A Scalable Approach to Aligning Natural Language and Knowledge Graph Representations

Batched Information Guided Optimal Transport

Alexander Kalinowski, Deepayan Datta, Yuan An

November 10, 2023

Outline

Representation Learning

- Paradigms for information representation
  - How to encode textual information in a computer readable way
  - How to connect and reason over such information
- The curse of dimensionality: large datasets, long computations
- Need to compress information to a smaller subspace
  - Neural network models for information **embedding**
  - *Today: comparing semantic information from these models across data domains*
- **To what extent do embeddings of data sources expressing the same 'semantic information' exhibit similar structural patterns when projected to low-dimensional spaces?**

Approach

- Let $\mathcal{C}$ be a corpus of NL sentences $\mathcal{C} = \{l_1, \ldots, l_n\}$.
- Let $\mathcal{G} = \{E, P, T\}$ be a KG featuring a set of entities E, a set of predicates P and a set of triples $T = \{t_1, \ldots, t_k\}$, $t_k = \{s, p, o\}$, where $s, o \in E$ and $p \in P$.
- Define two *embedding functions* f and g for $\mathcal{C}$ and $\mathcal{G}$, respectively, that create latent vector representations based on the input data, i.e. $f(l_i) = u_i$ and $g(t_j) = v_j$.
- Find an *optimal transportation plan* $\mathbf{P}$ that maps elements of $V_{\mathcal{C}}$ to elements of $V_{\mathcal{G}}$

Approach

- Let $\mathcal{C}$ be a corpus of NL sentences $\mathcal{C} = \{l_1, \ldots, l_n\}$.
- Let $\mathcal{G} = \{E, P, T\}$ be a KG featuring a set of entities E, a set of predicates P and a set of triples $T = \{t_1, \ldots, t_k\}$, $t_k = \{s, p, o\}$, where $s, o \in E$ and $p \in P$.
- Define two *embedding functions* f and g for $\mathcal{C}$ and $\mathcal{G}$, respectively, that create latent vector representations based on the input data, i.e. $f(l_i) = u_i$ and $g(t_j) = v_j$.
- Find an *optimal transportation plan* $\mathbf{P}$ that maps elements of $V_{\mathcal{C}}$ to elements of $V_{\mathcal{G}}$

Terminology I

Definition

Generally, an **embedding** is a map between two spaces $f: \mathcal{X} \to \mathcal{Y}$ in an injective, structure preserving manner. Typically, these spaces are viewed as Euclidean spaces $\mathcal{X} \subset \mathbb{R}^n$, $\mathcal{Y} \subset \mathbb{R}^m$, where m is orders of magnitude smaller than n. We can think of the embedding function f as a method for *compressing information* into more manageable subspaces. [Kenyon-Dean et al., 2020]

Sentence Embedding Generation

Table: Embedding Methods for Natural Language Sentences

Embedding	Dimensionality
Random	300
GloVe-Mean	300
GloVe-DCT 1	300
GloVe-GEM	300
SentBert	768
LASER	1024
OpenAI ADA	1536
OpenAI ADA + PCA	1536

Terminology II

Definition

A **knowledge graph** is a 3-tuple $G = (E, P, T)$, where E is a set of entities, P is a set of predicates, and T is a set of triples $\langle h, p, t \rangle$ connecting a head entity $h \in E$ to a tail entity $t \in E$ by a predicate $p \in P$ as an edge. [Nickel et al., 2016]

Definition

A knowledge graph is **multigraph** in that it permits different parallel predicates between pairs of entities. [Hamilton et al., 2017]

Definition

A **knowledge graph embedding** method represents entities $v \in E$ as d-dimensional vectors $\mathbf{e}_v \in \mathbb{R}^d$ and predicates $r \in P$ as a scoring functions $f_r : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. [Nickel et al., 2016]

Taxonomy of KGE methods

- Translation-based
  - **TransE** ($v_h + v_p \approx v_t$)
  - **RotatE** ($-\|v_h \circ v_p - v_t\|$)
- Semantic-matching/factorization
  - **RESCAL** $f_k(h,t) = h^\intercal P_k t$, $X_k \approx A P_k A^\intercal$
  - **DistMult** restricts P_k to diagonal entries
  - **ComplEx** $f_k(h,t) = \mathcal{R}(\langle h, p_k, t \rangle)$, where $p_k \in \mathcal{C}^r$
- Deep neural networks
  - **ConvE** utilizes 2-d reshaping and convolution operator
- Graph-walk
  - **node2vec** preserves graph structure via random walks
  - **Triple2vec** closest benchmark to our work

Pre-trained models
Optimized parameters
Transparent, open code
Reduced carbon cost!

