Towards Neural Prerequisite Predictors

11-747 Assignment 1:
Survey of Related Work and a Baseline Model

Fatima Al-Raisi

Contents

1 Introduction

This project aims to learn dependency relations between documents and dependencies among concepts across documents. For examples, if a news article j refers to an event or an entity described in more detail in news article i, then j *depends* on (the information in) i. Similarly, if course j builds on concepts explained in course i, then course j *depends* on course i or, equivalently, i is a prerequisite for j. In mathematics, the concept of "permutations" depends on "factorials" and "factorials" depend on "multiplication." Knowing dependency relations among concepts and between documents is important for content structuring, coherent content generation, and presentation/ordering of results in various information retrieval tasks.
We frame this problem and contextualize it in the educational domain due to the resemblance to curriculum planning and generation problem in education (1) and the availability of training data in the form of document dependencies defined as prerequisite relations between courses. The larger scope of this project in the educational domain includes automatic curriculum planning and generation. To produce a coherent curriculum, we need to know dependency relations between concepts within a course and proper ordering of courses in a sequence of courses.
In this milestone, we specifically focus on the problem of predicting the prerequisite relation between two courses and define it as a classification task where a sequence of two courses is judged as either coherent or incoherent depending on whether information is presented in the correct order. The objectives of this project include:

- learning fine-grained dependency relations between concepts
- learning a distributed representation of concepts and documents suitable for the task of course prerequisite prediction in STEM domains
- using both textual and concept-graph embeddings of concepts to improve learning of dependencies and equivalences between courses

Work in this area is essentially similar to well-studied problems in natural language processing, as summarized in Table 1. Progress towards solving these broader objectives will be reported in the second milestone and final project deliverables. In this milestone, I present related work, formulate the task as a simple document coherence binary classification task, and present first neural results using a convolutional neural network model.

NLP Problem	**Educational Parallel**	
Language Modeling	Curriculum Planning	given previous concepts, generate next concept(s)
SkipGram	Prerequisite Prediction	given a concept, predict previous concept(s)
Alignment Learning	Concept Dependency Learning	learn which specific concepts are strongly related

Table 1: NLP tasks and similar problems in the educational content design domain

2 Related Work

Relative to the size and rapid growth of research in educational data mining [36, 4, 27, 32], little research has been dedicated to modeling educational content. However, the availability of course textual description through MOOCs and websites of universities has facilitated the

emergence of research in this area. Automatic course prerequisite prediction based on course textual content is a relatively new problem. This problem is also of great relevance in the MOOCs and open-access to-information era. A learner navigating this vast and rapidly growing space needs to know cross-institutional dependencies and overlap between content from different sources in order to navigate the space efficiently and effectively.

Perhaps the most relevant work to this problem is the work on concept graph learning from educational data [43] where the task is to automatically learn cross-institutional prerequisite relations using training data consisting of observed prerequisites between courses from the same institution and a concept representation of each course. The goal is to map the courses, using their concept representation, into a canonical space of concepts and do the reasoning in that space. The concept graph is used in this case as an intermediate step or interlingua in a transfer learning setting where the inference is mapped from the course space to the concept space and then projected back to the course space. Figure 1 in Appendix A illustrates this process. Notice from the figure that the dependency between linear algebra and scalable algorithms is not directly observed in the course space, but can be inferred from the mapping in the concept space.

In [43], course prerequisite learning is defined as an edge prediction problem and formulated as a binary classification problem where the edge between two new courses x_i and x_j is classified as either present or not. The formulation for the classification approach is:

$$\min_A \sum_{i,j} \left(1 - y_{ij}(x_i^T A x_j)\right)_+^2 + \lambda \|A\|_F^2 \tag{1}$$

where n is the number of courses in a training set,
p is the dimension of the concept graph,
$x_i \in \mathbb{R}^p$ for $i = 1, 2, \ldots, n$ are the bag-of-concepts representation of a course in the training set,
X is an n-by-p matrix where each row is x_i^T
Y is an n-by-n matrix where each cell is the binary indicator of the prerequisite relation between two courses,
i.e., $y_{ij} = 1$ means that course j is a prerequisite of course i, and $y_{ij} = -1$ otherwise,
A is a p-by-p matrix, whose elements are the weights of links among concepts; i.e., A is the matrix of model parameters to be optimized given the training data in X and Y.

