Relational Inference with Freebase

Billy Jun, William Song, Jim Cai

December 2014

Abstract

We are interested in developing a method to accurately infer the type of an edge based on the neighboring entities and the local graph structure that it is embedded in. Working with a modified Freebase network, we project a node into a latent space learned from the neighboring relations with different entities and use its affinity to other nodes in such a space for making predictions. With this model, we were able to achieve an accuracy of 94.2%, which is an improvement over the Naive Bayes baseline (84.5%). With this method, we also hope to augment the current knowledge base and discover interesting relationships in the existing Freebase data.

1 Introduction

The proliferation of the internet has allowed new technologies to flourish. The collaborative knowledge base is a development that would not have been possible without the internet facilitating crowd sourced information accumulation. Wikipedia and Freebase are the two well-known examples of successful and useful knowledge bases. At the time of writing, Wikipedia contains more than 4 million articles and Freebase has over 2.5 billion facts. In this project, we aim to conduct relational inference on the Freebase network involving people. More specifically if there is a known relationship between two entities, we try to categorize that relationship. This answers more generally how well we are able to characterize two nodes' relationship in the context of their position in the knowledge graph.

To perform inference, we explore two methods for estimating the conditional probability that a given edge is a specific edgetype: Naive Bayes and Entity Vector Embeddings. We examine the local contexts of the two connected nodes and collect sufficient statistics for the Naive Bayes Model; at prediction time, we take the product of the relevant contexts to arrive at a probabiliy. For estimating a vector embedding for each entity, we follow a similar framework to Socher, et al. [5] by deriving an embedding for each entity. An embedding is a k dimensional vector that can be used to compare two nodes' qualities. Once we have an embedding we can use it to predict the likelihood of a particular edgetype between two nodes.

2 Related Work

2.1 Reasoning with Neural Tensor Networks for Knowledge Base Completion [5]

One downside of this approach is that since it only uses the word-based information, it fails to resolve categories that require contextual disambiguation; while the tensor-network system is nearly perfect for simple relations like gender and nationality on FreeBase, it had particular difficulties with causes of death and institutions.

The main contribution of this paper is the formulation of a new neural tensor network, suitable for the task of relation extraction between two entities. Each entity is represented by an embedding, which allows each dimension to encode a certain syntatic and/or semantic notion in a fuzzy manner [9]. The authors of this work extended this for the relation extraction task by learning a composition function that computes the score of a relation given two embeddings. The biggest strength of this approach is that this enables us to learn the latent embeddings and a scoring function that incorporate worldly knowledge without manual feature engineering. Our relation extraction problem can be posed as predicting the category of an edge in a knowledge graph where nodes are the entities.

2.2 Latent Space Approaches to Social Network Analysis [3]

The authors suggest a method to model relationships between nodes in a network by their pairwise distance as well as local structure; they emphasize that the distance needs only obey the triangle inequality. The transitivity of this metric allows their method to substantially improve the baseline stochastic block-model approach.

It is very promising that the authors are able to achieve reasonable results with just using a simple distance metric; they are also able to present a model-based spatial representation of the network relationships. One limitation that the authors do not address is the generalizability of such a latent space–they learn parameters on a given graph and then how well those parameters describe the specific data.

2.3 Link Prediction in Relational Data [6]

In this paper Taskar, et al. proposed a novel approach for predicting link existence and types in a relational graph using the method of relational Markov network. Relational Markov networks differ from the traditional Markov networks in that the attributes and potentials are specified at the template level, which is suitable for the task since multiple entities could satisfy the same template, and thus, should have shared parameters during training and inference.

The authors show that if multiple entities $A_1, .., A_n$ are shown in the same context as another entity E, then the relation types of $A_1 - E, ..., A_n - E$ should be the same or similar. Further, a link structure of $A - B$ and $B - C$ should strongly indicate the likelihood of the existence of a link $A - C$.

3 Approach

3.1 Problem Definition

Given a knowledge graph $G = (V, E)$, the objective is to predict the relation between two entities using the surrounding context. We shall use the following notation throughout this work to pose our problem and methods:

$G = (V, E)$	Knowledge graph (directed)
V	Set of entities
R	Set of all relation types
$E \subset V \times R \times V$	Two entities and the relation between them define an edge
$r : V \times V \to R^+$	Returns the set of all relations between two entities
$x : V \to \mathbb{R}^n$	Embedding vector of an entity

3.2 Naive Bayes Model

Our first approach is to model the conditional likelihood of the relation type of an edge. We use the context of the two nodes' merged ego networks and apply the Naive Bayes assumption. The intuition behind this method is that if we frequently observe that certain patterns of paths between two nodes give a high conditional probability for another length-1 path, we could predict with high confidence with the existence of that path. For instance, we would most likely observe from the data that given the length-2 path of $u \xrightarrow{\text{parent}} w \xrightarrow{\text{child}} v$, the conditional probability of seeing $u \xrightarrow{\text{married}} v$ is going to be high.

