One-Class Order Embedding for Dependency Relation Prediction

Meng-Fen Chiang  
Living Analytics Research Centre  
Singapore Management University  
mfchiang@smu.edu.sg

Ee-Peng Lim  
Living Analytics Research Centre  
Singapore Management University  
eplim@smu.edu.sg

Wang-Chien Lee  
The Pennsylvania State University  
University Park, PA 16802, USA  
wlee@cse.psu.edu

Xavier Jayaraj Siddarth  
ASHOK  
Living Analytics Research Centre  
xaviera@smu.edu.sg

Philips Kokoh PRASETYO  
Living Analytics Research Centre  
pprasetyo@smu.edu.sg

ABSTRACT
Learning the dependency relations among entities and the hierarchy formed by these relations by mapping entities into some order embedding space can effectively enable several important applications, including knowledge base completion and prerequisite relations prediction. Nevertheless, it is very challenging to learn a good order embedding due to the existence of partial ordering and missing relations in the observed data. Moreover, most application scenarios do not provide non-trivial negative dependency relation instances. We therefore propose a framework that performs dependency relation prediction by exploring both rich semantic and hierarchical structure information in the data. In particular, we propose several negative sampling strategies based on graph-specific centrality properties, which supplement the positive dependency relations with appropriate negative samples to effectively learn order embeddings. This research not only addresses the needs of automatically recovering missing dependency relations, but also unravels dependencies among entities using several real-world datasets, such as course dependency hierarchy involving course prerequisite relations, job hierarchy in organizations, and paper citation hierarchy. Extensive experiments are conducted on both synthetic and real-world datasets to demonstrate the prediction accuracy as well as to gain insights using the learned order embedding.

CCS CONCEPTS
• Computing methodologies → Machine learning; Learning to rank;

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6172-9/19/07... $15.00
https://doi.org/10.1145/3331184.3331249

1 INTRODUCTION
Motivation. Graph is a widely adopted data representation in diverse real-world scenarios. Learning representation of nodes and edges in graph structures has been well studied for many graph analytics applications [15][27][4]. In form of low-dimensional vectors, embedding representation typically preserves the degree of network proximity [23][22], as various applications, such as node classification and link prediction, requires the learned representations to capture properties of the graph structure. Nevertheless, little attention has been paid to exploring order embeddings that preserve relative order among a given set of nodes in an ordering hierarchy. Such order embeddings are expected to encode rich information of the hierarchical structure, and hence can be used to predict missing dependency links.

Indicative representations of text and partial ordering relations amongst entities, such as hierarchical structure, can capture semantic and structural information embedded among them. Entities with positive or negative ordering relations tend to be semantically related to each other, and thus are comparable. However, not all pairs of entities have an ordering relationship. Specifically, semantically unrelated pairs of entities are not comparable and thus should not have any ordering relations between them. For example, a course “machine learning” depends on several prerequisite courses, including “data structure”, “theory of computation”, and “artificial intelligence” as they are background to the machine learning course, as indicated in many computer science curricula. Moreover, intuitively, “probabilistic models” is semantically closer to “fundamentals of probability” than it is to “geographical information systems spatial databases”. Thus, to establish the relatedness between these courses, we may explore the semantic proximity of title and textual content of these courses. Indeed, most of today’s representation learning techniques aim to learn entity representations that are semantics-preserving, i.e., semantically similar entities are mapped into a nearby area in the semantic embedding space. In this paper, we argue that it is also essential for representation of entities to be order-preserving, i.e., antisymmetric relations between two entities are captured in the embedding space. There are relatively very little research on order-preserving embedding. One such order-preserving embedding technique, developed by Vilnis...
and McCallum, assigns each entity a location in the embedding space such that the relative positions of entities to the origin of the representation space (i.e., zero coordinates) determine the partial ordering among them [26].

*** wlee: In addition to motivating the technical aspect of embedding techniques, I think it’s important to motivate the application aspect, i.e., the needs and application of dependency relationships.

In this paper, we explore a compositional approach to incorporate both semantics- and order-preserving embedding to predict dependency relations of entity pairs. Past research on order embeddings [24] and [2] have shown to be effective for word hypernym classification and textual entailment tasks where only entities themselves and their relations are represented. The textual and content features of these entities are largely neglected. To the best of our knowledge, no prior research has attempted to combine both semantic- and order-preserving embeddings for prediction of dependency relations.

Objectives. This research focuses on two objectives: (1) designing a new framework to accurately recover missing dependency relations, which is essential in automatic completion of knowledge bases; (2) applying the proposed framework on real-world datasets, such as course dependency hierarchy, job hierarchy, and paper citation hierarchy, using the automatically learned order embeddings to gain insights on hierarchical ordering. Both objectives are to be achieved in solving the following fundamental research problem:

Problem: (Dependency Relation Prediction). Given a set of observed entity pairs with dependency relations, determine whether an unseen entity pair have a positive dependency relation or not.

Overview of Our Proposed Approach. We address the dependency relation prediction problem as a classification task and propose a two-step framework as shown in Figure 1. The Step 1 of the framework learns the order embedding for entity pairs. Here, the order embedding is a low-dimensional vector that can be used to differentiate ordering relations among entities according to their positions in embedding space. Specifically, we learn the ordering hierarchy (consists of order embeddings) for the entire set of entities. However, not every entity pair is comparable. To determine ordering relations for comparable entities, the Step 2 of the framework combines order embedding and word embedding of entities. Examples of the latter include GloVe [21] and Word2Vec [13]. Next, the framework adopts a compositional approach to combine features from word embeddings and order embeddings for a pair of entities, to train the dependency relation classifiers, which allows us to leverage on both textual content and order embedding information.

*** The paragraph above is a bit confusing to me! Do you mean to learn order embedding as a vector associated with an edge and also learning embeddings for nodes? Or are you actually learning node embeddings only but those nodes should preserve ordering relation? Could you clarify it.