The quantity $x_i^T A x_j$ represents the sum of weights of all paths from concepts in course i to concepts in course j through the concept graph A. If there are many concepts in course i that serve as prerequisites to concepts in course j, then the quantity $x_i^T A x_j$ is large and the prerequisite link y_{ij} is equal to 1. In this case, the objective is to minimize the difference between 1 (the ideal prediction for a prerequisite relation) and the multiplicative quantity $y_{ij}(x_i^T A x_j)$.

One limitation of the above approach is that it also captures overlap and association; i.e., symmetric relations and not only prerequisite which is a directed asymmetric relation. To address this limitation, the problem can be approached as a ranking problem where the objective is to score course and prerequisite pairs (positive examples) higher than unrelated pairs; i.e., the objective would be in this case to maximize the difference of system generated scores given to positive and negative pairs respectively as given by the following equation:

$$\min_A \sum_{(i,j,k)\in T} \left(1 - (x_i^T A x_j - x_i^T A x_k)\right)_+^2 + \lambda \|A\|_F^2 \tag{2}$$

where (i, j, k) is a course triplet where course i is a prerequisite for course j and course i and course k are not in a prerequisite relation. In other words, (i, j) is a positive example where (i, k) is a negative example.
In [43], different concept representations were attempted including a distributed representation obtained from neural models [2] but was reported to be inferior to other concept space representations. Since performance on different tasks can be sensitive to the choice of pretrained word embeddings, whether they are trainable, and the initialzation of unkown word vectors as reported in [10], we intend to run more experiments on different distributed concept representations and include results in the next milestone report.

In another related study [1], the goal is to generate study plans from a "noisy" concept graph constructed from Wikipedia. A subset of relevant Wikipedia concepts are first grouped using a similarity metric, then features such as in-degree, out-degree, number of languages, number of categories, etc. for the concept page are used to distinguish foundational concepts (higher in-degree, larger number of categories, etc) from advanced concepts. These features are used in an optimization problem that aims to find an ordering over learning units which fulfills desirable educational properties such as locality of reference and prerequisite compliance [1].

Related also to content coherence is the work on generating coherent news chains [39], where the task is to find a coherent chain (of articles) linking two input news article. In [39], the desirable properties of a coherent sequence are formalized. Sequences are scored according to whether they contain important keywords as a running theme, whether a path between two articles can be found through a bipartite graph of articles-keywords to solve the problem of missing words, and whether the sequence suffers from "jitterness" (where topics appear and disappear throughout the chain). These properties are formalized as well-defined objectives to connect articles in a proposed sequence and the problem of generating the optimal sequence is formalized as a search problem and solved using an integer linear program formulation.

Another related line of work is on adaptive instructional policies that attempt to optimize student performance and uses student performance trajectories as learning data [3, 11, 35]. The models are based on Bayseian Optimization such as multi-armed bandits [3], Partially Observable Markov Decision Process [35], and Bayesian Knowledge Tracing [9, 44]. These approaches differ in that they use an explicit (state-space) model of knowledge and in using learning data (student scores in assessment) to guide optimization. Some also assume (soft) prerequisite constraints as input which is what we need to learn in this project. We do not optimize on student performance metrics for a number of reasons including maintaining generalizability of the method to learning dependencies between documents and features in any domain.

3 Related Deep Learning Literature

We note that the above related work does not use or apply deep neural models to learn dependency relations between concepts or documents. However, there is a rich literature in deep learning that intersects with our problem in the following aspects:

- Creating distributed representations of concepts using textual context, knowledge graphs/ontologies, or both

- Creating document vector representations
- Mapping (pairs of) documents to different semantic classes

We summarize related work in each of the above directions in the following subsections.

3.1 Neural Models for Concept Representation

In [12], more than 5 million Wikipedia pages were used to train a Skip-gram model [26] with negative sampling to produce *concept* embeddings. The model was designed to learn both word and Wikipedia concept (page anchor) vectors, resulting in words and concepts being mapped to the same space which facilitates the comparison of the similarity among concepts and also between concepts and words. Their concept embeddings are not ambiguous; i.e., there is a different vector for concepts with similar surface forms but different mentions/page contents. Their embeddings perform competitively on phrase analogy and phrase similarity tasks [12].
Classical approaches to training word embeddings require large amounts of data to produce good representations. An approach for learning word embeddings from domain specific sparse data while incorporating domain knowledge in the form of annotations is presented in [37]. Assuming word w_t has a set of m_t annotations $(A_{t,1}, A_{t,2}, \ldots, A_{t,m_t})$, then given a sequence of T training words $w_1, w_2, \ldots, w_T$, the objective of the model is to maximize the following:

