represents a relation connecting them.

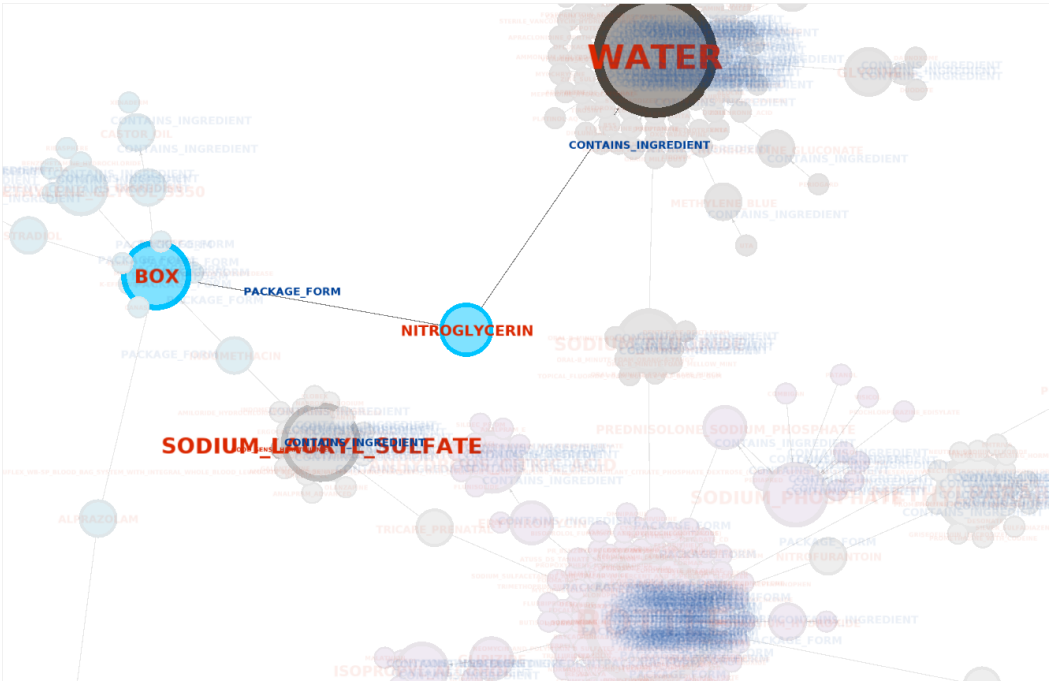

Figure 1: Knowledge Graph Visualization for the prescription data set, where the size of each node is proportional to the degree of that node. The highlighted medicine Nitroglycerin has two attributes in this example. One is that Nitroglycerin has package form box, and the other one is that Nitroglycerin contains water as an ingredient.

3.2 Problem Formulation

The problem we are considering is knowledge base completion. Formally speaking, assume the data set contains triples of the form (h, l, t), then given one entity h or t and the relation l, we want to predict the other entity t or h. For the prescription data set, we will only predict the right hand side entity t.

4 Approach

4.1 TransE Model

The first model we consider is the TransE model proposed in [8], which maps each entity or relation to a real valued vector in the low dimensional embedding space. TransE treats relations as translations between different entities in the embedding space. More specifically, given left hand side entity h, right hand side entity t and relation l, such that (h, l, t) is a triple in the knowledge base, the model tries to satisfy $\boldsymbol{h} + \boldsymbol{l} \approx \boldsymbol{t}$, where the boldface characters are the embedding vectors for each entity or relation.

Following the idea above, the loss function of the TransE model is defined as

$$\mathcal{L} = \sum_{(h,l,t)\in S} \sum_{(h',l,t')\in S'} \Big[\gamma + d(\boldsymbol{h} + \boldsymbol{l}, \boldsymbol{t}) - d(\boldsymbol{h}' + \boldsymbol{l}, \boldsymbol{t}')\Big]_+$$

Where $d(\boldsymbol{h} + \boldsymbol{l}, \boldsymbol{t})$ is the distance between $\boldsymbol{h} + \boldsymbol{l}$ and $\boldsymbol{t}$, usually taken to be the Euclidean distance. Here S is the set of true triples from the knowledge base and S' is the set of triples such that for each triple $(h, l, t) \in S$, we replace either h or t with a random entity: so it is the set of triples of the form (h', l, t) and (h, l, t') such that $(h, l, t) \in S$. The reason for adding the randomly sampled triples to the loss function is that we want the distance function to have small values for true triples and large values for random triples; otherwise setting everything to 0 will be a trivial solution to the minimization problem.