There are two major research challenges in the dependency relation prediction problem and the proposed framework. Firstly, ordering hierarchies are often incomplete as not all ordering relations can be observed in many real applications. Under the open world assumption [14], these non-existing relations can be either negative or unknown [15][27][4]. With only partial ordering relations observed and no knowledge of negative relations, the dependency relation prediction is therefore a one-class problem. To determine whether a dependency relation between two entities holds, we thus need good negative sampling strategies to collect diverse and highly likely negative relations between entities. Hence, the first research challenge lies in selecting a subset of “good” negative samples from a massive set of unobserved relation candidates.

The second research challenge is to demonstrate the advantages of fusing both order-preserving and semantics-preserving representations of entities for dependency relation prediction. Our research seeks to look into a detailed evaluation of our proposed compositional approach in comparison with two baseline approaches: (i) One focuses on learning semantic features from textual content, and uses the semantic features to determine dependency relations; (ii) Another approach is to use order embedding as features to determine dependency relations. To the best of our knowledge, such a study has not been conducted before despite its importance to knowledge base completion.

Contributions. The contributions of our work are summarized as follows:

- With vector order embedding model [24] used for learning ordering hierarchy among entities, we propose several negative sampling strategies based on graph centralities to effectively address the one-class problem, by using a training set of positive dependency relations and sufficient negative samples for order embedding learning.
- We propose the dependency relation classification framework that fuses both semantic and order embeddings of entities.
- We conduct extensive experiments to demonstrate the effectiveness of proposed framework using both synthetic ordering hierarchies and several real-world datasets.

2 RELATED WORK

Relational Learning. The goal of relational machine learning is to infer relationships between objects and answer queries [15][27][4]. Given a database of partially annotated relations, relational machine learning aims to learn rules or models to answer questions on unseen relations. This task arises in many settings such as biological pathways, analysis of question answering. A variety of techniques from Statistical relational models (SRL) community are proposed to tackle such tasks with partially annotated relations. SRL can be fundamentally divided into two categories: latent feature models and observable models. Markov Logic Nets (MRF) learns a set of
rules and weights to infer all unobserved relations jointly by maximiz- ing probability. Embedding approaches, e.g., RESCAL, learn embeddings to predict relations. For example, the neural embedding models predict scores based on subjects, objects, and predicate embeddings[6][3][17][11][16]. Graph mining approaches on the other hand explore graph features, common ancestor or random walks, to predict relations. Ultimately, Google knowledge vault projects, automatic knowledge graph construction from the Web, fusion latent and observable models to improve the modeling power.

**Prerequisite Relation Learning.** Recent work has studied prerequisite relations learning among of educational concept map [12][10][19][20][5][1][9]. Prerequisite chains play an important role in curriculum planning and reading list generation [8][7]. Liu et al. focus on learning prerequisite relation for relation prediction among a set of concepts in university courses [12], while Liang et al. focus on recovery of prerequisite relations [10]. Liu et al. address CGL (concept graph learning) for two-level prerequisite learning, course level and concept level. Given observed course-level prerequisite rations, CGL.Class and CGL.Rank explicitly learn full prerequisite relations between concepts from course pair relations. Pan et al. [19] propose to automatically identify all course concepts from online MOOCs video clips [20]. Chen et al. propose a structural EM to learn an optimal Bayesian Network structure, representing the dependency relations among skills, which best explains the distribution of student performance data [5]. However, the scalability and efficiency of the learning mechanism remain unclear as the data consists of only a handful of students, examine items and skill variables. Liang et al. explore applicability of active learning to address the issue of limited training data for prerequisite classification on pairs of concepts [9]. They adopt pool-based active learning scenario to train a classifier with a substantially small dataset by incorporating four types of “valuable” unlabeled instances in particular prioritized way at each iteration. To the best of our knowledge, there is no existing work on prerequisite relation learning which tackles relation recovery via representation learning from partial orders of structural data.

**Order Embedding Learning.** Order embeddings are shown to be effective for word hypernym classification, image-caption ranking and textual entailment [24][26][2][25]. Vendrov et al. [24] propose to learn asymmetric relationships with deterministic vector order embeddings (VOE) of non-negative coordinates with partial order structure from incomplete data. Due to its limitation of expressiveness of deterministic vector order embeddings, recent work incorporate uncertainty in learning order representation to enrich expressiveness and enable predict with uncertainty, such as probabilistic extensions of order embeddings [26][2] and box lattice representation of order embeddings [25]. Athiwaratkun et al. introduce density order embedding (DOE) to model hierarchical data via encapsulation of probability densities [2]. In particular, they propose a new loss function, graph-based negative sample selections, and a penalty relaxation to induce soft partial orders.

### 3 ORDER EMBEDDING LEARNING AND NEGATIVE SAMPLING STRATEGIES

#### 3.1 Vector-Based Order Embedding

There are a few order embedding methods proposed in the literature [2, 24, 26]. They all seek to assign a geometric representation to each entity such that the partially ordered entities are mapped into a latent space where certain geometric relationship between them are preserved. In this paper, we follow [24] to formalize the notion of partially ordered entities as follows.

**Definition 3.1.** (Partially Ordered entities) A partially ordered set of entities \( V \) has a binary order relation \( \preceq \) such that for \( v_i, v_j, v_k \in V \), the following properties hold: (1) \( v_i \preceq v_i \) (reflexivity), (2) if \( v_i \preceq v_j \) and \( v_i \neq v_j \), then \( v_j \npreceq v_i \) (antisymmetry), and (3) if \( v_i \preceq v_j \) and \( v_j \preceq v_k \), then \( v_i \preceq v_k \) (transitivity).

Motivated by Vendrov and others [24], we want to learn a mapping \( f \) from \( V \) into a **Vector-based Order Embedding Space** \( Y \) where each entity is represented as a geometric point and the ordering relation between two entities is represented as a geometric relationship between the points representing the two entities. For entities with unknown order relations, we can then use their point representations in the embedding space \( Y \) to infer their order relations.