$$\frac{1}{T}\sum_{t=1}^{T} \sum_{-C\leq j\leq C, j\neq 0} log\ P(w_t|w_{t+j}) + \sum_{0\leq k\leq M_t} log\ P(w_t|A_{t,k}) \tag{3}$$

where C is the size of the context window, w_t is the target word, w_{t+j} is a context word, and $A_{t,k}$ is the k^{th} annotation of word w_t. The quality of the word embeddings was evaluated on semantic relatedness of domain aliases in a sparse cybersecurity vulnerabilities domain and was shown to outperform other common embeddings by a large magin [37].
Concept embeddings can be improved by incorporating hierarchical information from large KBs or ontologies. This is the main idea in [19] where a distance metric is learned for each category (internal node) and entity-context similarity is measured under the aggregated metrics of all relevant categories. The goal is to find an entity (or concept) representation that is useful for predicting other entities in its *context* where context is defined by the entity heirarchical structure. More specifically, given an ontology and a set of entity pairs $\mathcal{D} = \{(e_T, e_C)\}$ where e_T, e_C denote target entity and context entity (appearing in e_T context), then for each entity e, the model learns both a "taget vector" $v_e \in \mathbb{R}^n$ and a "context vector" $\overline{v}_e \in \mathbb{R}^n$ by maximizing the objective:

$$\mathcal{L} = \frac{1}{|\mathcal{D}|} \sum_{(e_T,e_C)\in\mathcal{D}} log\ P(e_C|e_T) \tag{4}$$

where

$$P(e_C|e_T) = \frac{exp\{-d(e_C|e_T)\}}{\sum_{e\in\mathcal{E}} exp\{-d(e_C|e_T)\}} \tag{5}$$

and

$$d(e_C|e_T) = (v_{e_T} - \overline{v}_{e_C})^T\ M_{e_T,e_C}\ (v_{e_T} - \overline{v}_{e_C}) \tag{6}$$

where $M_{e,e'}$ is an aggregated metric of relevant categories along all inter-entity paths. The learned embeddings outperform other baselines on entity linking and entity in Wikipedia [19].

Work on distributed representations of concepts go back to the 1980's [17]. Earlier models were mostly based on viewing relations as linear transformations of concepts in some feature space. For example, Linear Relational Embeddings (LRE) [31] are based on triplet completion via predicting the third element given the two other elements in the (concept1, relation, concept2) triplet. LRE represents each concept as a learned vector in a Euclidean space and each relationship as a learned matrix that maps the first concept into an approximation of the second concept. The objective is to maximize the log probability of getting the right completion, summed over all training triplets and is given by:

$$D = \sum_{c=1}^{C} \frac{1}{k_c} log \frac{e^{\|R^c \cdot \mathbf{a}^c - \mathbf{b}^c\|}}{\sum_{\mathbf{v}_i \in \mathcal{V}} e^{\|R^c \cdot \mathbf{a}^c - \mathbf{v}_i\|}} \tag{7}$$

where k_c is the number of triplets having the first two terms equal to the ones of c but the third term different. Relations are modeled by linear operations in the feature space. LRE is shown to generalize well. It infers new triplets to complete missing links in the data. However, the method is limited to binary relations but has been extended in [30] by adding a layer of non-linear units to represent other types of relations.
More recently, distributed representation models based on distance in non-Euclidean spaces have been proposed [28]. This was motivated by noting that symbolic datasets often have complex latent heirarchical structures which cannot be captured by Euclidean vector spaces. The approach in [28] embeds symbolic data into an n-dimensional Poincaré ball. The hyperbolic geometry of this non-Euclidean space facilitates the learning of sparse representations of symbolic data, using Riemannian optimization, and simultaneously capture hierarchy and similarity. The learned embeddings, termed by authors as Poincaré embeddings, were shown to outperform Euclidean embeddings significantly on data with latent hierarchies. On the tasks of link prediction in network data and lexical entailment in textual data, Poincaré embeddings outperformed Euclidean-space embeddings in terms of representation capacity and generalization power. Another recent related work that appeared around the same time as the previous one focuses on learning embeddings of graphs in a non-Euclidean hyperbolic space [7]. Embeddings using this method were obtained for network data and visualizations were used to illustrate the usefulness of learning embeddings based on non-Euclidean distances. The authors further argue that power-law degree distributions, strong clustering and hierarchical community structures are naturally captured when graph data are embedded in a hyperbolic space. For domains with such distribution properties, these embeddings may be a better option to represent entities. The following section reviews some neural work on graph related problem.