Triple Embedding Aggregations

Table: A summary of aggregation operators for constructing triple representations [Grover and Leskovec, 2016]

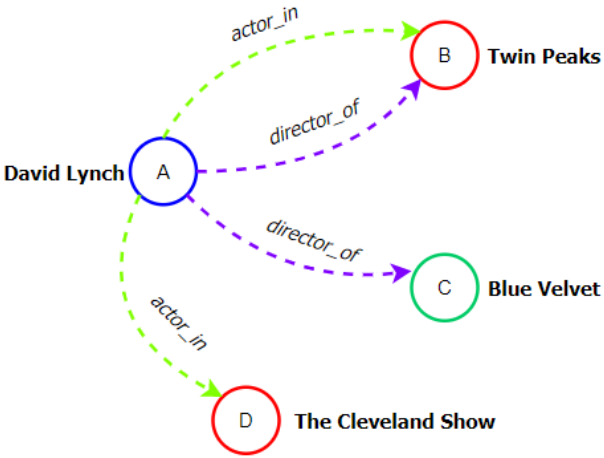
Operator	Definition
Average (AVG)	$[f(\mathbf{u}) \star f(\mathbf{v})]_i = \frac{f_i(\mathbf{u})+f_i(\mathbf{v})}{2}$
Hadamard (HAD)	$[f(\mathbf{u}) \star f(\mathbf{v})]_i = f_i(\mathbf{u}) * f_i(\mathbf{v})$
Weighted-L1 (L1)	$\lvert f_i(\mathbf{u}) - f_i(\mathbf{v})\rvert$
Weighted-L2 (L2)	$\lvert f_i(\mathbf{u}) - f_i(\mathbf{v})\rvert^2$
Concatenation (HT)	$[f(\mathbf{u}) \star f(\mathbf{v})]_i = f_i(\mathbf{u}) \Vert f_i(\mathbf{v})$

Notes:

'$*$' denotes element-wise product operation.

'$\Vert$' denotes vector concatenation operation.

Issues with KGE Aggregation



Entity Embeddings	A = [w1, ..., wk]
	B = [x1, ..., xk]
	C = [y1, ..., yk]
	D = [z1, ..., zk]
Predicate Embeddings	R1 = *actor_in*
	R1 = [q1, ..., qk]
	R2 = *director_of*
	R2 = [s1, ..., sk]

Determining Triple Similarity

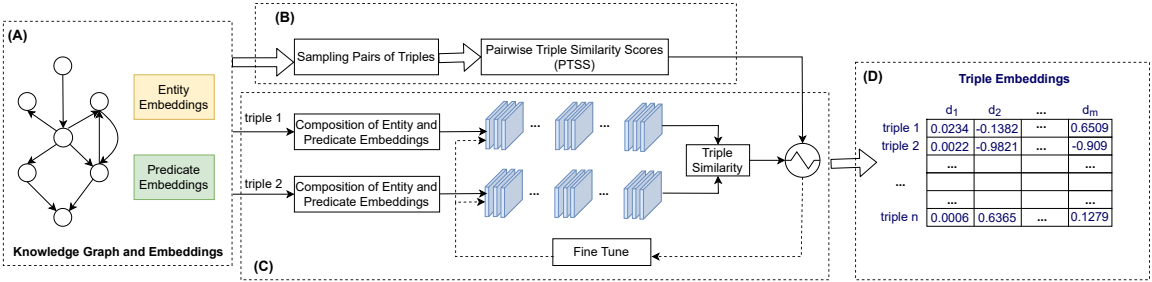
- Given two triples $T_a = \langle h_a, p_a, t_a \rangle$ and $T_b = \langle h_b, p_b, t_b \rangle$ in the knowledge graph, we define the *pairwise triple similarity score (PTSS)* between T_a and T_b as an average of the similarities between the embeddings of their head entities, tail entities, and predicates, respectively.