A crucial property of the mapping function \( f \), preserving the order relations in the vector-based order embedding space, is defined as follows:

**Definition 3.2.** (Order preserving) \( f : (V, \preceq) \rightarrow (Y, \succeq) \) is order-preserving if \( \forall v_i, v_j \in V, v_i \preceq v_j \iff f(v_i) \preceq f(v_j). \)

**Objective Function.** To learn a good order-preserving mapping \( f \), Vendrov et al. define the geometric relation between two entities \( v_i \) and \( v_j \) in the embedding space based on the conjunction of total order on each dimension \( d \) of embedding space \( Y \). That is, \( f(v_i) \preceq f(v_j) \) if and only if \( v_i,d \succ v_j,d, \forall 1 \leq d \leq N_Y \) where \( N_Y \) is the dimension size of \( Y \).

The above hard constraints can be implemented using a loss function that penalizes order violations in \( Y \in \mathbb{R}^{N_Y} \) for the given set of ordered pairs \( v_i \preceq v_j \):

\[
d(v_i, v_j) = ||\max(0, f(v_j) - f(v_i))||^2
\]

where \( d(v_i, v_j) = 0 \) if \( f(v_i) \preceq f(v_j) \) according to the conjunction of total orders; and \( d(v_i, v_j) > 0 \) if there is an order violation.

Given a set of positively ordered relations \( E^+ \) and a set of negatively ordered relations \( E^- \), the objective function to learn an order-embedding mapping \( f \) is defined as a max-margin loss that encourages positively ordered relations to have zero penalty, and negatively ordered relations to have penalty greater than a margin:

\[
O = \sum_{(v_i, v_j) \in E^+} d(f(v_i), f(v_j)) + \sum_{(v_i, v_j) \in E^-} \max\{0, \alpha - d(f(v_i), f(v_j))\}
\]

**3.2 Negative Sampling Strategies**

**Problem Analysis.** Let \( E^+ = \{(v_i, v_j)\} \) be the set of observed positive dependency relations, and \( E^- \) be the set of negative dependency relations to be determined. We derive \( P(v_i, v_j) \), the likelihood of \((v_i, v_j) \notin E^+ \) being a negative dependency relation. If \( P(v_i, v_j) = 1 \) or 0, it suggests \( v_i \preceq v_j \) or \( v_i \npreceq v_j \) with full certainty. Hence, in negative sampling, two questions should be answered: (1) what is the size of negative samples \( |E^-| \)? and (2) what is the sampling distribution \( P(v_i, v_j) \)?

For the first question, there are essentially the **full approach** and **subsampled approach** [28]. The full approach determines \( E^- \) to be
the set of all unobserved relations, i.e., \( E^- = \{(v_i, v_j) | (v_i, v_j) \notin E^+\} \). The full approach can be computationally expensive especially when the observed dependency relations are extremely sparse, i.e., the size of \( E^- \) can be exponentially greater than the size of \( E^+ \) [28]. Moreover, it may misjudge some unobserved positive relations as negative. The subsampled approach, on the other hand, aims to select a reasonable subset of the entire unobserved relations as negative samples. The subsampled approach is thus a necessary approximation of the full approach. The size of subsampled negative samples is advised to be a constant multiple of the positive samples, i.e., \( |E^-| = O(|E^+|) \) [28].

**Simple Sampling Strategies (S1, S2):** For the second question, one can assume a uniform distribution for \( P(v_i, v_j) \) for \( v_i \leq v_j \) not observed in \( E^+ \). In other words, for each positive dependency relation \( (v_i, v_j) \in E^+ \), we generate \( c \) negative pairs of either \((v_i, v_k) \notin E^+ \) or \((v_k, v_j) \notin E^+ \) where \( v_k \)'s are randomly selected from \( V \). This is also referred to Simple Strategy 1 (S1). Another simple negative sampling strategy is to reverse each positive dependency relation \( (v_i, v_j) \) and use the reverse relation as negative sample. We call this the Simple Strategy 2 (S2). Moreover, we explore three aspects of graph structure to derive negative samples: (1) local neighborhood, (2) global neighborhood, and (3) descendant structure. Table 1 gives an overview of the proposed sampling strategies.

**Local Neighborhood-based Sampling Strategies (L1-L6):** The key assumption behind local neighborhood-based negative sampling is that entities with large out-degree should be ones that many other entities may depend on or precede many other entities, while entities with large in-degree should be ones that depend on many other entities (or be precede by many other entities). Hence, the likelihood of an unobserved relation \((v'_i, v'_j)\) being negative can be estimated by the out-degree and in-degree of \( v'_i \) and \( v'_j \), respectively. We therefore propose L1-L6 strategies as summarized in Table 1. Given a positive dependency relation \((v_i, v_j)\), the strategies select negative samples as follows:

- **L1** gives \( v'_i \) with high in-degree (e.g., a senior-level job or course) a high probability for \((v'_i, v'_j) \in (V \times V) \setminus E^+ \) to be used as a negative sample.
- **L2** gives \( v'_j \) with high out-degree (e.g., an entry-level job, or an advanced course) a high probability for \((v_i, v'_j) \in (V \times V) \setminus E^+ \) to be used as a negative sample.
- **L3** considers \((v'_i, v'_j)\) a high probable negative sample if \( v'_i \) has high in-degree and \( v'_j \) has high out-degree.
- **L4-L6** follow the same intuition of L1-L3 respectively except that high relative out-degrees and in-degrees are used instead.

In the negative sampling process, we follow the sampling with replacement scheme, where the selected negative order relations are not removed from the population \( V \times V \setminus E^+ \). In other words, each relation candidate is independently drawn from the full set of unobserved negative relations according to the sampling distributions controlled by one of L1-L6 until the expected sample size \(|E^-_v| = c \times |E^+_v| \) (where \( c \) is a constant) is reached.

**Global Neighborhood Sampling Strategies (G1-G6):** Instead of deriving the probability of sampling alternatives of \( v_i \) and \( v_j \) to form negative samples using their local neighborhood properties (i.e., in-degree and out-degree), the global neighborhood sampling considers global neighborhood properties (i.e., ancestors and descendants). We define the descendants (or ancestors) to be entities that can be reached transitively by following the \( \leq \) relation (or inverse of \( \leq \) relation). These strategies assume that: (i) a node \( v_j \) with many ancestors suggests that \( v_j \) is unlikely to be ordered before others. Hence, \((v_j, v_i)\) is likely to be a negative order relation; (ii) a node \( v_j \) with many descendants suggests that it is unlikely to be ordered after others. Hence, \((v_i, v_j)\) is likely to be a negative order relation. Based on the above assumptions, we derive the sample strategies G1-G6 in Table 1.