3.2 Deep Models for Knowledge Graph Problems

Related work in this area can be broadly organized into four main categories:

- combining deep models with human structured knowledge
- creating concept/entity embeddings from textual and graph data
- knowledge graph traversal and completion (link prediction)
- learning and creating representations of graph kernels (subgraphs, nodes or motifs)

Along the first line of research is work on training DNNs with domain-specific features [8] and more recently a distillation framework that transfers knowledge in first-order-logic into neural networks [20] by combining distillation [18] and posterior regularization [13]. This work was generalized to other structured models including soft-logic rules in [21] where knowledge representation and regulated DNNs are jointly learned via bidirectional information transfer between the neural network and the structured constraints and optimizing weights using all components.
Along the second line of research is the early work on distributed representations of entities in a KB [5] where similarity between entities is defined per relation in terms of how likely the entities to be linked under this relation. It has also been proposed to embed relations and view them as transformations of entities in a common space [15], where transformations are interpreted as traversal operators. More recently, other approaches have been proposed to learn entity and relation embeddings such that the learned representations can be in different spaces since they are essentially different objects [23]. An interesting piece of work has recently proposed a deep architecture using attentive LSTM to combine structural and textual information about concept/entities to learn their embeddings [42]. It uses a gated mechanism to combine both types of information and weight each component to balance the two sources of information about an entity.
More recently, an impressive piece of work proposed to model temporal reasoning in dynamic graphs using deep networks [41]. This work extends the traditional triplet representation of KBs to include a time component and defines events (indexed by time) as relations (re)occurring between entities. Their approach combines previous work on point process models with deep learning and defines hidden state and embedding update functions to explicitly include the time component. They translate the intensity parameter λ in the point process model into a bilinear transformation in vector space in the deep model and propose a stochastic approximation to the intractable survival loss term [41].
A somewhat different line of work focuses on learning node, subgraph, and path embeddings for graphs and dynamic network data. For example, DeepWalk [34] models random walks in a graph using a SkipGram [25] model and extends related NLP problems like language modeling to walks in a graph. The learned latent representations are used in multi-label network classification tasks and were shown to improve F1 scores to up to 10% comparing with competing methods. Node2vec [14] learns continuous features of nodes by maximizing the likelihood of preserving the node's neighborhood in the network. Neighborhoods of nodes are sampled using a biased random walk procedure which smoothly interpolates breadth first and depth first search. LINE [40] is a large-scale information network embedding that preserves both first-order proximity and second-order proximity properties of the network, where first-order proximity refers to local pairwise edges and second-order proximity is defined by similarity of the nodes' neighborhood structures.
A framework for learning convolutional neural networks for arbitrary graphs is presented in [29] where a sequence of nodes is selected from the graph via a graph labeling [1] procedure, then for a subset of the nodes a local neighborhood graph is constructed and normalized, the normalized neighborhoods are then used as receptive fields in an existing CNN architecture. This approach is effectively learning a convolution-like function and has performed competitively with state-of-the-art graph kernel methods on graph classification tasks for different types of real data graphs [29].

[1] Graph labeling is a graph-theoretic term, not to be confused with labeling as in predicting labels in a supervised classification task.

We note that most work involving learning concept, document, and graph distributed representations for various tasks focus on *asymmetric* measures of similarity whereas our task requires capturing an asymmetric function (dependency between concepts or documents which is a directed relation). In Section 7 I briefly explain how to use ideas from related work to improve neural preprequisite prediction.

4 Data

The datasets consist of course descriptions (often including a short syllabus) from three different universities: MIT, CMU, and Princeton. This is a subset of the data used in [43] and is available here. The dataset also includes prerequisite links between courses offered by the same institution. To frame the problem as a classification problem, the course description of two courses is concatenated in the correct and reversed order to create positive and negative examples respectively. This also results in a balanced dataset and a simpler classification problem. The classification problem can be viewed as judging whether a document is coherent or not where a coherent document presents prerequisite content earlier. The final dataset contains 2909 documents with average length of 2320 words. To approach the problem in a more realistic way that reflects the skewness and imbalance of the inherent classes (where only a very small subset of document pairs form positive examples), I created data where the negative class also include (in addition to content in reversed order of dependency) the following: unrelated documents, related documents with no dependency relation, and the same document content repeated. These experiments will be reported in the next milestone. See Section 7 for more on planned work.