4.2 Bi-GRU RNN Embedding Model

Motivated by Tweet2vec [19], we propose a new model to solve the sparsity problem in the canonical embedding model. Our model incorporates the Bi-GRU RNN model into the entity embedding framework, with the embedding look up table for the left hand side entities replaced with a Bi-GRU recurrent neural network, which aims to exploit the regularity of medicine names in the character level and generate reasonably good embedding for rare or previously unseen entities.

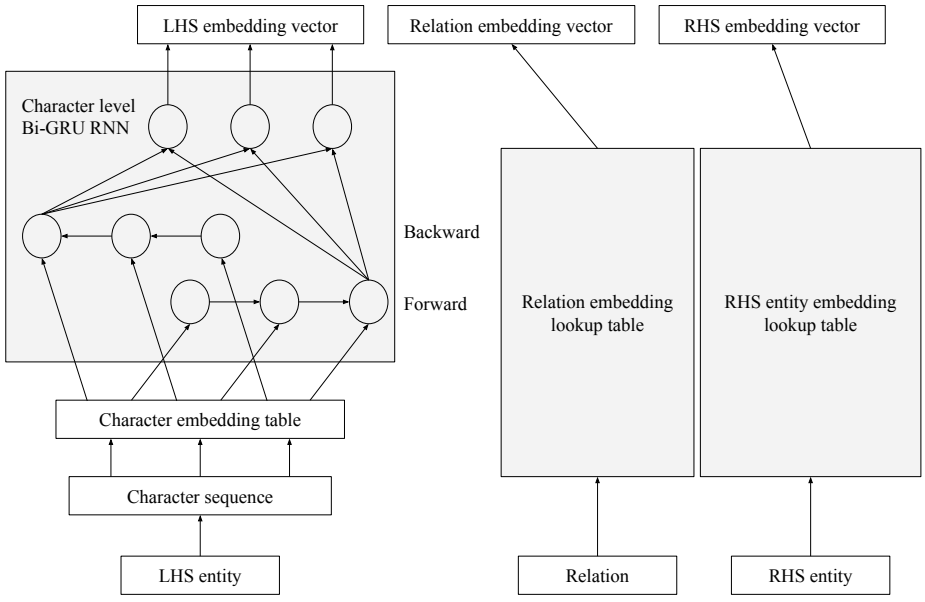

Figure 2: Bi-GRU RNN Embedding model

Figure 2 shows the high level architecture of our RNN embedding model, which shares the general framework as the TransE model [8]. The input triple is split into left hand side head h, relation l and

right hand side tail t, where each of them is a word (or words concatenated by underscore), the head entity h goes into the Bi-GRU recurrent neural network [19] and then converted into embedding vector $\boldsymbol{h}$ which we will discuss in detail later.

The relation l and the tail entity t are converted into one-hot vectors $v_e \in \{0,1\}^{|V_e|}, e \in \{l,t\}$ using dictionaries V_l, V_t built from training data, with only a single 1 in the vector indicating which entity or relation it is. The embedding vectors for l and t are $\boldsymbol{l} = v_l^T \boldsymbol{E}_l, \boldsymbol{t} = v_t^T \boldsymbol{E}_t$, where $\boldsymbol{E}_e \in \mathbb{R}^{|V_e| \times D}, e \in \{l,t\}$ are the embedding lookup tables for relations and right hand side entities respectively, D is the dimension for the entity embedding space. The triple $(\boldsymbol{h}, \boldsymbol{l}, \boldsymbol{t})$ is the embedding vector triple for the input (h, l, t).