**Descendants-Constrained Sampling Strategy (D0-DG6):** Other than measuring the likelihood for a relation to occur, another idea is to explore the relevance between a pair of entities in the sampling process. Athiwaratkun et al. introduced a structural constraint on relations by their relative positions in a network [2]. Specifically, for a node \( w \) with at least two descendants, they proposed to randomly select \( v_j \) \in \( T(w) \), where \( T(w) \) denotes all descendants of \( w \), and then randomly select \( v_i \) \in \( T(w) \setminus T(v_j) \). \((v_i, v_j)\) is taken as a negative sample if \((v_i, v_j) \notin E^+ \). In this manner, \( w \) is intuitively the common ancestor of \( v_i \) and \( v_j \) subject to \( v_i \) is not a descendant of \( v_j \). In essence, this structural constraint suggests a relevant but not close relations between \( v_i \) and \( v_j \). We refer to such structural constraint as descendants constraint. We incorporate such descendant constraint into previously defined sampling strategies. Each strategy subject to descendants constraint thus has a limited sample population. Table 1 depicts the specification of D0-D6. D0 is the original negative selection strategy proposed in [2]. Note that the set of unobserved relations subject to descendants constraint is a subset of the full unobserved relations. To sample negative relations by any of descendants-constrained strategy, we go through every node \( w \) with at least two descendants. For each node \( w \), we sample one unobserved relation \((v_i, v_j)\) according to \( P(v_i, v_j) \) among all possible unobserved relations \((v_i, v_j)\) that follow the descendants constraint. We repeatedly scan through every node \( w \) until the expected sample size \(|E^-_v| \) is reached. Likewise, the sampling strategies combined with the descendants constraint and global neighborhood (DG1-DG6) are summarized in Table 1.
Training/Test Graph Generation. Given a graph \( G \), we randomly determine a held-out dataset consisting of 20% relations for evaluation. The 20% held-out relations form a testing graph \( G_{tr} \), and the rest 80% relations form a training graph \( G_{tr} \).

### 4.2 Entity Ranking

We evaluate the quality of order embedding by the entity ranking task. Given a training graph \( G_{tr} = (V, E) \) with observed relations \( E \) and unobserved relations \( \bar{E} \), we aim to derive a ranked list of entities \( V \) according to their order embeddings learned from \( G_{tr} \).

**Order Embedding Learning.** Given a sample size \( n \), we collect a balanced training set for order embedding learning which consists of \( n \) observed dependency relations as positive training set (denoted as \( E^+ \)) and \( n \) unobserved dependency relations as negative training set (denoted as \( E^- \)). We follow the sampling with replacement scheme to collect positive and negative dependency relations. \( E^+ \) is determined by randomly drawn \( n \) relations from \( E \) in \( G_{tr} \), whereas \( E^- \) is drawn according to a specified negative sampling strategy from \( \bar{E} \). For instance, we may obtain \( E^- \) using 25 strategy by collecting each reversed observed relation in \( E \) from \( \bar{E} \), i.e., \( E^- = \{(v_j, v_i) | (v_i, v_j) \in E \} \). Finally, we learn order embeddings of \( \bar{E} \) from \( E^+ \) and \( E^- \) given embedding space dimensionality \( |D| \). The total number of learnable parameters is \( O = |V| \times |D| \).

**Ranking Quality Measurement.** We compare relative positions of entities \( V \) in the \( |D| \)-dimensional original space and the \( |D| \)-dimensional embedding space. To formally measure the ordering quality, we compute Kendall rank correlation coefficient \( \tau_d \in [-1, 1] \) along dimension \( d \in \{1,2\} \). Given the order relations \( R_{GT} \) as the ground-truth ranking along dimension \( d \) and the learned order embeddings \( R_{OE} \), we compute \( \tau_d \) with adjustment for ties as follows.

\[
\tau_d(R_{GT}, R_{OE}) = \frac{n_d - 2 \sum_{i=1}^{n_d} T_i}{n_d(n_d-1)}
\]

where \( n_d \) is the number of concordant pairs, \( T_i \) is the number of discordant pairs, \( n_d \) is the number of ties only in \( R_{GT} \), and \( n_t \) is the number of ties only in \( R_{OE} \). Finally, we take the average of rank correlation in both dimensions \( \mu(\tau) = \frac{1}{|D|} \sum_{d=1}^{2} \tau_d \) as the overall ranking quality.

**Results.** Table 3 gives a full comparison of order embeddings learned with various negative sampling strategies in terms of average rank correlations against the ground-truth ordering. Each point refers to the average rank correlation of one sampling strategy across ten graphs of the same density level, using 5-fold cross validations, i.e., average of 50 experiments. We compare the rank correlations of the compared order embeddings using graphs of three density levels \{sparse, moderate, dense\}. For ease of comparison, we discretize the average rank correlation \( \mu(\tau) \) into four levels: (i) tier 1 (0.75 ≤ \( \mu(\tau) \)), (ii) tier 2 (0.5 ≤ \( \mu(\tau) < 0.75 \)), (iii) tier 3 (0.25 ≤ \( \mu(\tau) < 0.5 \)), and (iv) tier 4 (\( \mu(\tau) < 0.25 \)).

We conclude the experimental results with three key observations. First, order embeddings perform better when the graphs are denser. For instance, when \( n=400 \) the performance (in terms of \( \mu(\tau) \)) of most strategies achieve tier 1 quality, compared with the

<table>
<thead>
<tr>
<th>Tier</th>
<th>None</th>
<th>Sparse</th>
<th>Moderate</th>
<th>Dense</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tier 1</td>
<td>None</td>
<td>Sparse</td>
<td>Moderate</td>
<td>Dense</td>
</tr>
<tr>
<td>Tier 2</td>
<td>Sparse, moderate, dense</td>
<td>Sparse, moderate, dense</td>
<td>Sparse, moderate, dense</td>
<td>Sparse, moderate, dense</td>
</tr>
</tbody>
</table>

Table 3: Tier-based Quality of Order Embeddings (\( n=400 \)).
Table 4: Sample Statistics and Performance Comparison (n=800, 10 Dense Graphs, 3-fold).