5 Model and Implementation

The model is a deep convolutional neural network with max-pooling and multiple filters of *different sizes* (implemented using Keras 1.0 Merge layer). The model has 2 convolutional layers followed by max-pooling followed by two fully connected layers. ReLU activations were used and a softmax for the final layer output.
Different variants of the CNN were used: one with fixed-size convolution windows, and others with multiple convolution windows of size 3, 4, 5, and 7. 128 filters (for each size) were used in each variant. The number of filters and window sizes (with the exception of 7) were based on the work on sentence classification using CNNs [22]. The model was trained for 20 epochs on categorical crossentropy loss using 80% of the data with mini-batch size of 64 and tested on the remaining 20%. Batch normalization was not performed. The filters with two different window sizes (3 and 5) variant significantly outperformed others in all experiments. The model where the embeddings were fine-tuned performed better despite the small size of the dataset.
In transfer learning experiments, the model is trained on data from a specific source and tested on data from another source, for example train on MIT courses and test on CMU courses. Note that CMU dataset contains CS and mathematics courses only whereas MIT dataset includes courses from other STEM fields and Princeton dataset contains mathematics courses only. We have not explicitly trained the model to do transfer learning but to compare with the non-neural baseline which explicitly enables transfer learning by using a large coverage concept graph as an inter-lingua, we include results on such experiments in the next section. We expect some degree of transfer through the use of distributed representations of words in the course description.
We report categorical classification accuracy results in the next section. For this specific dataset

and task (with tunable embeddings), RMSProp optimizer resulted in better accuracies (86%) compared to Adam (83%). The evaluation metric used is Keras built-in categorical accuracy which is defined as:
`K.mean(K.equal(K.argmax(y_true, axis=-1), K.argmax(y_pred, axis=-1)))` whereas in Keras binary classification accuracy 0.5 is the threshold to distinguish between classes and is defined as `K.mean(K.equal(y_true, K.round(y_pred)))`.
For word embeddings, the pretrained GloVe embeddings [33] with 100 dimensions were used and tuned. Vectors of unknown words were randomly initialized.
The model was implemented in Python3 using Keras 1.0 and is compatible with Python3.5. I referred to this tutorial on the Keras blog on text classification to implement the model. Code is available on `https://github.com/alraisif/neuralPrereq`. Further implementation and environment details are provided in Appendix B. The code takes 5 hours to run (using the combined dataset) on an AWS EC2 g2.2xlarge Ubuntu 16.4 instance. Results are presented next.

6 Results

Table 2 presents results of the implemented model. We cannot directly compare to the non-neural classification CGL baseline [43] since they evaluate on Mean Average Precision (MAP). Also, our data is more balanced than the data in [43] but they only apply the classification approach to within-institution prerequisite prediction. The average MAP result for MIT, CMU, and Princeton within-institution prerequisite prediction is 0.33 [43].

model	categorical accuracy
CNN fixed filter size fixed embeddings	0.4905
CNN multiple filter sizes fixed embeddings	0.8141
CNN fixed filter size trainable embeddings	0.5112
CNN 4 conv windows trainable embeddings	0.8417
CNN 2 conv windows trainable embeddings	0.8571

Table 2: Binary Classification Task. All models with 128 filters (for each window size)

Table 3 presents transfer learning results where the best performing model above is trained on data from a given source and tested on data from a different source. The non-neural CGL baseline [43] reports a MAP of 0.35 for MIT as source and CMU as target, and MAP of 0.45 for MIT as source and Princeton as target, both using the ranking approach.

source	target	categorical accuracy
MIT	Princeton	0.7111
MIT	CMU	0.6467
CMU	Princeton	0.5611
Princeton	CMU	0.6067

Table 3: Transfer Learning Experiments

7 Next Milestone

The work on concept graph learning [43] includes limited experiments with word embeddings as a choice for representing concepts. Their choice of embeddings [2] may not be the best for the problem. Planned extensions of this work include trying other word embeddings including GloVe and ConVec [12] *concept* embedding. To improve interpretability of the latent concept space, we plan to combine the current discrete word-based concept representation with distributed embeddings by concatenating the current bag-of-concept vector of a course with the output of a neural network encoding the course content and use combined representations as features in the non-neural CGL [43] baseline.