The left hand side entity l is split into a sequence of characters and converted to one-hot vectors $c_1, \cdots, c_m \in \{0,1\}^{|V_c|}$, with a single 1 in each vector indicating which character it represents, where V_c is the character vocabulary. Note here we can build this vocabulary for all possible characters if we are only considering English alphabets, numbers and some symbols (underscore or comma), as opposed to the situation where we generally cannot build a vocabulary for each word in a corpus. We obtain the character vector $x_i = c_i^T \boldsymbol{E}_c$ by taking the corresponding component from the character embedding matrix $\boldsymbol{E}_c \in \mathbb{R}^{|V_c| \times D_c}$. The sequence of character vectors $\{x_i\}_{i=1}^m$ then goes into the Bi-GRU recurrent neural network, with two layers of gated recurrent unit (GRU shown in 4). One layer processes the sequence in the normal order and the other layer processes in the reversed order, shown in 3.

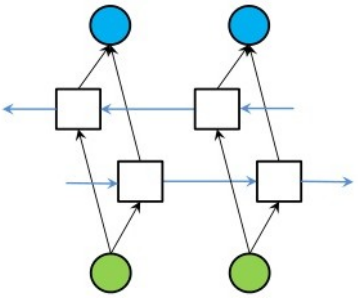

Figure 3: Bi-directional RNN, figure from here

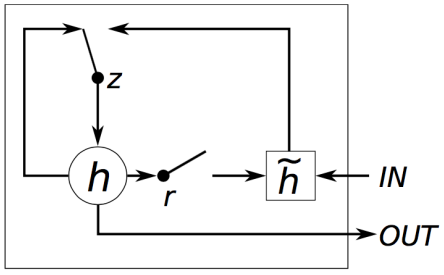

Figure 4: GRU, figure from [16]

The forward GRU layer and backward GRU layer have the same structure and they start from the same state $h_0 \in \mathbb{R}^{D_h}$ when processing input sequence $\{x_i\}_{i=1}^m$. On input x_t, these two layers update the hidden state h as follows [19]:

$$\begin{aligned} r_t &= \sigma(W_r x_t + U_r h_{t-1}) \\ z_t &= \sigma(W_z x_t + U_z h_{t-1}) \\ \tilde{h}_t &= tanh(W_h x_t + U_h(r_t \odot h_{t-1})) \\ h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t. \end{aligned}$$

Here r_t and z_t are the gates controlling how much we want the values of h_{t-1} and $\tilde{h}_t$ to go into the next state h_t, and $W_r, W_z, W_h \in \mathbb{R}^{D_h \times D_c}, U_r, U_z, U_h \in \mathbb{R}^{D_h \times D_h}$ are the parameters for the GRU. After going through these two layers, we take the value of the hidden unit in the forward and backward GRU layers h_f^*, h_b^*, and apply a linear transformation to get the final embedding vector $\boldsymbol{h}$ for the input sequence.

$$\boldsymbol{h} = W_f h_f^* + W_b h_b^* + c.$$

Here $W_f, W_b \in \mathbb{R}^{D_h \times D}, c \in \mathbb{R}^D$. Now we have the left hand side entity embedding $\boldsymbol{h}$ from RNN, relation $\boldsymbol{l}$ and right hand side entity $\boldsymbol{t}$ from lookup tables, and we define the loss function in a way similar to TransE [8] as follows:

$$\mathcal{L} = \sum_{(h,l,t)\in S} \sum_{(h',l,t')\in S'} \Big[\gamma + d(\boldsymbol{h}+\boldsymbol{l},\boldsymbol{t}) - d(\boldsymbol{h}'+\boldsymbol{l},\boldsymbol{t}')\Big]_+ + \lambda\big\|\Theta\big\|^2.$$

Here we add a regularization term and Θ denotes all the parameters in the RNN. To make a prediction on the right hand side entity t given the left hand side entity h and relation l, we will compute the embeddings for both h and l, then output the prediction as

$$t = \arg\min_{t\in S_{\text{rhs}}} d(\boldsymbol{h}+\boldsymbol{l},\boldsymbol{t})$$

.