$$PTSS(T_a, T_b) = avg(sim(\mathbf{e}_{h_a}, \mathbf{e}_{h_b}), sim(\mathbf{e}_{p_a}, \mathbf{e}_{p_b}), sim(\mathbf{e}_{t_a}, \mathbf{e}_{t_b}))$$

[Kalinowski and An, 2022]

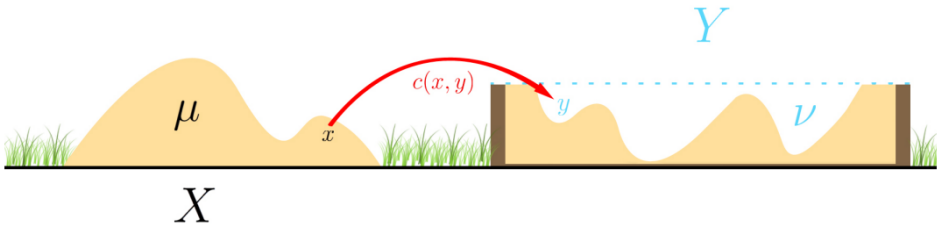
- The similarity function $sim(\mathbf{e}_1, \mathbf{e}_2)$ could be an arbitrary function capturing the geometric structures in KG embedding spaces.
- Extract from the knowledge graph a set of training examples containing both *"positive"* and *"negative"* examples.
  - Select N other triples with the same head entity, N other triples with the same tail entity, N other triples with the same predicate.
  - For negative examples, we select N other triples with no commonalities in any available slot.

PTSS Architecture



Repurposing Knowledge Graph Embeddings for Triple Representation via Weak Supervision [Kalinowski and An, 2022]

The Monge Problem



Generalization of Kantorovich Duality to multi-marginal optimal transportation and applications in economics [Vogler and Friesecke, 2019]

Defining Optimal Transport

Definition

Let $\mathcal{X}$ and $\mathcal{Y}$ be two arbirtary spaces with corresponding probability measures α and β. The transport map $T : \mathcal{X} \to \mathcal{Y}$ is defined to minimize

$$\min_{T}\{\int_{\mathcal{X}} c(x, T(x))d\alpha(x) : T_{\#}\alpha = \beta\}$$

where c is the cost matrix $C_{i,j} = c(x_i, y_i)$ and $T_{\#}$ is a push-forward operator that preserves the mass from α to β. [Peyré and Cuturi, 2020]

Kantorovich Relaxation

Definition

Let $P_{i,j} \in \mathbb{R}^{n \times m}$ be a probabilistic coupling matrix that defines the amount of mass flowing between x_i and y_j. Let $C_{i,j} = c(x_i, y_i)$ again be the cost function. The set of maps between the two spaces is then defined as

$$U(a, b) = \{P \in \mathbb{R}^{n \times m} : P\mathbf{1}_m = a, P\mathbf{1}_n = b\}$$

The Kantorovich optimal transport problem is then defined as

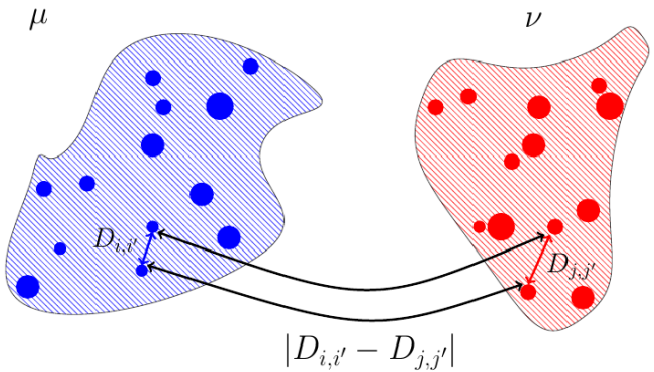
$$L_C(a, b) = \min_{P \ in U(a,b)} \langle C, p \rangle = \Sigma_{i,j} C_{i,j} P_{i,j}$$

[Peyré and Cuturi, 2020]

Optimal Transport and Distances

- In many cases, the cost matrix C defines a distance D between probability spaces
- $W_p(a, b) = L_{D^p}(a, b)^{\frac{1}{p}}$ then defines the **p-Wasserstein distance** [Peyré and Cuturi, 2020]
- Optimal transport (solved through Sinkhorn iterations) requires some definition of the cost to converge, i.e. **pre-regristration** of the spaces $\mathcal{X}$ and $\mathcal{Y}$
- Pre-registration between embedding spaces would require us to have labelled pairs
  - This may not be available or possible in knowledge discovery
  - Would like a more unsupervised-friendly approach