The dataset creation process resulted in a balanced data set. The prerequisite prediction task was defined as a binary document classification task. The next step would be to solve the following more realistic and challenging classification problem where the task is to predict whether the input document is coherent or incoherent on a less balanced dataset where every training instance is constructed from one of the following: course and prerequisite documents (positive examples), reversed pair of course and prerequisite document (negative example), two unrelated documents (negative example) and two related (topic or content-wise) but not dependent documents (negative example), and a document consisting of the same content repeated twice (negative example).

In the next milestone, we also intend to implement an RNN with attention as it encodes dependencies in a sequence which is exactly the kind of information needed in this task. We can generalize language modeling to model sequences of concepts and distinguish meaningful sequences from incoherent ones.

Finally, instead of approaching the problem using one network to do binary classification we can train a Siamese network [6, 16, 38] to encode each input document independently and then use an asymtric function to compute the difference between the two representations and use that result to map them to the correct semantic space (dependent or not dependent).

References

[1] R. Agrawal, B. Golshan, and E. E. Papalexakis. Toward data-driven design of educational courses: A feasibility study. In *Proceedings of the 9th International Conference on Educational Data Mining, EDM 2016, Raleigh, North Carolina, USA, June 29 - July 2, 2016*, page 6, 2016.

[2] R. Al-Rfou, B. Perozzi, and S. Skiena. Polyglot: Distributed word representations for multilingual NLP. *CoRR*, abs/1307.1662, 2013.

[3] R. Antonova, J. Runde, M. H. Lee, and E. Brunskill. Automatically learning to teach to the learning objectives. In *Proceedings of the Third (2016) ACM Conference on Learning @ Scale*, L@S '16, pages 317–320, New York, NY, USA, 2016. ACM.

[4] R. Baker. Data mining for education. In P. Peterson, E. Baker, and B. McGaw, editors, *International Encyclopedia of Education*. Elsevier, Oxford, third edition edition, 2010.

[5] A. Bordes, J. Weston, R. Collobert, and Y. Bengio. Learning structured embeddings of knowledge bases. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, AAAI'11, pages 301–306. AAAI Press, 2011.

[6] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah. Signature verification using a "siamese" time delay neural network. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 6*, pages 737–744. Morgan-Kaufmann, 1994.

[7] B. Chamberlain, J. Clough, and M. Deisenroth. Neural embeddings of graphs in hyperbolic space. *CoRR*, abs/1705.10359, 2017.

[8] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, Nov. 2011.

[9] A. T. Corbett and J. R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4(4):253–278, Dec 1994.

[10] B. Dhingra, H. Liu, R. Salakhutdinov, and W. W. Cohen. A comparative study of word embeddings for reading comprehension. *CoRR*, abs/1703.00993, 2017.

[11] S. Doroudi, K. Holstein, V. Aleven, and E. Brunskill. Sequence matters, but how exactly? a method for evaluating activity sequences from data, 2016.

[12] E. M. Ehsan Sherkat. Vector embedding of Wikipedia concepts and entities, 2017.

[13] K. Ganchev, J. a. Graça, J. Gillenwater, and B. Taskar. Posterior regularization for structured latent variable models. *J. Mach. Learn. Res.*, 11:2001–2049, Aug. 2010.

[14] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. *CoRR*, abs/1607.00653, 2016.

[15] K. Guu, J. Miller, and P. Liang. Traversing knowledge graphs in vector space. *CoRR*, abs/1506.01094, 2015.

[16] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, CVPR '06, pages 1735–1742, Washington, DC, USA, 2006. IEEE Computer Society.

[17] G. E. Hinton. Learning distributed representations of concepts. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, pages 1–12. Hillsdale, NJ: Erlbaum, 1986.

[18] G. E. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.

[19] Z. Hu, P. Huang, Y. Deng, Y. Gao, and E. P. Xing. Entity hierarchy embedding. In *ACL (1)*, pages 1292–1300. The Association for Computer Linguistics, 2015.