We also propose a variant of this Bi-GRU RNN embedding model, called the hybrid model, which combines the RNN embedding with a lookup table for the left hand side entities. More precisely, the only difference from the RNN embedding model is that the hybrid model also keeps a lookup table for the left hand side entities, and combines the output embedding vector $\boldsymbol{h}_{\text{rnn}}$ from the RNN with the embedding from the lookup table $\boldsymbol{h}_{\text{emb}}$ by applying a linear transformation on the concatenation of the two vectors to get $\boldsymbol{h} = W_{hy}(\boldsymbol{h}_{\text{rnn}}^T, \boldsymbol{h}_{\text{emb}}^T)^T + b$, where $W_{hy} \in \mathbb{R}^{D\times 2D}, b \in \mathbb{R}^D$. The loss function is the same as the RNN model.

5 Experiments & Results

5.1 TransE on Prescription Frequent Data Set

We firstly experiment with the public available TransE code [3] written in Python. We train TransE on the FB15k data set and get the following result in Table 3:

Data Set	Mean Rank	Hits@10	Hits@1
FB15k	160.86706	45.088%	15.033%
PRFreq	45.14663	54.224%	16.003%
PRFreq$^+$	2449.66003	27.988%	7.212%

Table 3: TransE on FB15k and prescription data set

which matches the result shown in the paper [8]. Here mean rank is the average ranking of the true right hand side entities according to the score computed and hit@k is the proportion of true right hand side entities ranking in the top k in the score table.

When we experiment with TransE on the prescription data set, we randomly partition the triples into training set, validation set and test set. For the test set, we remove only the right hand side of each triple, since the left hand side is always the name of the medicine. Intuitively speaking, we want to be able to answer questions concerning some properties of a medicine, such as the ingredients or package information.

In our random partition, some entities from the test set never appear in the training set, thus if we train TransE on the training set, the embeddings of these entities are undefined. This is mainly due to the sparsity of the data we mentioned before, since some entities only appear once or twice in the whole data set. So we use the Prescription Frequent data set here, which is the data set after removing infrequent entities (appearing no more than 5 times in the data set).

We train the TransE model on the Prescription Frequent data set, and use the trained model to predict the right hand side for each data point in the test set. Similar to the evaluation criteria in TransE [8], for each test triple (h, l, t), we compute the embeddings for h and l, along with the scores for all right hand side entities in the vocabulary, where the score is defined as $s_{t_i} = -d(\boldsymbol{h}+\boldsymbol{l},\boldsymbol{t}_i)$. We report the mean rank, hit@10 and hit@1 as the result.

We also tried to augment the Prescription Frequent data set using the unstructured data set extracted from DailyMed text paragraphs mentioned earlier, to see if the unstructured data can serve as an

[3] https://github.com/glorotxa/SME

additional source of information for building the embedding space. More precisely, we add high score triples from the unstructured data set to the Prescription Frequent training set, train the TransE model on the augmented data set, retrieving the embeddings only for the entities from the Prescription Frequent data set, and test on the original test set using the new embeddings. The results are listed in Table 3, where PRFreq stands for Prescription Frequent data set and PRFreq$^+$ stands for Prescription Frequent data set augmented by unstructured data set.

In both experiments, we are using $D = 50$ as the embedding space dimension (we also experimented with $D = 25$ on TransE for fair comparison since we want both models to have similar sizes), and L^2 distance as the distance metric. We set the margin $\gamma = 0.5$, the learning rate for embedding matrix $lremb = 0.01$, and random seed $s = 123$. The training is performed with mini-batch gradient descent with 100 batches, initial values of the embedding matrix are sampled uniformly from $(-\frac{6}{\sqrt{D}}, \frac{6}{\sqrt{D}})$. Notice here we use only one embedding matrix for all entities without differentiating which side they are from.

This result has several implications:

1. The TransE model has a reasonably good performance on the PRFreq data set, mainly because we are using this refined data set where infrequent entities are discarded. Also this result shows there is structure in the prescription data set, meaning that we can indeed learn from the data and make predictions on the attributes of the medicines.
2. The result for the augmented data set is not as good as using only the prescription data set. The unstructured data set is not so relevant to the prescription data set in the sense that it doesn't have lots of overlapping entities or relations. Therefore adding the unstructured data set into the training set introduces lots of noise and restrictions to the model, because we have to adjust the embeddings to also fit the extra data, which results in worse embedding for the original data set.

