Generating Vector Space Embedding for Lexical Graphs

Ali Saleh

1st of December 2017

Outline

Outline

1. Problem Definition
2. Related Work
   - Bilinear Link Model(BLM)
   - DeepWalk
   - Random Walkers
   - Direct Edge Weights
   - Trans*
3. Baseline Experiment
   - Experiment Design
   - Datasets Description
   - Results
4. Future Work
   - New Model

Problem Definition

- Exploiting word relations in knowledge graphs to generate proper embedding in vector space.
- Words that are close (Synonyms, Hyponyms, in the same local sub-graph, etc...) to each other should have embedding that are closer [1] to each other.
- Similar to word2vec, but working on knowledge graphs

[1] Closeness measured by cosine similarities between vectores

Outline

Bilinear Link Model (BLM)

- Maintain two representations (input, output)
- Assign link probability using softmax.
  $p(v|u,\theta) = \frac{\exp(In_u^T Out_v)}{\sum_{w \in V} \exp(In_u^T Out_w)}$
- Optimize model using Noise Contrastive Estimation

DeepWalk

For each node in the graph:

- Create personalized ranking (based on weights) for directly connected nodes
- Do weighted random walks based on connection weights between edges.
- Variable depth for how many levels to go for each node
- Output random walks connections as context to the source node

Finally a word2vecf model is trained using the words and it's context.

Random Walkers

Similar to DeepWalk, but without the destination nodes ranking. So for each node in the graph

- Do random walks on destination edges.
- Depth for how many levels to go for each node is set to 10
- Output random walks connections as context to the edge

Finally a word2vecf model is trained using the words and it's context.

Trans*

Trans* are a set of methods that are used to represent Knowledge Graphs in vector space. It's representing each connection (head, tail) as two vectors, and represent the relation as an operation from one the two vectors to the other

- TransE
- PTransE
- TransH
- TransR

Outline

Baseline Experiment

To evaluate the future work to be done in this thesis a number of related work models were recreated as baselines.
Source code from the original work was used when available.

Experiment Design

The experiment is consists of recreating the following models

- BLM
- DeepWalk
- Random Walks
- Direct Connections (naive baseline)

The training in conducted on two datasets separately

- WordNet
- Serelex

Experiment Design Cont.

The performance of each model is evaluated on the following gold standards

- WS-353
- SimLex-999

Each of them is a list of word pairs accompanied by a golden standard of words similarity (e.g. tiger,cat,7.35)
The similarity rating from the trained models is calculated using cosine similarity, then are compared to the golden standard ratings by means of Spearman's correlation coefficient.

Datasets Description

Both WordNet and Serelex are lexical graphs.
They contain words (Nouns, verbs, etc...)
Relations between words (hypernym, hyponyms, meronym, etc...)

Metric	Wordnet	Serelex
unique words	147306	419753
edges	2790502	11251236

Table: Datasets Statistics

Direct Edge Weights

Basic baseline that uses weight of words connections as a similarity metric between each two words, Zeros is used if no direct connection is available.

Results

	Serelex		WordNet		
Method	Task	Correlation	P-Value	Correlation	P-Value
BLM	WS-353	0.0376	0.4993	0.4143	1.88e-14
	SimLex-999	0.0026	0.9426	0.5149	6.35e-50
Random Walks	WS-353	-0.0888	0.1097	0.1509	0.00677
	SimLex-999	0.0214	0.5657	0.1125	0.0025
DeepWalk	WS-353	0.01797	0.747	-0.07180	0.1995
	SimLex-999	-0.083	0.0264	0.05297	0.1560
Direct Connection	SimLex-999	0.326	3.23e-19	0.3160	3.93e-18
	WS-353	0.587	1.38e-31	0.5997	0.2841

Table: Similarity tasks correlation score

Outline

New Model

A new model to be developed in this thesis which shares a lot of similarities with the BLM model. The objective function of the model is:

$$J(\mathbf{X}) = \sum_{i \in V} \sum_{j \in V} (w_{ij} - \mathbf{x}_i \cdot \mathbf{x}_j)$$

The following strategies

- Uses the wights of the edges in model optimization
- Explore The possibility of adding second order nodes (transitive paths from i to j through k) with compound weights
- Explore ways to down weight second-order nodes
- Try using NCE to achieve faster optimization

Thanks