<table>
<thead>
<tr>
<th>Strategy</th>
<th>OE(d-2)</th>
<th>Train (OE)</th>
<th>Train Test (LR)</th>
<th>OE@WE</th>
<th>Train (OE)</th>
<th>Train Test (LR)</th>
<th>OE@WE</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>63</td>
<td>291.5</td>
<td>146.6</td>
<td>71.7</td>
<td>63</td>
<td>281.4</td>
<td>74.9</td>
</tr>
<tr>
<td>L1</td>
<td>63</td>
<td>222.2</td>
<td>146.6</td>
<td>71.8</td>
<td>63</td>
<td>237.2</td>
<td>78.9</td>
</tr>
<tr>
<td>L2</td>
<td>63</td>
<td>218.4</td>
<td>146.6</td>
<td>69.2</td>
<td>63</td>
<td>230.4</td>
<td>78.9</td>
</tr>
<tr>
<td>L3</td>
<td>63</td>
<td>255.3</td>
<td>146.6</td>
<td>70.4</td>
<td>63</td>
<td>257.9</td>
<td>78.9</td>
</tr>
<tr>
<td>L4</td>
<td>63</td>
<td>178.1</td>
<td>146.6</td>
<td>70.3</td>
<td>63</td>
<td>186.3</td>
<td>78.9</td>
</tr>
<tr>
<td>L5</td>
<td>63</td>
<td>179.7</td>
<td>146.6</td>
<td>69.4</td>
<td>63</td>
<td>185.2</td>
<td>78.9</td>
</tr>
<tr>
<td>L6</td>
<td>63</td>
<td>219.3</td>
<td>146.6</td>
<td>70.5</td>
<td>63</td>
<td>218.9</td>
<td>78.9</td>
</tr>
<tr>
<td>G1</td>
<td>63</td>
<td>219.9</td>
<td>146.6</td>
<td>70.9</td>
<td>63</td>
<td>231.5</td>
<td>78.9</td>
</tr>
<tr>
<td>G2</td>
<td>63</td>
<td>217.7</td>
<td>146.6</td>
<td>70.6</td>
<td>63</td>
<td>229.1</td>
<td>78.9</td>
</tr>
<tr>
<td>G3</td>
<td>63</td>
<td>255.4</td>
<td>146.6</td>
<td>70.7</td>
<td>63</td>
<td>257.4</td>
<td>78.9</td>
</tr>
<tr>
<td>G4</td>
<td>63</td>
<td>173.8</td>
<td>146.6</td>
<td>71.1</td>
<td>63</td>
<td>178.1</td>
<td>78.9</td>
</tr>
<tr>
<td>G5</td>
<td>63</td>
<td>175.4</td>
<td>146.6</td>
<td>70.9</td>
<td>63</td>
<td>178.5</td>
<td>78.9</td>
</tr>
<tr>
<td>G6</td>
<td>63</td>
<td>216.1</td>
<td>146.6</td>
<td>71.3</td>
<td>63</td>
<td>215.5</td>
<td>78.9</td>
</tr>
<tr>
<td>D0</td>
<td>63</td>
<td>18.1</td>
<td>146.6</td>
<td>61.1</td>
<td>63</td>
<td>20.0</td>
<td>78.9</td>
</tr>
<tr>
<td>D1</td>
<td>63</td>
<td>14.0</td>
<td>146.6</td>
<td>61</td>
<td>63</td>
<td>18.9</td>
<td>78.9</td>
</tr>
<tr>
<td>D2</td>
<td>63</td>
<td>13.3</td>
<td>146.6</td>
<td>61.4</td>
<td>63</td>
<td>18.8</td>
<td>78.9</td>
</tr>
<tr>
<td>D3</td>
<td>63</td>
<td>15.5</td>
<td>146.6</td>
<td>59.6</td>
<td>63</td>
<td>19.9</td>
<td>78.9</td>
</tr>
<tr>
<td>D4</td>
<td>63</td>
<td>13.6</td>
<td>146.6</td>
<td>61.1</td>
<td>63</td>
<td>18.6</td>
<td>78.9</td>
</tr>
<tr>
<td>D5</td>
<td>63</td>
<td>13.2</td>
<td>146.6</td>
<td>61.7</td>
<td>63</td>
<td>17.6</td>
<td>78.9</td>
</tr>
<tr>
<td>D6</td>
<td>63</td>
<td>15.3</td>
<td>146.6</td>
<td>60.5</td>
<td>63</td>
<td>19.7</td>
<td>78.9</td>
</tr>
<tr>
<td>DG1</td>
<td>63</td>
<td>14</td>
<td>146.6</td>
<td>61</td>
<td>63</td>
<td>18.9</td>
<td>78.9</td>
</tr>
<tr>
<td>DG2</td>
<td>63</td>
<td>13.3</td>
<td>146.6</td>
<td>61.4</td>
<td>63</td>
<td>18.1</td>
<td>78.9</td>
</tr>
<tr>
<td>DG3</td>
<td>63</td>
<td>15.6</td>
<td>146.6</td>
<td>60.2</td>
<td>63</td>
<td>19.9</td>
<td>78.9</td>
</tr>
<tr>
<td>DG4</td>
<td>63</td>
<td>13.7</td>
<td>146.6</td>
<td>60.9</td>
<td>63</td>
<td>18.4</td>
<td>78.9</td>
</tr>
<tr>
<td>DG5</td>
<td>63</td>
<td>13.1</td>
<td>146.6</td>
<td>61.4</td>
<td>63</td>
<td>17.5</td>
<td>78.9</td>
</tr>
<tr>
<td>DG6</td>
<td>63</td>
<td>15</td>
<td>146.6</td>
<td>61.3</td>
<td>63</td>
<td>19.7</td>
<td>78.9</td>
</tr>
<tr>
<td>A:S1+L1+D5</td>
<td>63</td>
<td>161.4</td>
<td>146.6</td>
<td>716</td>
<td>63</td>
<td>263.1</td>
<td>78.9</td>
</tr>
<tr>
<td>A:S1+L1+D5</td>
<td>63</td>
<td>183.7</td>
<td>146.6</td>
<td>726</td>
<td>63</td>
<td>237.1</td>
<td>78.9</td>
</tr>
<tr>
<td>A:S1+L1+D5</td>
<td>63</td>
<td>236.3</td>
<td>146.6</td>
<td>724</td>
<td>63</td>
<td>237.1</td>
<td>78.9</td>
</tr>
<tr>
<td>A:S1+L1+D5</td>
<td>63</td>
<td>233.9</td>
<td>146.6</td>
<td>714</td>
<td>63</td>
<td>238.2</td>
<td>78.9</td>
</tr>
<tr>
<td>A:S1+L1+D5</td>
<td>63</td>
<td>233.9</td>
<td>146.6</td>
<td>718</td>
<td>63</td>
<td>238.2</td>
<td>78.9</td>
</tr>
<tr>
<td>A:S1+L1+D5</td>
<td>63</td>
<td>218.1</td>
<td>146.6</td>
<td>704</td>
<td>63</td>
<td>182.6</td>
<td>78.9</td>
</tr>
</tbody>
</table>