More rigorously, let $c_r \in \mathbb{N}^{(|R|+1)^3}$ denote a vector of length $(|R| + 1)^3$ used for accumulating the sufficient statistics for relation type r. During training, for every relational typed edge $r \in R(u, v), \forall u, v \in V$ we gather the frequency of all possible paths up to length 3 between u and v; we increment c_r accordingly by mapping the path as defined by the relation type of the edges $S = s_1 s_2 s_3$ to its corresponding index, $s_i \in R \cup null$. We use a length of $(|R|+1)^3$ instead of $|R|^3$ to account for the fact that some of the alternate paths will have length less than 3, so we designate the *null* relation type for such a distinction. At the end of training, for each relation r, its corresponding c_r accumulates the distribution of paths between nodes connected by r of length up to 3, as represented by $p(S = s_1 s_2 s_3 \mid r)$. Given the prior distribution of $p(r)$, we can use Bayes rule and the Naive Bayes assumption to compute the likelihood of a relation between u, v as

$$p(r|S : S \in ego(u, v), 1 < |S| \leq 3) \propto \prod_{S:S \in ego(u,v), 1<|S|\leq 3} p(r)p(S|r)$$

where $ego(u, v)$ refers to the union of the ego networks of u and v. To account for the fact that many of the relational paths will be non-existent in the training data, we apply Laplace smoothing with $\lambda = 1$. To compute all paths up to a certain length between u and v, we simply applied depth limited DFS from u. Note that in our equation, S is denoted as a set. This was intended since we found later that there are a significant number of large cliques in

the graph. If we use multiset for update, it would not only be computationally expensive, but it also imposes a strong prior for the relational paths in the complete graphs.

One drawback of the approach is that the model does not scale well when we try to model longer paths as the size of our feature vector grows exponentially with respect to the length of the path. Further, the conditional indepence assumption we have used is not well justified in the context of relational inference, as the different paths between two nodes are more than likely to be conditionally dependent (e.g. if two actors played in some movie with another actor, then it would increase the seeing another such relation in the two actor's ego network). Therefore, we intend to use the Naive Bayes model as a baseline for comparison with the Entity Vector Embedding model.

3.3 Entity Vector Embedding Model

We will leverage the neural network language model technique [9] to learn continuous representations of entities to perform efficient graph inference. To motivate our method, consider a path between two entities in our knowledge graph: $v_1 \xrightarrow{r_1} v_2 \xrightarrow{r_2} \cdots \xrightarrow{r_n} v_{n+1}$. The k-dimensional embedding of this knowledge is defined as the average of individual embeddings computed by the composition function f, which will be revisited later:

$$f(v_1, v_{n+1}) = \frac{1}{n} \sum_{i=1}^{n} f(v_i, r_i, v_{i+1}) \in \mathbb{R}^k.$$

Once the continuous representation is computed, this can be used as an input for various classification and ranking tasks. In this project, we are interested in predicting the likelihood of a certain link. The model is given by

$$\Pr[(v_i, v_{i+1}) \text{ is related}] = \sigma(W_2\, g(W_1\, f(v_i, v_{i+1}) + b_1) + b_2),$$
$$W_1 \in \mathbb{R}^{m \times k}, W_2 \in \mathbb{R}^{R \times m}, b_1 \in \mathbb{R}^m, b_2 \in \mathbb{R}^R$$

where $\sigma(\cdot)$ is the sigmoid function and $g(\cdot)$ is the rectilinear nonlinear function applied to the hidden layer. During training time, the parameters and the embeddings will be learned simultaneously as done in training neural probablistic language models [2] via backpropagation from the cross-entropy loss.

Next, we present our method to learn representations of larger structural properties of the knowledge graph. Instead of operating over a single chain, each query pair (v, v') will be assigned an embedding as follows:

1. Compute the flow graph $G' = (V', E')$ from the source entity v to the sink entity v' using a similar approach as mentioned in the previous section:
   (a) Run BFS from v on G and from v' on G^{rev}. Intersect the two graphs.
   (b) Run MST from v on this graph with uniform edge weights to remove cycles.
   (c) Repeatedly remove sinks that are not v'.
2. Set the weight of each edge $w = 1/|E'|$.