Gromov-Wasserstein Distances



$$D \in \mathbb{R}^{n \times n}, D' \in \mathbb{R}^{m \times m}$$

[Peyré and Cuturi, 2020]

Minibatch OT

Algorithm 1 Minibatched Barycentric Mapping

Input: batch size k, embedded source space X_S, embedded target space X_T;

Output: updated source space $\tilde{Y}_S$, updated target space $\tilde{Y}_T$;

for $t = 1, \ldots, k$ **do**

Select a source batch A from X_S;

Select a target batch B from X_T;

Compute the cost matrix C from A and B;

Compute the batched transportation plan $G \leftarrow \underset{\Pi \in U(A,B)}{\arg\min} \langle C_{A,B} \rangle$;

Update $\tilde{Y}_S|_A \leftarrow \tilde{Y}_S|_A + G.X_T|_B$;

Update $\tilde{Y}_T|_B \leftarrow \tilde{Y}_T|_B + G^T.X_S|_A$;

end for

return $\frac{n}{k}\tilde{Y}_S,\ \frac{m}{k}\tilde{Y}_T$

Information-fused OT

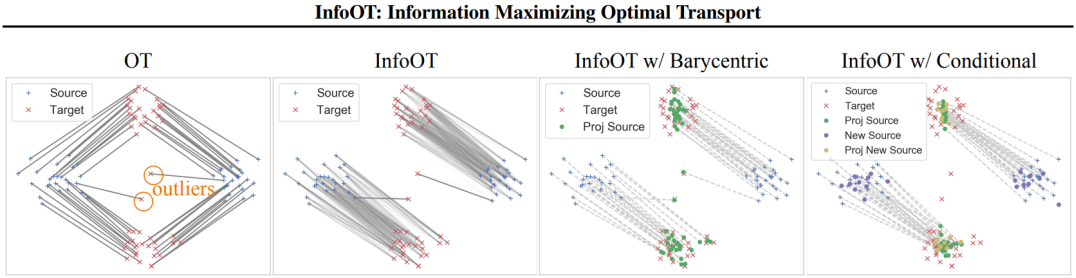


Figure 1: Illustration of InfoOT on 2D point cloud. Compared to classic OT, InfoOT preserves the cluster structure, where the source points from the same cluster are mapped to the same target cluster. For projection estimation (dashed lines), the new conditional projection improves over barycentric projection with better outlier robustness and out-of-sample generalization.

Information Maximizing Optimal Transport [Chuang et al., 2023]

BIG-OT

Algorithm 2 Batched Information Guided Optimal Transport (BIG-OT)

Input: batch size k, embedded source space X_S, embedded target space X_T, number of guides n_guides, best_guides $= \{\}$;

Output: optimal transportation plan G;

for $j = 1, \ldots,$ n_guides **do**
  Randomly choose k samples from $X_s \to \tilde{X}_S$;
  Randomly choose k samples from $X_T \to \tilde{X}_t$;
  Pre-compute the cost matrices $\tilde{C}_s, \tilde{C}_t$ from $\tilde{X}_s$ and $\tilde{X}_t$;
  Compute the GW-distance using the pre-computed costs: $\mathcal{GW}(\tilde{C}_s, \tilde{C}_t) = d_j$;
  Update best_guides$[d_j] = [\tilde{X}_s, \tilde{X}_t]$;
end for
Find minimum $d_j \to$ best_distance;
Solve $G \leftarrow$ InfoOT(best_guides[best_distance]);
return G

Distant Supervision Benchmark

Table: Summary Statistics of RE Datasets

Dataset	Train	Test	Relations	Sentences	Triples
NYT-FB	455,771	172,448	53	320,648	378,511
Wikidata	372,059	360,334	353	856,217	372,059