5.2 Bi-GRU RNN Embedding on Prescription Unique Data Set

We implemented the RNN embedding model in Python using Theano [23] with the Lasagne library [24] for building neural networks on Theano. Our implementation was based on the Tweet2Vec code in [19]. To compare our RNN embedding model with TransE, we also implemented TransE with Lasagne on Theano, but here we use three embedding matrices, one for left hand side entities, one for relations and one for right hand side entities. Here we modify the TransE model such that the model maps all the unseen entities to one special vector, so that we can make comparisons with our RNN model.

For the RNN embedding model, we take the character embedding dimension to be $D_c = 8$, and the entity / relation embedding dimension to be $D = 50$. The training here is done using Nesterov's momentum accelerated mini-batch gradient descent with batch size 100. The initial learning rate is 0.01 and momentum is set to be 0.9. We also update the learning rate (divided by 2) when the learning process slows down. Parameters are initialized by random sampling from a normal distribution with mean $\mu = 0$ and variance $\sigma = 0.01$. For the TransE model, we also take $D = 50$, and use the same training procedure. For the hybrid model, we take both the RNN embedding dimension and lookup table embedding dimension to be $D = 50$.

We experiment all three models on the Prescription Frequent data set (PRFreq). Results are listed in Table 4.

Model	Mean Rank	Hits@10	Hits@1
RNN	66.8304	45.055%	13.015%
Hybrid	41.3376	50.206%	16.552%
TransE $D = 50$	32.8015	53.571%	15.453%
TransE $D = 25$	31.5885	52.987%	12.843%

Table 4: Three models on Prescription Frequent data set

The result shows TransE with embedding dimension 50 has a better performance than the RNN model on the dense prescription data set, which is as expected, since TransE is using a lookup table

for every entity and it does not generalize, while RNN model is able to process unseen entities. Another interesting point is that the hybrid model has a similar performance as the TransE model on this data set, showing that combining two models helps in this case.

We also train all three models on the Prescription Unique data set (PRUniq), the one without removing infrequent entities. Results are listed in Table 5.

Model	Mean Rank	Hits@10	Hits@1
RNN	223.8726	37.183%	10.577%
Hybrid	207.6983	35.678%	9.346%
TransE $D = 50$	160.0409	36.559%	9.254%
TransE $D = 25$	164.0306	37.477%	8.262%

Table 5: Three models on Prescription Unique data set

The result shows that our RNN embedding model outperforms TransE in hit@1 criteria and has similar Hits@10 as TransE, though TransE has a lower mean rank, mainly because it has a better fit on the frequent data in the training set. Another observation is that reducing the embedding dimension in TransE results in a larger mean rank, but slightly better Hits@10 and worse Hits@1.