3. Compute the weighted distributed representation:

$$f(v, v') = \sum_{(v,r,v') \in E'} w_{(v,r,v')} f(v, r, v')$$

In our original formulation, we distributed 1 so that the weight of an edge is inversely proportional to the number of paths along that edge. The intuition of this procedure was to discount the evidence of an edge present in many paths. However, we found that this preprocessing added a lot of overhead in training. Also, inspired from the recent state-of-the-art result in paragraph embedding learning achieved with a simple average composition [4], we set the weights of each edge to be uniform.

So far, the composition function has not been clearly defined. There are a number of desirabie properties in this function such as differentiability, but one of the most important is the asymmetry in the arguments. This is necessary because a relation is intrinsically directional: for example, [George H. W. Bush] $\stackrel{\text{father}}{\longrightarrow}$ [George W. Bush] but not [George W. Bush] $\stackrel{\text{father}}{\longrightarrow}$ [George H. W. Bush]. The tensor network model satisifies all desired properties:

$$f(v, r, v') = x(v)^\top W_r^{[1:k]} x(v') + V_r \begin{bmatrix} x(v) \\ x(v') \end{bmatrix} + U_r \in \mathbb{R}^k,$$

where $W_r^{[1:k]} \in \mathbb{R}^{n \times n \times k}$, $V_r \in \mathbb{R}^{k \times 2n}$ and $U_r \in \mathbb{R}^k$ are the composition parameters for the relation r. In practice, $x(v)$'s are stored in a matrix $X \in \mathbb{R}^{n \times |V|}$ (e.g. $x(v) = X \cdot \mathbf{1}_v$) and errors are backpropagated to this X matrix for each training instance.

For evaluation, we will initially sample a fraction of edges in the original graph and hide them for development and test. Note that these edges are not present during training. Their properties can be inferred efficiently without having to retrain or finetune, since the individual entity embeddings and relation composition parameters are known.

4 Data and Evaluation

4.1 Freebase Graph

FreeBase is a comprehensive knowledge base detailing most of the popular named entities in the world and their intricate relationships. It is edited by the community (much like Wikipedia) and details categorical relations between entities. We leverage Freebase Easy [1], an entity resolved data dump of Freebase.

The Freebase is represented as a network, with people representing nodes and their shared relationship as edges. For example, Michelle Obama is connected to Barack Obama by a *Spouse* edge. Also, people can be related to non-people entities. For instance, Barack Obama can be the *President of* the United States.

4.2 Relation Augmentation

Apart from direct familial relationships such as spouse, children, sibling, coworker, etc. we want to introduce more connections in the network in order to capture more interesting concepts that can be infered from the existing data. As such, if two people share an attribute, we connect them directly with a typed edge. For instance, if two actors played in the same TV episode, we connect them with an edge indicating that they've played in *some* TV episode together. We do not use the actual TV episode as part of the edge type in order to reduce the cardinality of the set of edge types that we need want to predict. In the end, we end up with 34 edge types. We manually filter through the edge attribute types in order to filter out ubiquitous attributes like gender and religion and exclude attributes that are too sparse. Since we assumed the network to be directed, we also add a reverse edge for non-reciprocal relationships to effectively make each typed edge bidirectional for better estimation because most of the edgetypes are bidirectional. For instance, we added an *parent* edge to reciprocate the *child* relationship.

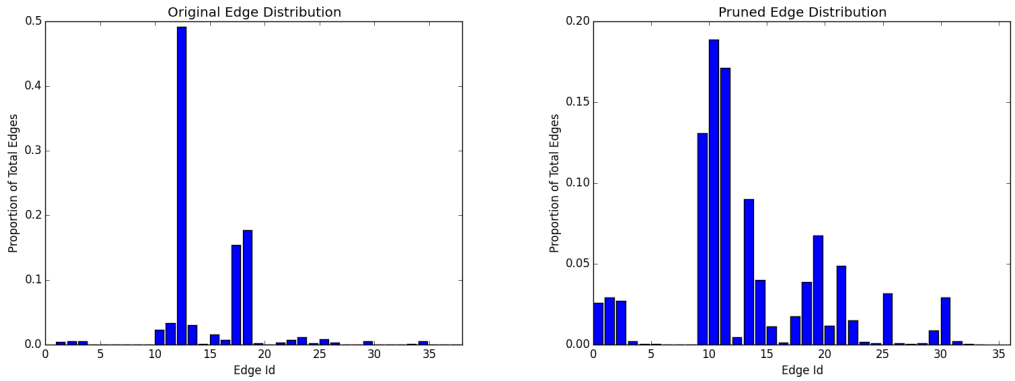


Figure 1: Comparison of edge type distribution after removing some of the most common edge types