[20] Z. Hu, X. Ma, Z. Liu, E. H. Hovy, and E. P. Xing. Harnessing deep neural networks with logic rules. *CoRR*, abs/1603.06318, 2016.

[21] Z. Hu, Z. Yang, R. Salakhutdinov, and E. P. Xing. Deep neural networks with massive learned knowledge. In *EMNLP*, pages 1670–1679. The Association for Computational Linguistics, 2016.

[22] Y. Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014.

[23] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 2181–2187. AAAI Press, 2015.

[24] D. Merkel. Docker: Lightweight linux containers for consistent development and deployment. *Linux J.*, 2014(239), March 2014.

[25] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[26] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546, 2013.

[27] S. K. Mohamad and Z. Tasir. Educational data mining: A review. *Procedia - Social and Behavioral Sciences*, 97:320 – 324, 2013. The 9th International Conference on Cognitive Science.

[28] M. Nickel and D. Kiela. Poincaré embeddings for learning hierarchical representations. *CoRR*, abs/1705.08039, 2017.

[29] M. Niepert, M. Ahmed, and K. Kutzkov. Learning convolutional neural networks for graphs. *CoRR*, abs/1605.05273, 2016.

[30] A. Paccanaro. Learning distributed representations of high-arity relational data with non-linear relational embedding. In *Proceedings of the 2003 Joint International Conference on Artificial Neural Networks and Neural Information Processing*, ICANN/ICONIP'03, pages 149–156, Berlin, Heidelberg, 2003. Springer-Verlag.

[31] A. Paccanaro and G. E. Hinton. Learning hierarchical structures with linear relational embedding. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 857–864. MIT Press, 2002.

[32] A. Peña Ayala. Review: Educational data mining: A survey and a data mining-based analysis of recent works. *Expert Syst. Appl.*, 41(4):1432–1462, Mar. 2014.

[33] J. Pennington, R. Socher, and C. D. Manning. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[34] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 701–710, New York, NY, USA, 2014. ACM.

[35] A. N. Rafferty, E. Brunskill, T. L. Griffiths, and P. Shafto. Faster teaching via pomdp planning. *Cognitive Science*, 2015.

[36] C. Romero and S. Ventura. Educational data mining: A survey from 1995 to 2005. *Expert Systems with Applications*, 33(1):135 – 146, 2007.

[37] A. Roy, Y. Park, and S. Pan. Learning domain-specific word embeddings from sparse cybersecurity texts. *CoRR*, abs/1709.07470, 2017.

[38] R. Salakhutdinov and G. Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009.

[39] D. Shahaf and C. Guestrin. Connecting the dots between news articles. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 623–632, New York, NY, USA, 2010. ACM.

[40] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 1067–1077, Republic and Canton of Geneva, Switzerland, 2015. International World Wide Web Conferences Steering Committee.

[41] R. Trivedi, H. Dai, Y. Wang, and L. Song. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3462–3471, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[42] J. Xu, K. Chen, X. Qiu, and X. Huang. Knowledge graph representation with jointly structural and textual encoding. *CoRR*, abs/1611.08661, 2016.

[43] Y. Yang, H. Liu, J. Carbonell, and W. Ma. Concept graph learning from educational data. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, WSDM '15, pages 159–168, New York, NY, USA, 2015. ACM.

[44] M. V. Yudelson, K. R. Koedinger, and G. J. Gordon. Individualized bayesian knowledge tracing models. In H. C. Lane, K. Yacef, J. Mostow, and P. Pavlik, editors, *Artificial Intelligence in Education: 16th International Conference, AIED 2013, Memphis, TN, USA, July 9-13, 2013. Proceedings*, pages 171–180, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