Prediction Quality Measurement. To avoid training and testing sampling in favor of some particular representations, we collect the observed relations in $G_{tr}$, denoted as $E^p_{tr}$, and randomly select unobserved relations in $G_{tr}$, denoted as $E^u_{tr}$, where $|E^u_{tr}| = |E^p_{tr}|$. We conduct 5-fold cross validation to avoid biased data split for each training graph $G_{tr}$ of the same density level. Given a set of relation representations of particular form (WE, OE, OE@WE) from $E^p_{tr}$ and $E^u_{tr}$, we utilize logistic regression (LR) to learn a binary classifier to differentiate positive and negative dependencies. Specifically, we learn the linear function $p(y | e = (v_i, v_j))$, where $y = 1$ if $v_j$ depends on $v_i$, and $y = 0$ otherwise. We measure the prediction quality using accuracy as follows: $\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$.}

4.3 Dependency Relation Prediction

Given a training graph $G_{tr} = (V,E)$ with partial-observed relations and a testing graph $G_{ts}$ with a set of held-out relations, we aim to learn a dependency classifier from $G_{tr}$ to precisely predict dependency and non-dependency relations in $G_{ts}$.

**Representation Fusion and Dependency Classifiers.** We explore the expressive power of three types of representations: (i) semantic attributed representation (WE); (ii) order embedding (OE); and (iii) fused semantic order embeddings (OE@WE), where $\oplus$ is the concatenate operator. Semantic attributed representation is generated during the process of semantics relation generation, which is our simulation of the real-world word embeddings. Order embeddings are learned with respective negative sampling strategies. Given a representation of any type for each entity $v_i \in V$, we form the representation for each relation $(v_i, v_j)$ by taking subtraction $v_j$ from $v_i$.
in semantics, (iii) non-ordered but similar in semantics, and (iv) non-ordered and dissimilar in semantics. The error cases of OE-driven classifier, which is trained without the semantic knowledge, lie under quadrant ii, whereas the error cases of WE-driven classifier, which is trained without the ordering knowledge, lie under quadrant iii. The OE@WE-driven classifier can better differentiate cases lies under quadrants ii and iii because it is trained with both ordering and semantic knowledge, and thus outperforms others.

**Sample Ensemble Study.** Another key observation in Table 4 is that sample ensembles can effectively lead to better performance. For instance, S2 alone, which takes the reverse of the observed relations as negative samples, is obviously a winning strategy amongst all. Nonetheless, the overall predictive power can be further optimized with the sample ensemble technique, providing .724 accuracy compared to S2 at .72 in Table 4. This suggests that the negative samples contributed by S1, S2, L1, and DG2, not only increase in quantity (236.3 unique negative samples on average) but also provide diverse and useful signals, resulting in accurate order embeddings.

### 5 EXPERIMENTS ON REAL DATASETS

We conduct experiments on real-world networks for two purposes: (1) to quantitatively and qualitatively study the quality of order embedding, and (2) to demonstrate the usefulness of order embeddings for entity ranking and dependency relation prediction tasks.

#### 5.1 Datasets

We first describe thee real-world datasets for entity ranking task and dependency relation prediction task. The statistics of each network and their transitive closure is summarized in Table 2.

**Course Dependency Hierarchy (Course).** We use the course dependency dataset1 from 11 cs-related universities in US. There are 654 unique courses, and 861 dependency relations among courses. For each course, we extract the course title (TT) and course description (DE) from course content. After removing stop words, we obtain a vocabulary of 3,987 words.

**Singapore Organization Job Hierarchy (OrgJob).** There are 3,297 standardized job titles and 26,550 immediate reporting lines between supervisor and subordinate job titles from Singapore organization job hierarchy. We develop a job title parser to extract job function from each job title [18].

**ACM Citation Network (Citation).** There are 1.1M articles with at least one citation in the latest ACM citation network 2. We retrieve those articles in five different research areas: KDD, ICML, NIPS, WWW, and SIGIR, resulting in 5,859 papers and 60,759 citations. For each paper, we extract paper title (TT) as the textual content. After removing stop words, we obtain 6,548 vocabularies.

#### 5.2 Experiment Setup

**Representation Baselines.** We include several representation baselines for comparison in our experiments.

- Pre-Trained Word Embedding (WE): Glove vectors [21] are global vectors trained for word representations. For fair comparison, we use the vectors of length 50 dimensions trained on 6 billion words from Wikipedia corpus. Another widely used word vectors are the Skip-Gram-based Word2Vec [13]. We have self-trained vectors of 44,449 words and of 50 dimensions on our job posts corpus3 using the Skip-Gram model of Word2Vec. The job posts corpus is domain-focused and contains ~544,369 unique job posts.

- **Vector Order Embedding (OE):** We adopt vector order embedding (VOE) [24] to learn order embeddings for entities. The parameter settings of VOE includes learning rate $\lambda=0.01$, batch size = 256, and 80 epochs for all experiments.