4.3 Graph Analysis

During pre-processing we noticed that the appearance of the edge types (Film performance, Employment history, Record Label) exhibited power-law-like qualities. In order to reduce the bias of our model, we removed the three most frequently appearing edgetypes, which contribute to 81% of the edges. We are left with approximately 4 million edges on around 700,000 entities. The new edge type distribution is much more balanced and well suited for inference.
The Freebase people graph is quite disconnected, with a diameter of 67 and an average clustering coefficient of 0.4752. The node out degree distribution and size of weakly connected components both exhibit power-law properties. Some examples of edge types in our final graph:

- *Spouse*: Barack, Michelle Obama
- *Child*: Barack, Malia Obama
- *Film appeared In*: Ford, Fisher
- *Record label*: 50 cent, Eminem

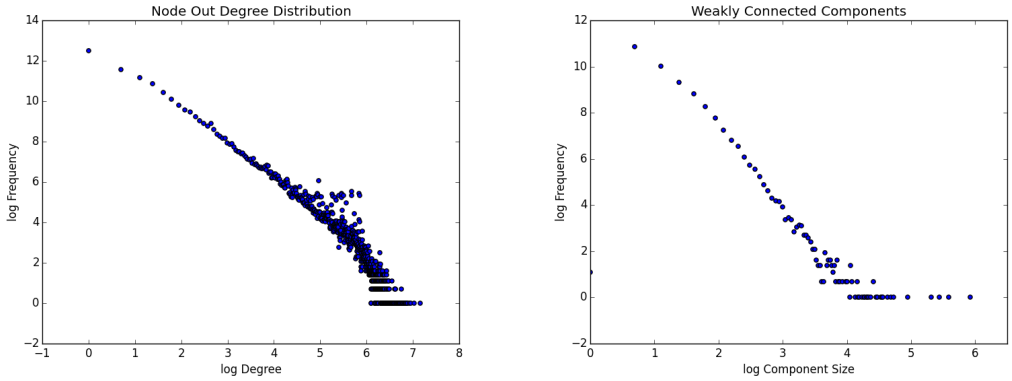


Figure 2: Degree distribution and component size distribution of our curated network.

- *Competed together*: Phelps, Thorpe
- *Romantic relation*: Pitt, Aniston

Since we curated the graph, we are interested in its ability to still accurately represent relationships between people. We utilized community detection with BIGCLAM [10] as a way to sanity-check whether there are reasonable communities within the data, and leverage the library within SNAP to discover 10,000 communities. We examine the communities discovered and for many of them we are able to identify the shared link between members. Here we present two of the commnities we found in Table 1.

Gaelic Athletic Association	**Austrian Royalty**
Robert O' Keefe	Princess Tatiana Galitzine
Liam Clifford	Princess Alexandra Galitzine
Dan O' Rourke	Prince Dimitri Galitzine
Seamus MacFerran	Archduke Martin Carl Amedeo Mario
Peter Kelly	Prince Franz-Josef of Bavaria
Patrick Breen	Prince ludwig Ferdinand of Bavaria
John Dowling	Hermann van Pels
James Nowlan	Infanta Alicia, Dowager Duchess of Calabria

Table 1: Two communities detected using the BigCLAM algorithm.

4.4 Evaluation

Our Freebase graph by construction is a multigraph. However, in practice, the vast majority of query pairs have only one edge type between them. Rarely did we find two or three edge types between two entities. For this reason, despite the fact that the edge type prediction is a multi-label binary classification task, we evaluated by checking if the highest scoring edge type is present in the ground truth label set. Note that negative samples were never used for training and evaluation, because it is inherently unclear to decide whether two entities are truly unrelated or their relations are unannotated in our moderately sparse graph. Quantitatively,

the accuracy metric is defined as follows:

$$A(p, y) = \frac{\sum_{i=1}^{n} \mathbf{1}\{\arg\max_j p_j^{(i)} \in y^{(i)}\}}{n}.$$