Appendix A: Mapping from Document Space to Feature Space

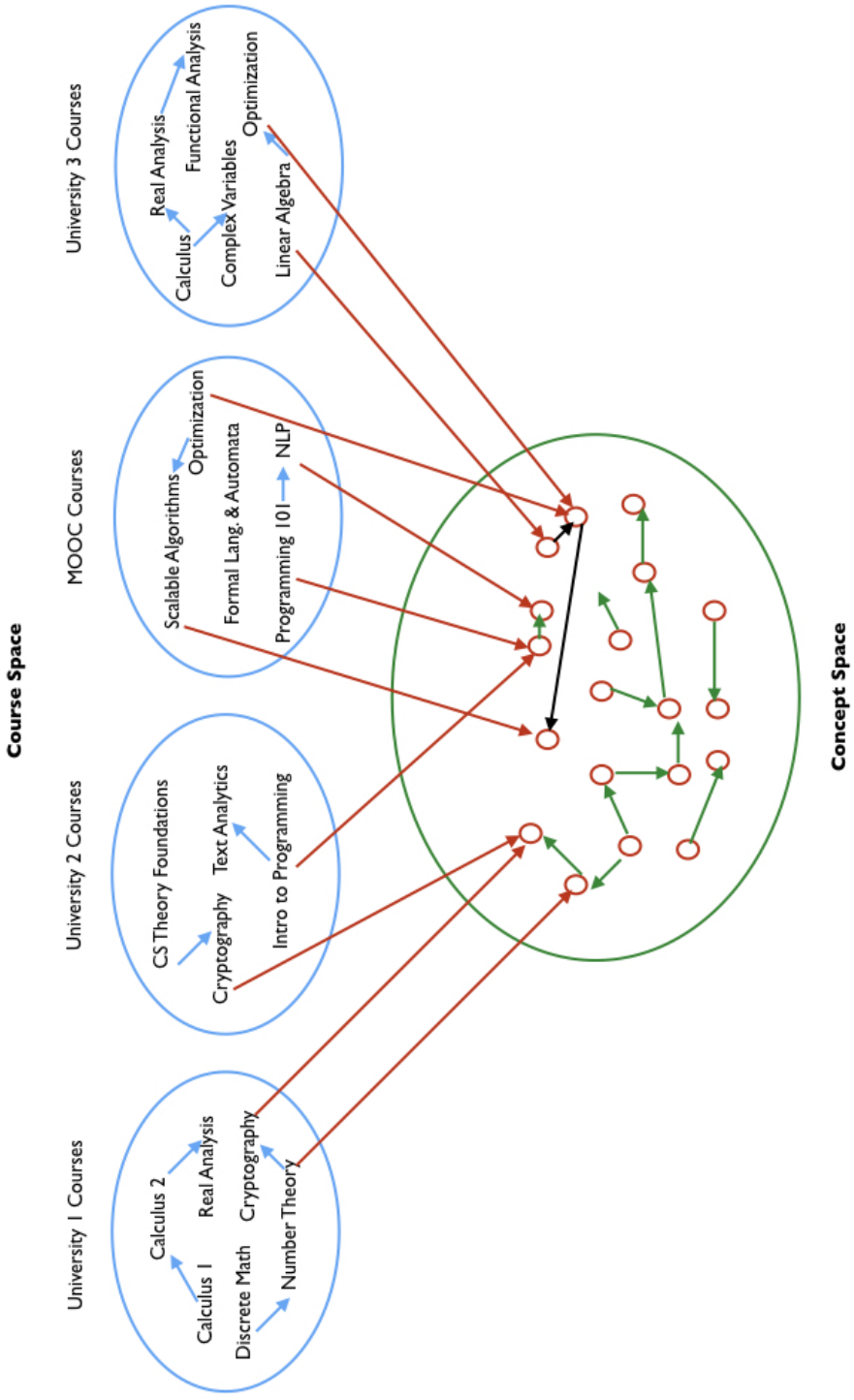

Figure 1: Transferring edge prediction from course space to concept space and projecting back to the course space

Appendix B: Environment Setup and Running Code

The code was written in Python3 with Keras using Theano as a backend. The code is compatible with python3.5 and Keras 1.0 but not with earier The code was run on AWS EC2 g2.2xlarge Ubuntu 16.4 instance. All required packages were installed using miniconda3 following the instructions here. The provided can be simply run using python3 without chaning any input configuration. A separate .py file was provided for the main experiment and additional transfer learning experiments.

Environment:
The following change was made to Keras text.py preprocessing code which can be found in `keras/preprocessing/text.py`:

replace the following line

```
text = text.translate(maketrans(filters, split*len(filters)))
```

with

```
try :
    text = unicode(text, "utf-8")
except TypeError:
    pass
translate_table = {ord(c): ord(t) for c,t in  zip(filters, split*len(filters)) }
text = text.translate(translate_table)
```

Depending on python version installation and the environment configuration, it may be necessary to set `LA_ALL` variable for proper handing of utf-8 encoding of input:

```
export LC_ALL=de_DE.UTF8
```

I recommend using docker [24] to create an image of the environment when testing the submitted code.