1. Results measured using the precision@10 metric
2. Current SOTA: KGPool [Nadgeri et al., 2021] (P@10 92.3)

Hyperparameters and Compute

- Eight sentence embedding methods (8)
- Six knowledge graph embedding models, with and without PTSS post-processing (12)
- Initial entropic regularization 0.01
- $n = [5, 10, 20, 50, 100]$ guidebooks, with entropic backoff (increase entropy by 1.1 when GW fails to converge) (5)
- Batch size $\in [128, 256, 512, 1024, 2048]$ (5)
- Ten random seeds per trial (10)
- Total experiments: **24,000**

NTY-FB Results

Table 5.7 Results (P@10) for the entire suite of BIG-OT experiments on the NYT-FB dataset. Best values in each column are highlighted in bold. The checkmarks (✓) indicate the use of PTSS, the x-marks (✗) indicate the use of the aggregation via concatenation. Values have been truncated to three decimal places and represent the average across 10 trials.

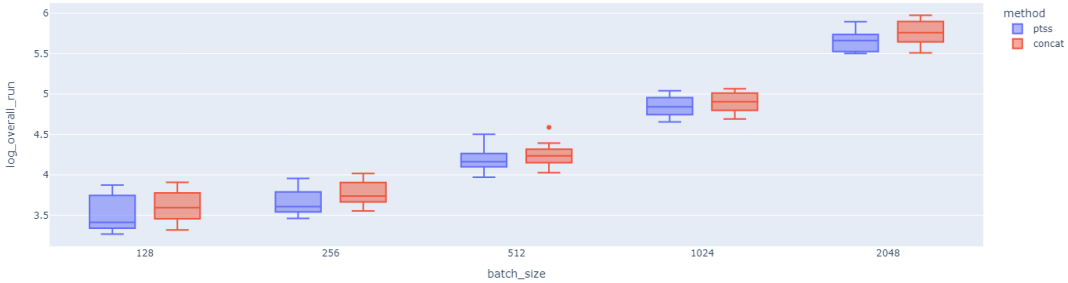
Model	ComplEx		ConvE		DistMult		RESCAL		RotatE		TransE	
PTSS	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓
Random	4.50	8.73	5.59	8.56	7.88	8.50	3.91	8.64	4.27	8.44	3.65	3.23
GloVe-mean	84.4	87.0	**85.5**	86.8	76.0	87.2	83.7	96.8	83.9	97.2	83.1	85.9
GloVe-DCT	84.5	87.1	85.3	86.1	77.8	87.0	83.8	86.9	84.2	86.3	83.4	86.8
GloVe-GEM	84.3	96.4	85.0	95.1	83.0	**96.8**	**93.8**	**96.9**	84.1	96.3	83.1	96.0
SentBert	**84.6**	**96.8**	85.3	**96.8**	**91.9**	96.5	93.7	96.8	**84.3**	**96.8**	82.7	**96.7**
LASER	84.5	89.4	85.5	84.2	81.5	85.5	83.8	86.3	84.2	86.9	**83.4**	85.7
Ada	74.6	76.9	75.6	76.7	71.3	77.2	73.8	77.2	74.2	77.2	73.4	76.3
Ada+ABT	74.4	77.3	75.6	76.0	69.0	77.0	73.8	75.8	74.8	77.1	73.8	76.3

Benchmark Comparison

Table: Comparison of the closest benchmark (KGPool [Nadgeri et al., 2021]) to the best model from our approach (BIG-OT) on the NYT-FB dataset. Models are compared on the precision at 10 metric (P@10).

Model	P@10
KGPool+gnn ($\alpha = 1$)	90.1
KGPool+gnn ($\alpha = 2$)	91.0
KGPool+gnn ($\alpha = 3$)	92.3
KGPool+gnn ($\alpha = 4$)	90.6
BIG-OT (best)	96.8

NYT-FB Run Time



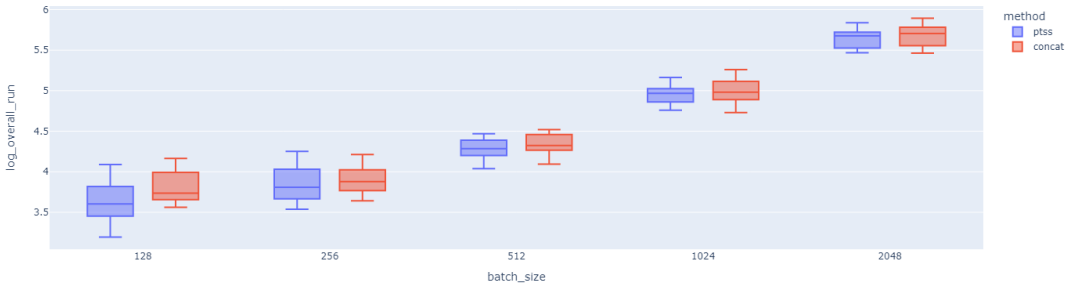
BIG-OT with PTSS achieved average accuracy of 96.8 in an average of 28.5 minutes, while concatenation performed at 93.6 in an average of 41.3 minutes. KGPool, each epoch of training their model took on the order of 51 minutes and was trained for 100 epochs for a best P@10 of 92.3.