- **Fusion (OE@WE):** We represent each entity using both word and order embeddings learned on each real-world dataset, based on various negative sampling strategies. The representations of entity $v_i \in V$ is represented as a concatenation of WE@OE. The resulting dimension of vector WE@OE consists of word embedding (50 by default) and order embeddings ($|D| \in [1, 3, 5]$).

#### 5.3 Entity Ranking

Given a training graph $G_{tr}=(V,E)$ with observed relations $E$ and unobserved relations $\bar{E}$, we aim to derive a ranked list of entities $V$ according to their order embeddings learned from $G_{tr}$.

**Course Dependency Hierarchy.** We evaluate the quality of order embedding by the entity ranking task. A valid dependency relation of a course pair $(v_i, v_j)$ represents course $v_i$ depends on course $v_j$, in which case $v_i$ is the relatively more advanced class and $v_j$ is the fundamental one. A course pair $(v_i, v_j)$ is regarded as positive if the dependency is observed in $G_{tr}$; otherwise, the course pair is considered as a negative one. Given the sample size $n$ and a sampling strategy, we sample $n$ positive relations $E^+_n$, and $n$ negative relations $E^-_n$ to learn order embeddings for each course. Table 5 shows the top-10 fundamental courses and bottom-10 advanced courses ranked by order embeddings using S2 strategy with the sample size $n=5000$. The top-10 fundamental courses tend to have higher in-degree and out-degree ratio (42.6 versus 0.1 on average), compared to the top-10 advanced courses (0.0 versus 7.0 on average). This suggests that top fundamental courses are truly prerequisite in the course dependency hierarchy and vice versa.

**Organization Job Hierarchy.** A valid dependency relation of a job pair $(v_i, v_j)$ represents career progression from job $v_i$ to $v_j$, in which case $v_i$ is the relatively junior to $v_j$ which is more senior. Given the sample size $n$ and a sampling strategy, we sample $n$ positive relations $E^+_n$, and $n$ negative relations $E^-_n$ to learn order embeddings for each job title. Table 5 shows the top-10 senior jobs and bottom-10 entry-level jobs in Singapore organization job hierarchy by order embeddings using S2 strategy with sample size $n=10K$. The top-10 senior jobs tend to have higher in-degree and out-degree ratio (1,579.6 versus 225.5 on average), compared to the top-10 entry-level jobs (0.8 versus 161.2). This suggests that top senior jobs are truly high ranking positions in the organization job hierarchy and vice versa.

**Citation Network.** A valid dependency relation of a paper pair $(v_i, v_j)$ represents paper $v_i$ cites $v_j$, in which case $v_i$ is the relatively earlier work and $v_j$ is the recent one. Given the sample size $n$ and a sampling strategy, we sample $n$ positive relations $E^+_n$, and $n$ negative relations $E^-_n$ to learn order embeddings for each paper. Table 5 shows the top-10 earliest work and bottom-10 latest work in ACM citation network by order embeddings using S2 strategy with sample size $n=175,000$. The top-10 earliest papers tend to

---

1https://github.com/harrylclc/concept-prerequisite-papers
2https://aminer.org/citation
3https://www.mycareersfuture.sg
Table 5: Top-10 and Bottom-10 Entities by Order Embeddings (d = 1, 2).

<table>
<thead>
<tr>
<th>Course Title</th>
<th>Job Title</th>
<th>Paper Title</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming Methodology</td>
<td>Prime Minister</td>
<td>AIR: A Retrieval Tool from the Basis of Morphological Analysis</td>
<td>1985</td>
</tr>
<tr>
<td>Intro to Computer Science</td>
<td>Minister</td>
<td>A Decision Theory Approaches to Optimal Automatic Indexing</td>
<td>1982</td>
</tr>
<tr>
<td>Introduction to Computer Programming</td>
<td>Trade and Industry Minister</td>
<td>Artificial Intelligence Implications for Information Retrieval</td>
<td>1983</td>
</tr>
<tr>
<td>Introduction to Programming Techniques</td>
<td>Infrastructure Minister</td>
<td>Evaluation of The 2-Poisson Model as a Basis for Using Term</td>
<td>1983</td>
</tr>
<tr>
<td>Computer Science I Fundamentals</td>
<td>Defence Minister</td>
<td>An Approach to Natural Language for Document Retrieval</td>
<td>1987</td>
</tr>
<tr>
<td>Introduction to EICS I</td>
<td>Permanent Secretary</td>
<td>An Evaluation of Term Dependence Models in Information Retrieval</td>
<td>1982</td>
</tr>
<tr>
<td>Programming Abstractions</td>
<td>Manpower Minister</td>
<td>IR, NLP, AI and UFOs: or IR-Relevance, Natural Language Problems</td>
<td>1986</td>
</tr>
<tr>
<td>Object-Oriented Programming I</td>
<td>Finance Minister</td>
<td>The Maximum Entropy Principle in Information Retrieval</td>
<td>1986</td>
</tr>
<tr>
<td>Discrete Structures</td>
<td>Education Minister</td>
<td>The Automatic Indexing System AIR/PHYS</td>
<td>1988</td>
</tr>
</tbody>
</table>

Table 6: Performance comparison on course dependency hierarchy (n=5K), where TT (DE) denotes course titles (descriptions).