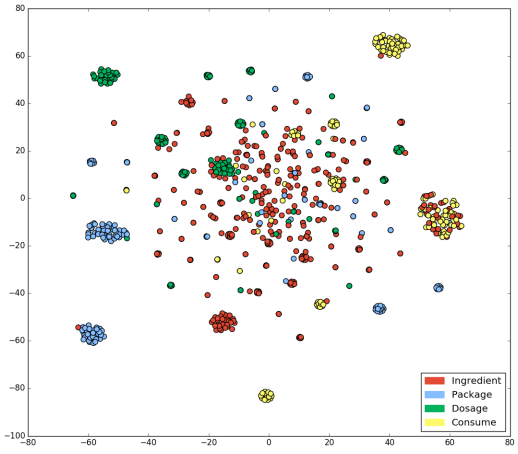

Figure 5: RNN model relations visualization

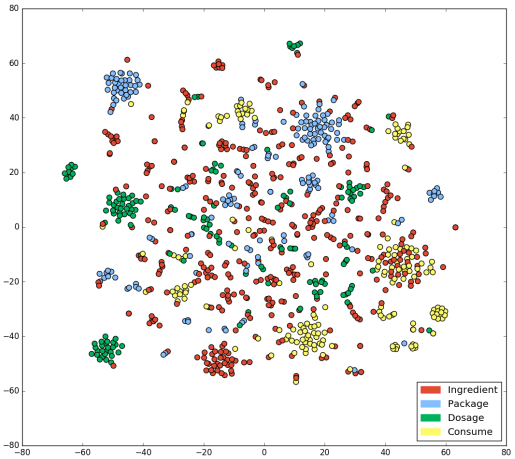

Figure 6: TransE model relations visualization

We visualize the result on test data that appears no more than 6 times in the training set (choosing 6 to control the number of points). Each point in the plots of Figure 5 & 6 is computed by $\boldsymbol{t} - \boldsymbol{h}$, where (h, l, t) is a triple in the test set. We use t-SNE [25] to reduce the dimension of each point to 2; t-SNE is a common dimension reduction technique for high dimensional data and it preserves the similarity and dissimilarity between points. We color each point by the relation each triple has: intuitively we want to see how well each model differentiates different relations in the embedding space. The plots show that the RNN model has a cleaner separation of the four relations than TransE, meaning that points with the same relations are closer to each other and form tighter clusters, while points are spreading across the whole plot under the TransE model.

Using the model trained on Prescription Unique data set, if we only look at the rare entities in the test set, we have the following result, with max freq indicating the maximum frequency that the left hand side entity appears in the training set.

MEDICINE NAME AND RELATION		PREDICTED ATTRIBUTE
Diphenhydramine	*contains ingredient*	water, glycerin, **sodium hydroxide**, sodium chloride, edetate disodium
Rizatriptan Benzoate	*contains ingredient*	**magnesium stearate**, microcrystalline, cellulose, titanium dioxide, lactose monohydrate
Soltamox	*contains ingredient*	water, **propylene glycol**, sodium hydroxide, glycerin, titanium dioxide
Hydromorphone HCL	*has dosage form*	**injection**, solution, capsule, kit, tablet
Avelox ABC Pack	*has dosage form*	capsule, **tablet**, solution, injection, kit
Neomycin	*has package form*	**bottle**, vial, tube, carton, plastic
Dipentum	*has package form*	bottle, **plastic**, vial, blister pack, single-dose
Rapaflo	*consumed in*	**oral**, intravenous, topical, intramuscular, ophthalmic
Ultravate PAC-Cream	*consumed in*	**topical**, intravenous, oral, respiratory (inhalation), subcutaneous

Table 6: Example predictions on the Prescription Unique test set using RNN model

	Max Freq = 2		Max Freq = 1		Max Freq = 0	
Model	Mean Rank	Hits@250	Mean Rank	Hits@250	Mean Rank	Hits@250
RNN	1374.0093	14.019%	1400.9230	20.512%	1491.0322	22.580%
Hybrid	1367.4392	10.280%	1355.1282	5.128%	1477.7419	0.000%
TransE $D = 50$	1461.9813	9.345%	1667.1025	5.128%	1783.1290	3.226%

Table 7: Three models on Prescription Unique data set

And from Table 7 we can see the RNN model has a better result in both the mean rank and hit@250 criteria than TransE, meaning that when an entity is rare in the training set, RNN model is able to produce more meaningful embeddings. While the hybrid model has a slightly better mean rank than the RNN model, its performance under the hit@250 criteria is similar to that of the TransE model, which is much worse than the RNN model in these cases.