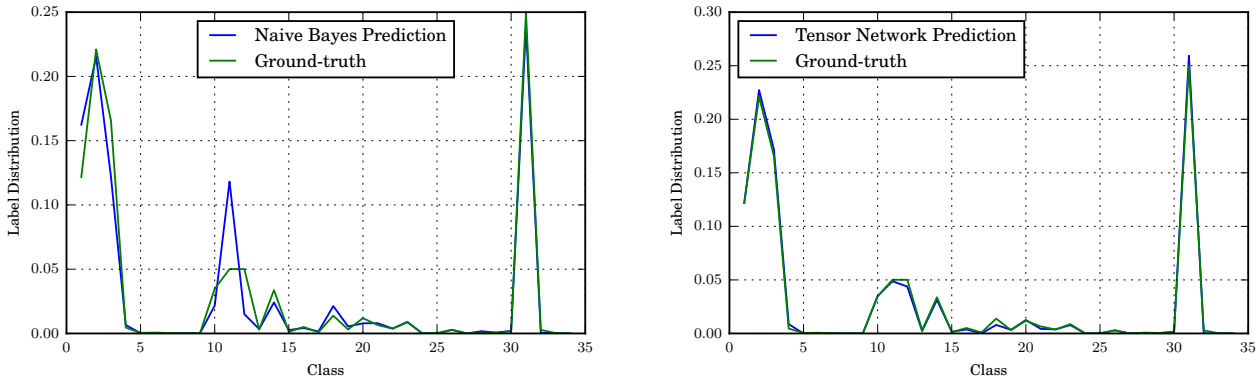
For the vector embedding model, we also evaluated the quality of our learned embeddings by visualizing them with the t-SNE dimensionality reduction algorithm [7]. The actual implementation we used is an approximated version called Barnes-Hut-SNE [8] which is much more resource efficient that can scale to large data.

4.5 Results

The results of the baseline and our entity embedding model are documented in Table 2. We didn't try embedding sizes of larger than 50, because the number of parameters in our tensor network has a cubic complexity. It turns out using an embedding size of 15-20 was sufficient.

Model	**Train Accuracy**	**Test Accuracy**
Majority Classifier	–	18.9%
Naive Bayes	89.1%	84.5%
Entity Embedding Vector	95.5%	94.2%

Table 2: Edge Type Prediction Accuracy Rate

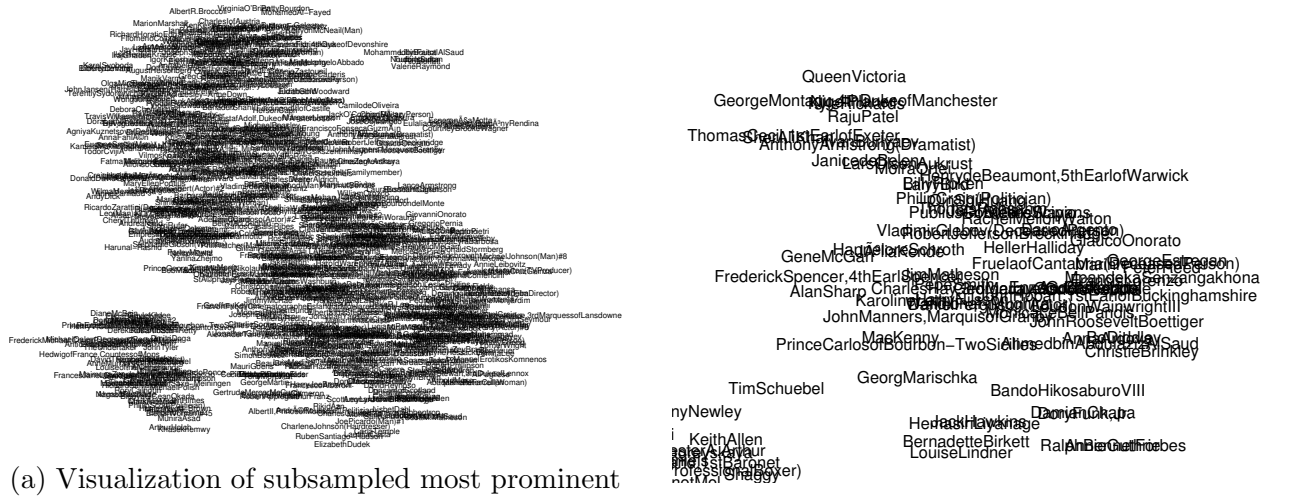


(a) Naive Bayes Prediction Distribution

(b) Tensor Network Prediction Distribution

Figure 3: Label distribution showing where most mistakes were made

Next, we used the t-SNE algorithm to reduce the dimensionality of our entity vectors to two. Even with the resource efficient approximated implemenetation, visualizating 200,000 entities were quite time-consuming. As such, we selected 50,000 most salient entity vectors Figure 4a by the criterion of $\mathbb{E}[V] + \sqrt{\text{Var}[V]}$, which is inspired by the UCB-1 algorithm. From this, we found a number of clusters grouped by certain regularities. There were major groups by occupations, ethnicity, etc. The t-SNE also helped discover the royal family in UK (Figure 4b).