<table>
<thead>
<tr>
<th>Strategy (OE)</th>
<th>Train (OE)</th>
<th>Train/Test LR</th>
<th>WE</th>
<th>OE(d=1)</th>
<th>OE(d=2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>1675</td>
<td>5000.0</td>
<td>3726.4</td>
<td>847.2</td>
<td>641.6</td>
</tr>
<tr>
<td>S2</td>
<td>1675</td>
<td>1675.6</td>
<td>3726.4</td>
<td>847.2</td>
<td>641.6</td>
</tr>
<tr>
<td>L1</td>
<td>1675</td>
<td>4732.0</td>
<td>3726.4</td>
<td>847.2</td>
<td>641.6</td>
</tr>
<tr>
<td>L2</td>
<td>1675</td>
<td>4951.4</td>
<td>3726.4</td>
<td>847.2</td>
<td>641.6</td>
</tr>
<tr>
<td>L3</td>
<td>1675</td>
<td>4908.4</td>
<td>3726.4</td>
<td>847.2</td>
<td>641.6</td>
</tr>
<tr>
<td>L4</td>
<td>1675</td>
<td>4609.6</td>
<td>3726.4</td>
<td>847.2</td>
<td>641.6</td>
</tr>
<tr>
<td>L5</td>
<td>1675</td>
<td>4949.6</td>
<td>3726.4</td>
<td>847.2</td>
<td>641.6</td>
</tr>
<tr>
<td>L6</td>
<td>1675</td>
<td>4895.4</td>
<td>3726.4</td>
<td>847.2</td>
<td>641.6</td>
</tr>
<tr>
<td>G1</td>
<td>1675</td>
<td>4715.8</td>
<td>3726.4</td>
<td>847.2</td>
<td>641.6</td>
</tr>
<tr>
<td>G2</td>
<td>1675</td>
<td>4950.0</td>
<td>3726.4</td>
<td>847.2</td>
<td>641.6</td>
</tr>
<tr>
<td>G3</td>
<td>1675</td>
<td>4902.0</td>
<td>3726.4</td>
<td>847.2</td>
<td>641.6</td>
</tr>
<tr>
<td>G4</td>
<td>1675</td>
<td>4580.4</td>
<td>3726.4</td>
<td>847.2</td>
<td>641.6</td>
</tr>
<tr>
<td>G5</td>
<td>1675</td>
<td>4946.2</td>
<td>3726.4</td>
<td>847.2</td>
<td>641.6</td>
</tr>
<tr>
<td>G6</td>
<td>1675</td>
<td>4983.3</td>
<td>3726.4</td>
<td>847.2</td>
<td>641.6</td>
</tr>
</tbody>
</table>

have more general topics and higher in-degree and out-degree ratio (471.9 versus 173.2 on average) in the ACM citation network, compared to the top-10 (0.8 versus 0.5).

5.4 Dependency Relation Prediction
We follow Section 4.3 to explore the expressive power of the following three types of representations: semantic attributed representation (WE), order embedding (OE), and fused semantic order.

212
embeddings (OE@WE). To predict whether an unseen antisymmetric relation \((\mathcal{v}_l, \mathcal{v}_j)\) holds or not, we train a dependency classifier using logistic regression which takes representations of entity pairs and the label of entity pairs as input. We report the classification accuracy using 5-fold cross validation.

**Representations Study.** Table 6 compares the performance of predictions made by various representations. An observation made from Table 6 is that the prediction accuracy using OE representation is better than using either WE or OE@WE. This suggests that OE is effective and reliable, especially when the textual content between a relevant entity pairs is very different. For example, “machine learning” course transitively depends on “data structure”, but nonetheless the dependencies do not necessarily reflect in the semantic similarity by course title (TT) or course description (DE). The expressive power of OE is also observed on Orgjob and Citation networks in Table 7, respectively.

**Dimensionality Study (d)**. Another observation in Table 6 is that the common sense of “the higher dimensionality the greater performance” does not necessarily hold. OE \((d = 3)\) does give a better performance than OE \((d = 1)\), but OE \((d = 3)\) is still superior to OE \((d = 5)\) in most cases as reported in Table 6. The order embedding method seeks to map each entity in a given hierarchy into a low dimensional vector with a fixed dimension. Unfortunately, we have no knowledge of the intrinsic dimensionality to describe a particular hierarchical structure. To unravel the true dimensionality, we therefore empirically extend our experiment to multiple configurations. Table 6 suggests that \(d = 3\) should suffice to preserve course dependency hierarchy.

**Subsampled and Ensemble Study.** A key observation throughout the synthetic and real datasets is that S1 consistently outperforms other strategies despite their simplicity and low overheads as reported in Table 6 and 7. In particular, the overall predictive power can be further optimized with the sample ensemble technique, providing .901 \((d=5)\) accuracy with 4.4% improvement compared to S2 at .863 \((d=5)\) in Table 6. This suggests that the negative samples contributed by S1, S2, L4, and D6 provides diverse and useful signals (4,685.2 unique negative samples on average), resulting in accurately learning order embeddings.

### 6 CONCLUSION AND FUTURE WORK

In this paper, we integrate the notions of order embedding and semantic proximity to model dependency relations. In practice, negative dependency relations/samples often are missing from the dependency relation graphs. To address this one-class problem, we explore multiple negative sampling strategies based on graph-specific centralities to collect diverse and precise negative information. We learn order embeddings from positive and negative training dependency relations by minimizing the max-margin loss on order-violation penalties. We conduct extensive experiments, using both synthetic and several real datasets, to demonstrate the usefulness of order embeddings and to gain better understanding of order embedding through the tasks of entity ranking and dependency relation prediction. Our studies show that: (i) the proposed negative sampling strategies enable effective and efficient order embedding learning, (ii) ensembles of diverse negative samples lead to robust and mostly better performance, (iii) while order embedding can give strong features, blended feature representations can contribute to better prediction accuracy when dependent entities share similar semantic. As for the future work, one direction is to extend the work to other order embedding models. Another interesting direction is to learn order embeddings for words to directly infer dependency relations.
ACKNOWLEDGMENT
This research is supported by the National Research Foundation, Prime Minister’s Office, Singapore under its International Research Centres in Singapore Funding Initiative. This work is supported in part by the National Science Foundation under Grant No. IIS-1717084.

REFERENCES
[10] Chen Liang, Jianbo Ye, Zhaohui Wu, Bart Pursel, and C Lee Giles. 2017. Recovering Concept Prerequisite Relations from University Course Dependencies. In AAAI.
[19] Liangming Pan, Chengjiang Li, Juanzi Li, and Jie Tang. 2017. Prerequisite relation learning for concepts in moocs. In ACL.
[20] Liangming Pan, Xiaochen Wang, Chengjiang Li, Juanzi Li, and Jie Tang. 2017. Course Concept Extraction in MOOCs via Embedding-Based Graph Propagation. In IJCNLP.
[23] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. In WWW.