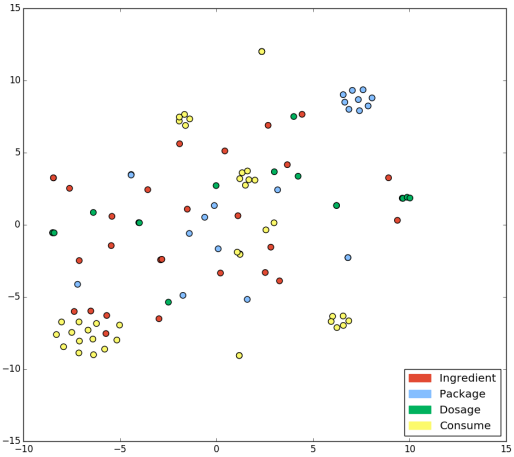

Figure 7: RNN relations on rare triples

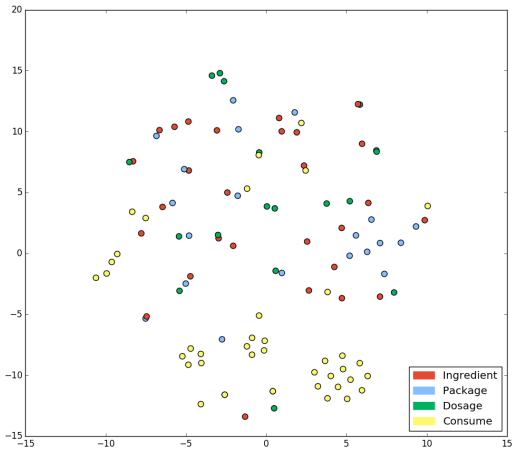

Figure 8: TransE relations on rare triples

We also perform the same visualization for test data that appears no more than twice in the training set. Both visualizations show that the RNN model does a better job in separating the relations while TransE is mixing lots of points in the middle, especially when the data is sparse (infrequent or unseen).

Model	RNN	Hybrid	TransE $D = 50$	TransE $D = 25$
Number of parameters	152,882	310,732	294,200	147,100

Table 8: Number of parameters in each model

Besides the difference in performance on the data sets, the RNN model also differs from TransE in model size (the number of parameters). See Table 8 for a comparison of different model in size. We can see that building a lookup table requires lots of parameters. Take the RNN model as an example; there are 141,200 parameters in the right hand side entity embedding lookup table, while the Bi-GRU recurrent neural network only needs about 10,000 parameters. The RNN model outperforms TransE with similar model size on infrequent data, suggesting that the RNN model is more efficient in modeling such sparse knowledge bases.

6 Conclusion

We explored various approaches of performing knowledge base completion on a prescription data set, and proposed a Bi-GRU RNN embedding model to deal with the sparsity issue that appears in the data set and many real-world situations. Our work shows that the Bi-GRU RNN embedding model outperforms the baseline model (TransE) when entities appear only a few times in the training set, and it can deal with any input entities, regardless of whether the entity is in the training set or not, while existing models can only predict on previously seen entities.

Future work could further analyze the Bi-GRU RNN embedding model and its variants, and apply this model to more domains where one can also exploit the regularity in the entity names such as chemical substances or entities with name in other languages. It's also interesting to see the performance of this model under different loss functions or different parameter settings including entity embedding dimension and character embedding dimension.

Acknowledgement

Thanks to Bhuwan Dhingra for useful discussions about the RNN model and providing the Tweet2Vec code for implementing the RNN model.

References

[1] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM, 2008.

[2] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[3] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM, 2007.

[4] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Learning multi-relational semantics using neural-embedding models. *arXiv preprint arXiv:1411.4072*, 2014.

[5] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine learning*, 62(1-2):107–136, 2006.

[6] Ni Lao, Tom Mitchell, and William W Cohen. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 529–539. Association for Computational Linguistics, 2011.

[7] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934, 2013.

[8] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795, 2013.

[9] Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer, 2006.

[10] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[11] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[12] Ni Lao and William W Cohen. Relational retrieval using a combination of path-constrained random walks. *Machine learning*, 81(1):53–67, 2010.

[13] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, volume 5, page 3, 2010.

[14] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. Learning structured embeddings of knowledge bases. In *Conference on Artificial Intelligence*, number EPFL-CONF-192344, 2011.

[15] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[16] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[18] Cícero Nogueira dos Santos and Bianca Zadrozny. Learning character-level representations for part-of-speech tagging. In *ICML*, pages 1818–1826, 2014.

[19] Bhuwan Dhingra, Zhong Zhou, Dylan Fitzpatrick, Michael Muehl, and William W Cohen. Tweet2vec: Character-based distributed representations for social media. *arXiv preprint arXiv:1605.03481*, 2016.

[20] Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. Finding function in form: Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096*, 2015.

[21] Kristina Toutanova, Danqi Chen, Patrick Pantel, Pallavi Choudhury, and Michael Gamon. Representing text for joint embedding of text and knowledge bases. *ACL Association for Computational Linguistics*, 2015.

[22] Lidong Bing, Sneha Chaudhari, Richard C Wang, and William W Cohen. Improving distant supervision for information extraction using label propagation through lists. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 524–529, 2015.

[23] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

[24] Sander Dieleman et al. Lasagne: First release., August 2015.

[25] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.