Wikidata Results

Table 5.8 Results (P@10) for the entire suite of BIG-OT experiments on the Wikidata dataset. Best values in each column are highlighted in bold. The checkmarks (✓) indicate the use of PTSS, the x-marks (✗) indicate the use of the aggregation via concatenation. Values have been truncated to three decimal places and represent the average across 10 trials.

Model	ComplEx		ConvE		DistMult		RESCAL		RotatE		TransE	
PTSS	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓
Random	4.08	6.56	6.52	6.15	2.91	6.14	3.77	6.15	2.41	6.11	3.91	4.90
GloVe-mean	93.8	95.3	91.4	95.7	93.5	96.0	91.2	96.1	92.9	96.1	94.5	94.9
GloVe-DCT	90.2	95.1	89.7	94.8	91.7	94.7	90.5	95.3	92.0	95.6	92.9	93.2
GloVe-GEM	**93.9**	96.1	91.0	**96.4**	93.2	**96.2**	**91.4**	96.1	92.9	**96.0**	**94.8**	95.3
SentBert	93.8	96.2	**91.6**	96.2	**93.4**	95.9	91.2	95.4	**92.9**	95.8	94.5	**95.9**
LASER	93.0	**96.6**	91.0	94.3	93.2	96.1	90.9	**96.5**	92.5	95.9	93.8	95.8
Ada	87.5	90.3	88.4	91.2	92.4	94.3	89.2	95.6	88.4	93.4	91.6	94.3
Ada+ABT	87.2	90.2	88.2	90.8	91.6	93.5	88.3	94.9	88.1	92.7	91.0	93.7

Wikidata Run Time



Using batch size 2048 yielded 92.4 P@10 in 31.5 minutes using concatenation, 95.9 in 28.7 minutes with PTSS. The run-time per epoch for Wikidata was also around 51 minutes, meaning that our slowest models ran in $\frac{1}{200}$th of the time.

Summary of Findings

- BIG-OT was most successful with largest batch size and smallest number of guidebooks
- Utilizing PTSS KG embeddings improved both speed and P@10
- Performance gains over closest benchmark (96.8 versus 92.3)
- Performance gains are achieved using no supervised learning pairs in a fraction of the time

Chuang, C.-Y., Jegelka, S., and Alvarez-Melis, D. (2023).
Infoot: Information maximizing optimal transport.

Grover, A. and Leskovec, J. (2016).
Node2vec: Scalable feature learning for networks.
In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 855–864, New York, NY, USA. Association for Computing Machinery.

Hamilton, W. L., Ying, R., and Leskovec, J. (2017).
Representation learning on graphs: Methods and applications.

Kalinowski, A. and An, Y. (2022).
Repurposing knowledge graph embeddings for triple representation via weak supervision.
In *2022 International Conference on Intelligent Data Science Technologies and Applications (IDSTA)*, pages 129–137.

Kenyon-Dean, K., Newell, E., and Cheung, J. C. K. (2020).
Deconstructing word embedding algorithms.

CoRR, abs/2011.07013.


Nadgeri, A., Bastos, A., Singh, K., Mulang', I. O., Hoffart, J., Shekarpour, S., and Saraswat, V. (2021).
Kgpool: Dynamic knowledge graph context selection for relation extraction.


Nickel, M., Murphy, K., Tresp, V., and Gabrilovich, E. (2016).
A review of relational machine learning for knowledge graphs.
Proceedings of the IEEE, 104(1):11–33.


Peyré, G. and Cuturi, M. (2020).
Computational optimal transport.


Vogler, D. and Friesecke, G. (2019).
Generalization of kantorovich duality to multi-marginal optimal transportation and applications in economics.