(a) Visualization of subsampled most prominent entities

(b) Discovery of royal family

Figure 4: Visualization of entity embeddings using t-SNE on a 2D plane

4.6 Error Analysis

If we examine a few of the model predicted errors from our vector embedding model, we notice some interesting patterns. Val Bisoglio and Roger Perry were both predicted to have "Starring roles in a TV show", but actually they only share a "TV Episode Performance". This error is actually quite subtle because both people are prominent television actors and have starred as well as appeared in many shows. Because we are predicting the relationship between people rather than the relationship between a person and another entity (TV show entity in this case), the model loses precision and cannot use the fact that an entity already has a sufficient number of starring roles. More generally, for relationships that involve a third entity, it is conceivable that the model chooses the one that is more representative of the people involved,whereas in reality it is probably more accurate to consider the entity involved (especially for very similar relations).

Another common mistake is again with similar relationships. After analysis, it is understandable that the model confuses "Influences" and "Academic Advisor to" which we see fairly often (as with Sylvia Plath and Elizabeth Wurtzell). Because the only notion of time in the graph is the people that one associates with, it is conceivable that "Influences" is difficult to accurately estimate because it is one of the only relations that transcends time (Plato can "influence" Chomsky). While "Academic Advisor" is a subset of "Influences", only we understand the difference while the model is noticeably fuzzy about the characteristic differences between the two.

Comparing the results between Naive Bayes and the Entity Vector Embedding model, we found that although the latter made fewer mistakes overall, it made slightly more mistakes in familial relations. For instance, occassionally siblings are confused with parents. We believe the conditional probability method used by Naive Bayes provides a much stricter barrier that makes it difficult to make mistakes on relations that it rarely sees in training, whereas due to the slightly more distributional nature of the Entity Vector Embedding model (and possibly insufficient training time) it is unable to fully avoid making those mistakes. Furthermore, our

neural network weakly incorporates the order information along paths, unlike Naive Bayes, and our weighting scheme may not fully take advantage of the directionality of the flow graph.

5 Conclusion and Future Work

In this project, we curated a large relational graph of prominent people using a subset of Freebase data. We built two independent models that accurately predict the relationship between two nodes based on the relations among the surrounding entities. In general, our Entity Vector Embedding model was quite successful, but we can still improve its capability to incorporate relational information.

In our entity embedding formulation of the conditional probability estimation, we were limited by computation time in exploring the context between two nodes. We constrained our exploration by only traversing a path of maximum length n in the graph traversal; although the running time increases exponentially as the path length increases, the quality of entity vector embeddings also increases drastically because the entity vectors are updated more frequently as they will appear in more paths. Ideally, we'd also be interested in using our tool to perform anomaly detection and knowledge augmentation for Freebase or other knowledge graphs.

References

[1] Bast, et al., "Easy Access to the Freebase Dataset." WWW Companion '14

[2] Bengio, Yoshua et al. "A Neural Probabilistic Language Model." Journal of Machine Learning Research 3 (2003) 1137-1155.

[3] Hoff, et al., "Latent Space Approaches to Social Network Analysis." *Journal of the American Statistical Association*, 2002.

[4] Le, Quoc and Tomas Mikolov. "Distributed Representations of Sentences and Documents." Proceedings of the 31 st International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP Volume 32

[5] Socher, et al., "Reasoning with Neural Tensor Networks for Knowledge Base Completion." *NIPS*, 2013.

[6] Taskar, et al., "Link Prediction in Relational Data." *NIPS*, 2003.

[7] Maaten, Laurens van der et al. "Visualizing Data using t-SNE." JMLR, 2008.

[8] Maaten, Laurens van der. "Barnes-Hut-SNE." CoRR, 2013.

[9] Mikolov, et al., "Efficient Estimation of Word Representations in Vector Space." *CoRR*, 2013.

[10] Yang, Jaewon and Jure Leskovec. "Overlapping Community Detection at Scale: A Nonnegative Matrix Factorization Approach." WSDM'13, February 4–8, 2013, Rome, Italy.